



CENTRO INTERNACIONAL DE ESTUDOS  
DE DOUTORAMENTO E AVANZADOS  
DA USC (CIEDUS)

TESIS DE DOCTORADO

# **FROM LINE MATCHING TO 3D SCENE ABSTRACTION**

Roi Santos Mateos

ESCUELA DE DOCTORADO INTERNACIONAL

PROGRAMA DE DOCTORADO EN INVESTIGACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

SANTIAGO DE COMPOSTELA

2019





## DECLARACIÓN DEL AUTOR DE LA TESIS

**From line matching to 3D scene abstraction**

D. Roi Santos Mateos

*Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:*

- 1) *La tesis abarca los resultados de la elaboración de mi trabajo.*
- 2) *En su caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
- 3) *La tesis es la versión definitiva presentada para su defensa y coincide con la versión enviada en formato electrónico.*
- 4) *Confirmando que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*

*En Santiago de Compostela, 28 de Enero de 2019*

Fdo. Roi Santos Mateos





## AUTORIZACIÓN DEL DIRECTOR / TUTOR DE LA TESIS

**From line matching to 3D scene abstraction**

D. Xose M. Pardo López  
D. Xose R. Fernández Vidal

INFORMAN:

*Que la presente tesis, corresponde con el trabajo realizado por D. Roi Santos Mateos , bajo nuestra dirección, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores de ésta no incurre en las causas de abstención establecidas en Ley 40/2015.*  
*En Santiago de Compostela, 28 de Enero de 2019*

Fdo. Xose M. Pardo López

Fdo. Xose R. Fernández Vidal





UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Centro de Investigación en Tecnoloxías da Información

Tesis doctoral

## **FROM LINE MATCHING TO 3D SCENE ABSTRACTION**

Presentada por:

**Roi Santos Mateos**

Dirigida por:

**Xose M. Pardo López,**

**Xose R. Fernández Vidal**

January 28th, 2019



**Xose M. Pardo López**, Profesor Titular de Universidad del Área de Lenguajes y Sistemas  
Informáticos de la Universidad de Santiago de Compostela

**Xose R. Fernández Vidal**, Profesor Titular de Universidad del Área de Física Aplicada de la  
Universidad de Santiago de Compostela

**HACEN CONSTAR:**

Que la memoria titulada **FROM LINE MATCHING TO 3D SCENE ABSTRACTION** ha sido realizada por **D. Roi Santos Mateos** bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información de la Universidad de Santiago de Compostela, y constituye la Tesis que presenta para obter al título de Doctor.

January 28th, 2019

**Director**

Xose M. Pardo López

**Director**

Xose R. Fernández Vidal



**A mis padres.**



*Imagine, if you will, nothing.*

*Now imagine that it's endless.*

*Now triple that.*

Doug DeMuro



## **Agradecimientos**

Agradezco a todas las personas que directa o indirectamente han contribuido a la realización de esta tesis.

En el Citius, mis directores de tesis Pardo y Xose por buscar tiempo siempre para revisar mis artículos y enviarme las correcciones. Estoy seguro que sin ellos no habría llegado a publicar el artículo de reconstrucción 3D basado en líneas. No me puedo olvidar de la relación cordial que he mantenido durante estos años con otros IPs del Citius como Roberto, Diego Cabello, Caba, Víctor y Fran. Gracias a Félix y María por ayudarme a preparar el Doctoral Meeting. A Diego y las Rosas de secretaría. Mis compañeros de mesa Alberto y Álvaro, y demás compañeros que he tenido en el laboratorio durante estos años: Lebo, Ahmed, María, Chema, Estéban, Rodrigo, Nico, Dani, Mauro, Julio y Xose. A Osama por sentarse conmigo a depurar mi código aquella tarde de sábado.

Querá agradecer a la Unión Europea las numerosas oportunidades que me ha ofrecido durante mi vida, desde mi beca Erasmus en mi último año de Física hasta parte de la financiación que he recibido durante mi estancia en el Citius. Comparto los valores de la UE y me siento orgulloso de ser europeo. También quiero agradecer al Ministerio de Educación, Cultura y Deporte, a la Xunta de Galicia y a la USC por su apoyo durante el tiempo de realización del trabajo que ha derivado en esta tesis.

Durante el primer año de doctorado, mientras desarrollaba el código de reconstrucción 3D, tuve la oportunidad de trabajar con un equipo excelente en un campo totalmente nuevo

para mí. Quería agradecer a todo el equipo de S4SD. A Ángel por sentarse conmigo a enseñarme tanto sobre calorimetría, binding molecular y algoritmos de simulación, estadística y optimización, a Eva y Juan por su compañía durante esos meses. A Fernando de Babcock por enseñarme sobre calibración de cámaras en el Citius.

Quiero agradecer a todas las personas que me han ayudado a ver que sí podía terminar Física, y que después sí que iba a merecer la pena. Los profesores y compañeros que he tenido en Galicia, Bratislava, Lisboa, Badajoz, Viena y Bruselas. Muchas gracias a Stanimir Valtchev de la Universidade Nova de Lisboa por su apoyo. Gracias a Helmut Dannerbauer. A todos los que se han sentado a aprender cosas conmigo durante mis estancias tras graduarme. Gracias al profesor Vladimír Černý y Vladimír Balek por sus clases durante mi Erasmus.

Gracias a mi amigo Andrés por tantas experiencias que compartimos. Y a mi compañero de piso en Viena, Alex.

Debo recordar también a otros compañeros con los que no he podido compartir tanto tiempo como hubiera deseado, como los que conocí en el congreso CIARP Gastón, Takeshi y Yunbin.

Por encima de todo, gracias a mis padres, porque sin ellos nada habría sucedido.

January 28th, 2019

# Contents

<b>Notations</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Motivation . . . . .	3
1.2 Challenges . . . . .	5
1.3 Objectives of the work behind this thesis . . . . .	19
1.4 Contributions of this thesis . . . . .	19
1.5 Organization of the thesis . . . . .	22
<b>2 Background</b>	<b>23</b>
2.1 Detection of primitives . . . . .	26
2.2 Matching of primitives . . . . .	26
2.3 3D reconstruction . . . . .	27
<b>3 Related works</b>	<b>31</b>
3.1 Detection of lines in images . . . . .	33
3.2 Line matching . . . . .	38
3.3 3D abstraction based on straight segments . . . . .	41
<b>4 Straight segment detection</b>	<b>47</b>

4.1	Overview of the Line Detection method . . . . .	49
4.2	Energy based edge detection . . . . .	52
4.3	Edge Detection . . . . .	57
4.4	Straight line segments detection . . . . .	59
4.5	Experimental results for line detection . . . . .	65
<b>5</b>	<b>Straight line segment matching</b>	<b>79</b>
5.1	High level overview . . . . .	80
5.2	Low level description . . . . .	81
5.3	Experimental results . . . . .	106
5.4	Applications for <i>SALM</i> . . . . .	122
<b>6</b>	<b>3D abstraction based on lines</b>	<b>123</b>
6.1	High level overview for the <i>3DwSkt</i> 3D sketching method. . . . .	124
6.2	Low level description of the <i>3DwSkt</i> method. . . . .	128
6.3	Line intersections . . . . .	137
6.4	Experimental results of 3D sketch generation . . . . .	142
6.5	Applications for the 3D sketching method <i>3DwSkt</i> . . . . .	157
	<b>Conclusions</b>	<b>159</b>
	<b>Bibliography</b>	<b>163</b>
	<b>List of Figures</b>	<b>183</b>
	<b>List of Tables</b>	<b>190</b>

# Notations

**Table 0.1:** Table of Notations used in this thesis

---

$\mathbb{R}^2$	$\triangleq$	Euclidean 2D plane
$\mathbb{R}^3$	$\triangleq$	Euclidean 3D space
$x$	$\triangleq$	2D vector in $\mathbb{R}^2$
$\mathbf{x}$	$\triangleq$	3D vector in $\mathbb{R}^3$
$\mathbf{A}$	$\triangleq$	Matrix
$l_i$	$\triangleq$	2D straight line segment (projection of a 3D segment in image) with index $i$ .
$\Gamma_i$	$\triangleq$	3D straight line segment with index $i$ .
$Y^j$	$\triangleq$	Camera plane with index $j$ .
$\mathcal{R}$	$\triangleq$	Set of intersections of coplanar lines, projected in 3D.
$\mathbf{P}$	$\triangleq$	Projection matrix.
$\mathbf{K}$	$\triangleq$	Camera calibration matrix.
$\mathbf{R}$	$\triangleq$	Camera rotation matrix.
$\mathbf{t}$	$\triangleq$	Camera translation vector.
$\mathbf{E}$	$\triangleq$	Essential matrix.
$\mathbf{F}$	$\triangleq$	Fundamental matrix.

---

**Table 0.2:** Table of acronyms used in this thesis

---

<i>PCA</i>	$\triangleq$	Principal Component Analysis.
<i>kNN</i>	$\triangleq$	k Nearest Neighbors.
<i>DLT</i>	$\triangleq$	Direct Linear Transformation.
<i>RANSAC</i>	$\triangleq$	Random Sample Consensus.
<i>SBA</i>	$\triangleq$	Sparse Bundle Adjustment.
<i>SLAM</i>	$\triangleq$	Simultaneous Localization and Mapping.
<i>RMS</i>	$\triangleq$	Root Mean Squared.
<i>GNSS</i>	$\triangleq$	Global Navigation Satellite System.
<i>ALS</i>	$\triangleq$	Airborne Laser Scanning.



# Chapter 1

## Introduction

### 1.1 Motivation

Implementing environment comprehension into machines is a challenge for mankind. The ability to fetch, classify and interrelate the perceptible elements captured by cameras is the link between our global net of cameras and the Artificial Intelligence. Therefore, the next giant leap will occur when the digital captures of the outside world are observed by neural networks, without requiring a human interpreter and translator. In order to make pictures understandable by machines, these have to be reduced to atomic describable concepts like points, lines or ellipses. The most simple forms, like points or lines, are referred to as primitives. The most traditional techniques detect primitives and classify them attending to their apparent attributes. During these early stages many problems had to be solved, originated by limitations of the digital technology, the similarity between primitives in the same image, the impossibility of characterizing them unequivocally, and the nature of the capture of light with changes on illumination or contrast. These approaches for description and matching of primitives have been developed in parallel with spatial abstraction methods, in such a way that nowadays it is common to derive from a series of pictures an unique 3D representation

including estimations for some of the captured primitives with the relative position and orientation of the cameras. This memory is focused on a single kind of primitives: the **straight line segments**. It goes through straight segment matching between images and the other operations that lead to the creation of 3D representations from these detected primitives. Straight line segments are frequently found in captures of man-made environments. The inclusion of straight lines in 3D representations provide structural information about the captured shapes and their limits, such as the intersection of planar structures.

Photogrammetry is an accurate, inexpensive, and non-contact metrology technique which involves estimating the three-dimensional coordinates of points on an object from multiple overlapping photographs taken from different poses. It has many different applications for unmanned vehicles, computer vision, robotics, measurement techniques, remote sensing or cartography. Photogrammetry is taking the place of other large scale metrology systems that have been developed during the last decades. such as coordinate measuring machines [51], LIDAR [25] or optical scanners [106]. Unlike photogrammetry, these other systems are either expensive or impracticable in many industrial environments, such as shipbuilding or in the in-place repairing of windmill turbine blades, where very large pieces are handled [27] under uncontrolled illumination conditions.

One of the key steps to obtain three-dimensional coordinates of points from multiple images is to identify homologous features, such as points or lines, across the different views. To achieve a reliable feature matching, most of the industrial photogrammetric systems usually require to manually place a set of retro-reflective landmarks on the surface of the object and control the illumination conditions. These retro-reflective targets are often illuminated by a light source near the lens, in order to produce images with a high contrast between the target and the background. Moreover, the combination of low aperture and high shutter speed virtually suppresses background details allowing a reliable segmentation of the targets [24]. The main drawback of the use of targets is the need to attach them to the object surface in such numbers that the presence of a target is required wherever a measurement is required. This

manual placement of targets becomes complicated and expensive in many industrial environments with difficult access, or where large pieces are handled. On the other hand, controlled lighting is difficult to set in many outdoor settings when large focal lengths are handled and where the light conditions can not be controlled.

## 1.2 Challenges

Research about straight segments have always been developed after works related to feature points. Lines have often been left as a complement for applications of these works devoted to feature points. There are reasons for the line based SfM to be more complex than the point based one:

1. Detection of points is restrained to sole coordinates in images, while line detection extends to several pixels that are ideally adjacent to other pixels of the line. Nevertheless this condition does not always apply in real images, caused by digital noise, occlusions or changes in illumination. **Various different continuity criteria must be verified in order to obtain a reliable edge detection in the image.** Then, detected edges have to be fit to straight lines. Fitting to straight segments can be accomplished applying linear regression for the points comprising an edge in the image.
2. A set of pictures of the same scene may feature different kinds of viewpoint changes among captures, including camera rotations, zooms and translations. These **changes in the camera viewpoint produce a morphological transformation of the primitives in the captured frame**, which translates into displacements of the detected primitives, changes on their shape, distortions, fragmentation or even the impossibility to detect the same primitive in another image by employing the same operations that served to detect it in one of the pictures. Some of these transformations are not applicable to points, for instance a fragmentation. A point is either fully present or not, but it

should not be such a thing as a detected fragmented point. Therefore, **there are more morphologic transformations that can affect 2D line segments than the ones that can affect points, due to camera viewpoint change**. Generally, prominent viewpoint transformations increase the difficulty in matching primitives, because the greater the transformation of the same primitives among different images of the scene, the greater the difficulty to match them.

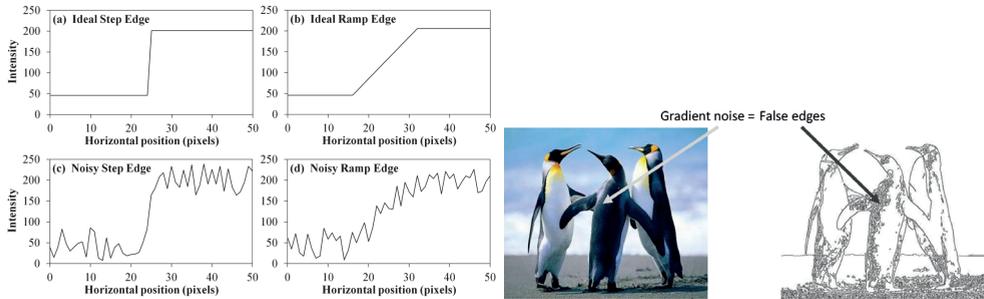
3. Matching primitives between images is not always accurate, specially when dealing with line segments. When finding counterparts for primitives detected in other images, it is common to come out with several mismatched primitives. These wrongly matched primitives are referred to as **matching outliers**. Matching outliers can produce that the description of the structure of groups of primitives will not be correctly compared to others, and employing inaccurate structure descriptions to propagate the matching to other images may cause problems. In the last point it was explained how **line segments are subject to more morphological transformations than points. The description of individual lines are therefore more subject to these transformations, and less truthfully at the end. This fact forces line matching methods to rely more on structure of neighborhoods than points**. The accuracy of the description of these neighborhoods compared with the real morphological transformation of the lines it comprises are highly dependent on the ratio of matching outliers.
4. In the frame of 3D reconstruction from relations between feature points, known the relative position of two cameras and the position of one point on the first image, **there is a constraint that forces the counterpart of this point on the second image to lay on a line. It is called epipolar constraint. But a single infinite 2D line represented in two images does not feature epipolar constraint. The only point-to-point valid correspondences in matched segments under a viewpoint change are their end-points**. For this case of a line segment, in order to be able to estimate its position in

3D, is required to detect in the images both endpoints of the line segment. In some cases it may be difficult to accurately detect the end of a line segment in an image. For instance, a segment can end by merging with another edge under a different slope, progressively dimming until it vanishes, by intermittent occlusions, or being abruptly fragmented. Moreover, one or both segment endpoints may lay in the limits of the frame, and in this case it will not be possible to extract the 3D pose of the line.

The above mentioned tasks portrait the main differences between lines and points raised out during the engineering of a complete line-based 3D sketch generation method from images. For each stage of the method described in this thesis, the author encountered specific tasks and problems that had to be solved in the frame of the current state-of-the art. These difficulties are highlighted in this section for the reader to understand the challenges faced by the author during the last years. Most of the work of the author was centered on the key tasks: detection of borders, matching lines over pairs of views, comparing the line matching performance against competition, relate the matched primitives among sets of more than two images, estimation of spatial lines, optimizing the abstraction and exploiting the resulting 3D structure.

### **Reliable edge detection on images**

The problem encountered on many of the published edge detection methods[47, 10, 126, 16, 112], as they are based on gradients, is that the image contrast determines whether an edge is detected or not. A graphic example of this challenge is shown in Fig. 1.1. The plots show the intensity values on the pixels close to an edge in the presence of noise. Digital captures use to show higher level of noise in dark regions, and regions with higher gradients. Nevertheless, finding these in a region does not imply the presence of edges, so the challenge here is to be able to discriminate edges in the presence of noise, high contrast or abrupt changes of illumination. Another difficulty raised with these approaches is that the location of the edge



**Figure 1.1:** Gradient-based edge detectors lead to false edges in regions with changes of illumination or noise.

may not be accurate due to the Gaussian smoothing. Some solutions taken for these problems, like special thresholds or using masks[128], revealed themselves too restrictive and missed important edges that would be observed by humans.

### Straight segments matching using groups of segments

Line matching is rarely reported in the literature. Methods that look for segment counterparts by using trifocal tensor[49] among sets of three images, or the epipolar constraint[121] for pairs of lines, require known relative geometry for the cameras that sourced the images. Another group of approaches employ descriptors for the appearance of segments, based either on their surrounding texture [159, 143], colors[9], supposing coplanarity with matched feature points to create projective invariants[81, 62], rooted on affine transformations[29] or constructing the invariants attending to the distance from segments to matched feature points[31]. For the methods of this second group, the difficulty yields on the requirement of a specific kind of scene for satisfactory matching results. In scenes that show man-made objects with high repeatability of segments featuring similar length and orientation it is complicated to distinguish the right correspondences. This can be worked out by embedding the groups of segments into polygons that take the lines as their sides[142].



**Figure 1.2:** Example of line matching using the method described in this thesis, shown to highlight the difficulties due to line occlusions on a viewpoint change. Segment 6, colored in magenta, is mismatched due to occlusion by the limits of the frame. Segment 17, colored in cyan, is mismatched due occlusion by viewpoint change. Images from the dataset [74].

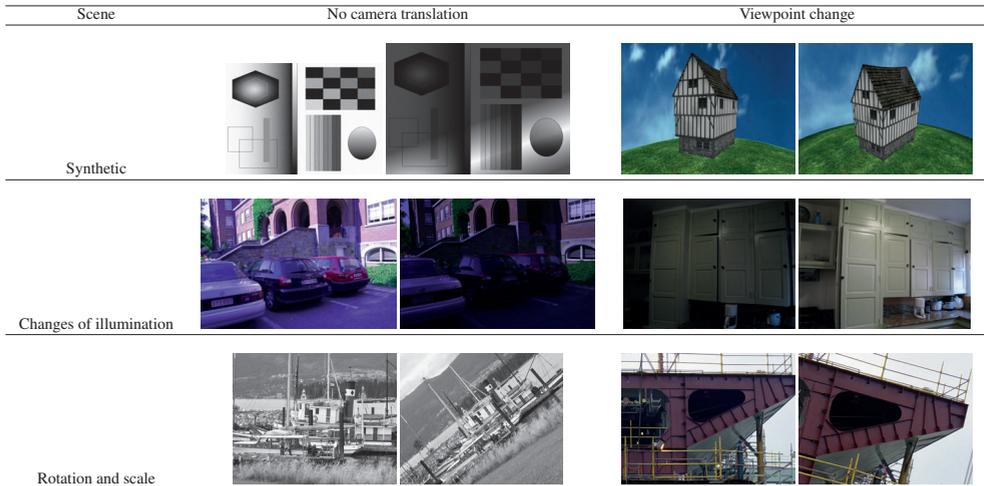
Line segments are usually matched independently over pairs of images attending to their appearance and the structure plotted by the group of lines of its surroundings[80]. On other methods, the appearance of each segment is embed into a line descriptor[159, 143]. For this latter group of approaches, the descriptor matrices extracted from one image are compared to the ones obtained from line detections in other images. Nevertheless, if the line structure is employed, it can be more reliable for matching between pairs of images, but in sets comprising three or more views, the structure of detected lines surrounding a segment might change for its counterpart in other images. This is illustrated in the line matching in Fig. 1.2: Firstly, partial occlusion produces that segment numbered 17 is not correctly matched, therefore the shape of this line neighbourhood is different for both images. The problem escalates when a new image of the scene is added, and the matching algorithm looks for a counterpart for segment 17. The second photo leaves the segment 6, on the left of the image, out of frame. Therefore this segment is incorrectly matched to the most similar detection in the neighborhood.

In the previous example it was shown how the usage of structure for line matching can be a two edged sword. It can be profitable for matching lines in a pair of pictures featuring a slight viewpoint change, while for wider viewpoint translations segments might see their neighborhood changed in a way that it no longer resembles the original one. A promising matching algorithm adapts to the scene according to average measures of morphology for the structure drawn by straight segments in the images. This ideal algorithm uses the average measures to dynamically update its preferred attributes employed to distinguish similarity between segments and their neighborhoods. An option to distinguish right matches in different kinds of scenes is to give different weight to the measures that show the least variability in the particular scene. This approach for measuring similarity between lines is computationally more expensive than using fixed routines, because the first one requires component analysis and subsequently repeating the operations to measure segment attributes after the weights are updated. Nevertheless, finding the right counterparts in this stage and minimizing the matching outliers will ease the following processes that involve matching these lines in groups of more than two segments, and estimating the 3D pose in the abstraction.

### **Measuring the performance of line matching methods**

Line matching methods must be experimentally proved against datasets that include synthetic images, real scenes featuring camera translations, rotations and changes of illumination conditions. Different kinds of scenes are classified in Fig. 1.3, with examples taken from public databases[29, 123, 74, 61].

The length of the matched segments or the similarity between counterparts are quantitative measures that can be automatically computed. Nevertheless, in order to quantify and compare the performance of a matching method, there are more characteristics that should be screened in order to highlight differences between methods. Moreover, not all the characteristics can be obtained with the computer. Indeed, a fraction of the score given to seg-



**Figure 1.3:** Classification of scenes employed for quantitative experiments. Taken from public datasets[29, 123, 74, 61]

ment matching in quantitative results unavoidably has to be obtained from human assessed Ground-Truth. This is the case of the inlier count for the matching. Simple decision rules can be predefined to be followed by a human assessment, like the maximum perpendicular distance in pixels from the line segment to the human Ground-Truth observation for being considered an inlier. This distance is often set to around 5 pixels[138]. In order to accurately assess the correctness of a line correspondence, the human can be instructed to set a minimal angle of deviation compared to the actual perceived line, which uses to be around 5 degrees[138]. In Fig. 1.4 a matching result with the method described in this thesis. The rotation and translation of the camera is evident between both pictures. There are literally hundreds of similar human-perceivable line segments in both images, and these are separated by few pixels. A human has to assess if the algorithm was able to correctly assess the same line with a predefined reasonable error in slope and normal distance to the human-perceived line.



**Figure 1.4:** Example of line matching using the method described in this thesis against pictures of the dataset [74]. The high density of short segments in the images raises the need of specific rules for discerning correct correspondences using the Ground Truth human assessment.

A good scoring system must comprise a blend of evaluations of similarity between counterparts, namely inlier ratio, fragmentation of segments, number of correspondences, difference in length between counterparts and processing times.

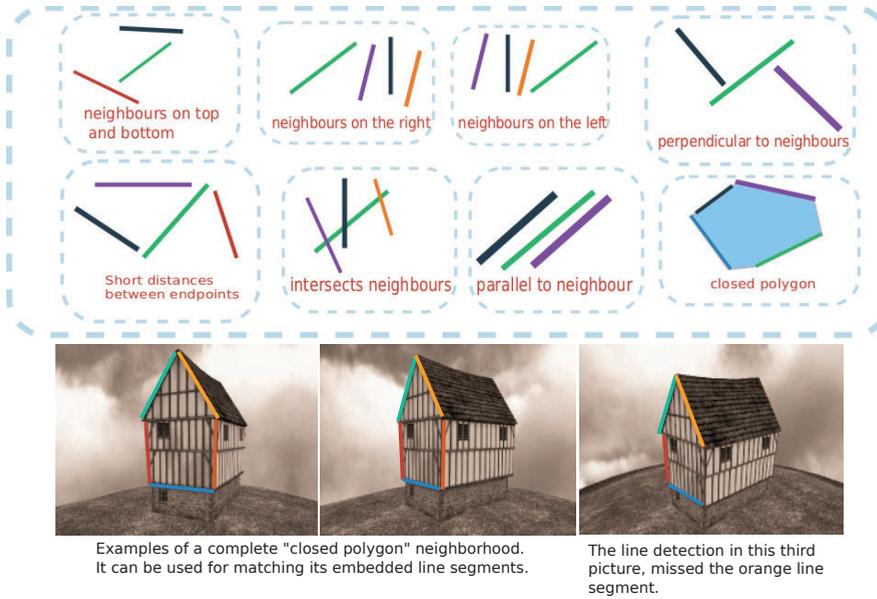
### **Merging counterparts from three or more views into unique multi-view entities**

Multi-view line matching can be performed by matching lines among all the possible pairs of images available, or alternatively employing a line descriptor for describing all the detected line segments and then look for coincidences throughout all the images.

Moving from pairwise matching to a set of entities referring to three or more views might carry complex relations. Matching two views implies to deal with a single global transformation of each kind, meaning that it will carry a single translation, a single rotation and a single zoom or scaling. Nevertheless, including more than two captures of the scene in the set can bring different transformations between the views. For instance, two views can be related by a camera rotation, while the next view keeps the same rotational angle with respect to the

second view, while translating the camera transversally to its optical axis. Therefore, the observed characteristics that are optimally chosen to relate the lines between these views differ. These characteristics that may be common to pairs of views in a dataset comprising more images, are related to the structure of line neighborhoods and the individual morphology of the lines:

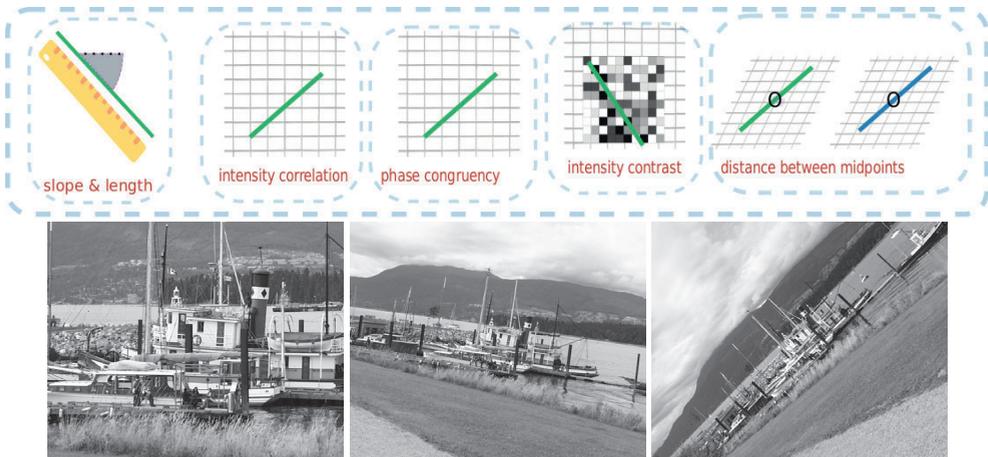
1. Structural characteristics of line neighborhoods that are valid to distinguish similarities between two views might no longer work well when relating the same structures to the ones drawn by the lines detected in a third view. The neighborhoods that are common to a set of views of a scene might be difficult to relate to a third view that incorporates new neighbors to the first ones, like occurred when a large camera zoom is revealing new details of the scene. These newcomers might not have a counterpart in other view because the detail of the picture did not reveal it. Therefore in this case adding a view to the scene produces that the relations between neighborhoods that are working for other views are not valid for the newly appended view. Fig. 1.5 shows different kinds of neighborhoods or groups of lines, and an example of how these may not be completely detected for all the views of the scene.
2. Morphological attributes of individual line segments are extracted when matching two views because these offered the least variability for the global population of detected line segments among both views. These are the attributes to exploit in order to distinguish potential counterparts. Nevertheless when matching among three or more views, there may be different global transformations involved. For instance, a case in which a global rotation of 20 degrees between detected lines is persistent between two views, while these views are related to a third view by a global reduction on the length of the detected segments. Therefore, attributes that are good for both views might not be the most relevant for a third view that feature a different kind of global transformation. Fig. 1.6 shows examples of individual attributes and a scenario with different global



**Figure 1.5:** On the top, examples of kinds of neighborhoods, defining the structure of groups of line segments. On the bottom a simplified visual representation showing how the orange line was not detected in the third image, and therefore the description of this neighborhood can just be used with the first two pictures

transformations. In this case the attributes of slope and length cannot be employed globally to define a line segment in the three images.

Line matching outliers in pairs of images might compromise the understanding of the 2D segments structure. Moreover, for a 3D reconstruction which takes three or more images, pairwise matchings have to be put together into a unique multi-view entity which can have a counterpart in each image. Unfortunately, this merging process might propagate errors of matching, aggravating the generated issues. A segment that was correctly matched in a pair of images might be wrongfully related to another segment in a third view. This line that was otherwise meant to be projected in space with the correct pose from the first pair of cameras, might be distorted into a wrong pose after adding the geometric information from the third



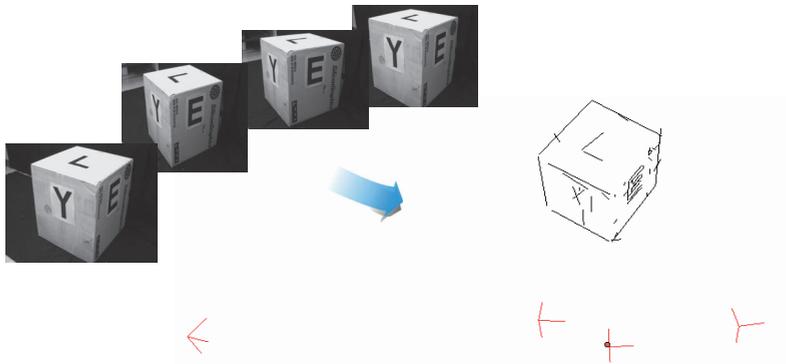
**Figure 1.6:** On the top, examples of individual appearance attributes that serve to distinguish sparse segments independently of their neighboring lines. On the bottom, a dataset featuring different zooms and rotations between images. It is an example for which **the slope and length of the segments are transformed differently** and hence have to be compared separately **for each possible pair of images of the set**.

camera. Matching outliers will produce inconsistencies, logic errors or incorrectly posed 3D segments. Ultimately, when the geometric outlier counterpart is put together with pose estimations from other views they can waste an otherwise correct portrait of a recognizable 3D structure, like a planar surface. This is why it is crucial to detect line matching outliers when all the segment matching information is put together.

### Estimate 3D lines

Each multi-view entity relates a perceivable line segment in the real world to its counterparts in images, as long as these have been correctly detected and matched. The process of generating a 3D representation from different pictures of the scene is visually represented in Fig. 1.7.

For the SfM problem, the poses of the cameras that took the pictures are not provided,



**Figure 1.7:** Representation of the challenge of converting a set of 4 pictures into a 3D sketch featuring the line segments and camera axis.

and it is up to SfM to simultaneously estimate the poses for the cameras and primitives. In the present case, SfM must estimate the pose of the lines in space relative to the cameras. The first requirement is the camera calibration matrix  $K$  for each camera, which provides the transformation between each point in one image, in homogeneous coordinates, to a ray in Euclidean three-dimensional space. Secondly, SfM has to estimate the projection matrices  $P$  for the cameras, representing a map from 3D to 2D:

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \quad (1.1)$$

where  $x$  is a 2D point on the image, and  $X$  its projection in 3D space.  $K$  is intrinsic to each camera, while  $P$  is extrinsic and embeds the 3D translation and rotation of the camera's image plane. The estimated translation is valid up to scale.

A common space can be built to host the cameras and spatial lines. For this new common space the camera that took the first processed picture takes the place of the origin, and for the rest of cameras  $P$  can be estimated from the lines matched between the captured images. Alternatively, camera poses can also be retrieved from a feature-point based SfM pipeline and these cameras be employed for the estimation of spatial lines. For instance, the feature-point descriptor SIFT[82] can be used to match points in images with a low ratio of outliers. These

feature point relations are obtained both in the foreground and background. Having a set of relations between points or lines in two images allows to estimate the homography constraints between both views by applying the 5-point algorithm[102] using the points or the segment endpoints. A purge of outliers can be performed employing RANSAC[38] for robust estimation. Therefore, a set of stereo 3D projections is obtained combining the available images pairwise, and each stereo system featuring both camera poses and a point cloud. The objective is to obtain an unique 3D point cloud sketch, embedding all cameras and point matches. Hence, camera poses are sequentially stacked, and the 3D estimations for the feature points in the new 3D space can be computed as the center of gravity for their position relative to the common camera in both stereo systems. Finally a sparse bundle adjustment[137] is used to minimize the pixel distance of the back-projected 3D point and the original observation of this point on each image in homogeneous coordinates. These reprojection errors on the planes of the cameras are minimized employing the Levenberg-Marquardt algorithm. The resulting keypoint-based 3D reconstruction contains the optimized 3D estimations for the cameras and the point cloud.

Several straight segment matching methods are based on texture descriptors[159, 143], coloring[9] or in keypoint-line projective invariants[81, 62]. It was also pointed out that under these conditions, matching results will be influenced by the level of texture in the images. In the case that a low number of detected segments can be distinguished by employing image texture based descriptor, or in the case that a low number of feature points are identified throughout the set of images, the resulting set of matched lines will not be satisfactory. On the other hand, if line matching is rooted on weak epipolar constraints[56], line matching will be highly dependent on the accuracy of the camera poses.

The extrinsic parameters for cameras are needed to project the matched lines into space. Having the same segment completely detected and without fragmentation for both views under viewpoint change, endpoints are the only points in a segment with known exact counterpart in the other image. Unfortunately, segment detection is not accurate in the location of

the endpoints. Therefore, the most accurate abstractions will be the ones built rooted on camera extrinsics obtained from a dense feature point based SfM. As written above, known the projection matrices  $P$  of two cameras, a point on an image projects as a 3D ray in Euclidean space. And this 3D ray projects like an infinite 2D line on any plane different than the one that contains the point. Therefore, each 3D point  $X_p$  will have its image into an epipolar line  $e_p$  contained in the image. As the unknown point is constrained into a line in the other image plane, analogously a segment will be constrained between both epipolar lines corresponding to the segment endpoints. This weak epipolar constraint can be employed for matching segments between images[56].

### **Bundle adjustment for line segments**

In the case of feature points, the final position of the projected features relative to the camera poses is estimated throughout an optimization process. As a part of most SfM pipelines, bundle adjustment[137] is based on Levenberg-Marquardt, and it rearranges the poses of the cameras and 3D points. The cost function of this optimization process is engineered to find the minimum distance error between the reprojection of every 3D point onto each camera plane and their original observation. A limit value for the residual is usually set to stop the iterative process for the event of convergence, while another threshold is set to end the optimization when reaching a maximum number of iterations.

Along matched segments under a viewpoint change, the only point-to-point valid correspondences are their endpoints. Segment's endpoint location are noticeably less accurate than a rotation and scale invariant feature point. Employing line endpoints as the sole set of geometrical constraints in the adjustment might not be adequate to improve the 3D sketch. Recurrent segment mismatches, fragmentation or the inaccurate placement of counterparts may prevent the convergence of the optimization.

It is possible to perform a line-based Bundle Adjustment by converting the primitives into

Plücker coordinates [109, 7] within the cost function of the optimisation process. This allows a reduction in the number of parameters.

### 1.3 Objectives of the work behind this thesis

The objectives of the work is to develop a method that inputs images of a scene and is able to build a straight line based 3D abstraction comprised by the estimations of the poses in space of the segments and the cameras. Three more specific checkpoints have been set to reach the goal:

1. Implement a robust straight line detection method for images based on the state of the art edge detection. This segment detection method will be to some extent resilient to the presence of low texture, different illumination conditions, blurring and noise in the images. It will also avoid fragmentation line segments as far as it goes.
2. Engineer a line matching method for pairs of images that employs the individual segment appearance and the structural description of groups of segments. This will input the detected segments in both images and look for the counterparts of each one, minimizing redundancy.
3. Design a method for generation of straight line based 3D abstractions of a scene from sets of three or more images. The system will aim for difficult datasets with low number of images of the scene or object, captures showing low texture, blurring or noise.

### 1.4 Contributions of this thesis

The contributions of the method presented in this thesis are related to the following tasks in the current state-of-the-art:

1. A detection method for straight segments in images, engineered to overcome the common problems found in the methods based on gradient, complications described in Subsection 1.2.
2. New ways to describe groups of straight line segments based on their individual appearance characteristics and the ones of groups of segments.
3. A novel method for finding counterparts for the detected straight lines among different images of a common scene. The matching method is engineered to solve the problems depicted in Subsection 1.2, and commonly found in line matching methods.
4. A novel method to minimize line matching errors in scenes with flat surfaces is presented. It is based on grouping segments according to 3D structures.
5. A method that build 3D sketches or abstractions based on the lines matched through different images. The method obtains spatial representations of the objects, scenes or synthetic models captured in the photos. The 3D sketch features a spatial representation of the cameras and lines. Moreover, the 3D structures in the obtained sketch is exploited to extract geometrical information. The difficulties encountered in the current state-of-the-art are explained in Subsections 1.2 and 1.2.

In order to perform the edge detection, each image is downsampled in order to build a Gaussian scale-space. Edges are extracted in the original image and in the downsampled octaves. The resulting edge skeletons are fitted to straight lines by a robust linear regression, followed by a threshold for the angle of curvature. Final line detections in the original image are obtained by merging the straight segments detected in all the scaled images.

In this thesis we used both segment appearance and polygonization for describing the structure of groups of segments to perform the line matching. We believed this approach will perform better for scenes with low texture, poor visibility or blurring, where a dense point cloud is not possible to obtain.

The measurement of similarity between straight line segments includes both structure and appearance. A voting mechanism is employed within every neighborhood in order to relate one line to each potential counterpart on the other image. The number of votes and the similarities of their neighborhood will determine the final score, and hence the definitive counterpart for each segment that reached this stage.

Camera poses are known at this point, either estimated from a feature point based SfM pipeline, or from the set of line endpoints. For each pair of views, line segments that were matched between both images, are forward projected from the cameras into space. Therefore, multiple stereo subsystems are built, and each one solely features the line segments they have in common. These stereo systems are represented on the upper region of Fig. 6.3. Next, the unique space is created. The first camera is located at the origin of the coordinate system. From the geometrical information of all the subsystems, the rest of cameras are sequentially stacked respect to the position of the first camera.

Human-made environments, indoor spaces and objects, often show geometric similarities in their surfaces. These surfaces are commonly either contained in common planes or are orthogonal to their neighboring surfaces. For instance, such similarities are observed in windows of a building. Fitting lines to different planes allows to group features and classify them. As a consequence of this geometrical grouping, different spatial lines and points are related by simple 3D geometries.

The 3D sketch is exploited in order to find potential line matching outliers. As represented, 3D segments are fitted into planes by employing RANSAC as hypotheses generator within the whole set of lines. After the plane with more inliers is fitted, the lines related to this newly unveiled plane are discarded for the next iterations of the algorithm. Finally, several planes may be fitted and their related lines grouped accordingly. The lines that are not referring to any fitted plane are not grouped. The grouped coplanar segments are intersected in every original image. The line intersections are grouped according to the matched lines in the group. Weak epipolar constraints are used to distinguish potential matching outliers, as the

lines whose intersections with coplanar neighbors are not holding against these constraints.

## 1.5 Organization of the thesis

A short chapter with the main concepts and definitions used in this thesis is followed in Chapter 2. A bibliographic survey including related works is included in Chapter 3, and it goes through the most salient publications related to the topics covered in this thesis. This comprises works about detecting edges in different images, straight segment detection and their applications, line matching between different images, and multi-view 3D abstraction from images. Chapter 4 is devoted to straight segment detection in images, and also covers edge detection in images. Chapter 5 is about straight line segment matching between different images. Chapter 6 goes through 3D abstraction from images based on lines. Finally, Chapter 5 exposes the conclusions of the present work, and the future prospects for this branch of research.

## Chapter 2

# Background

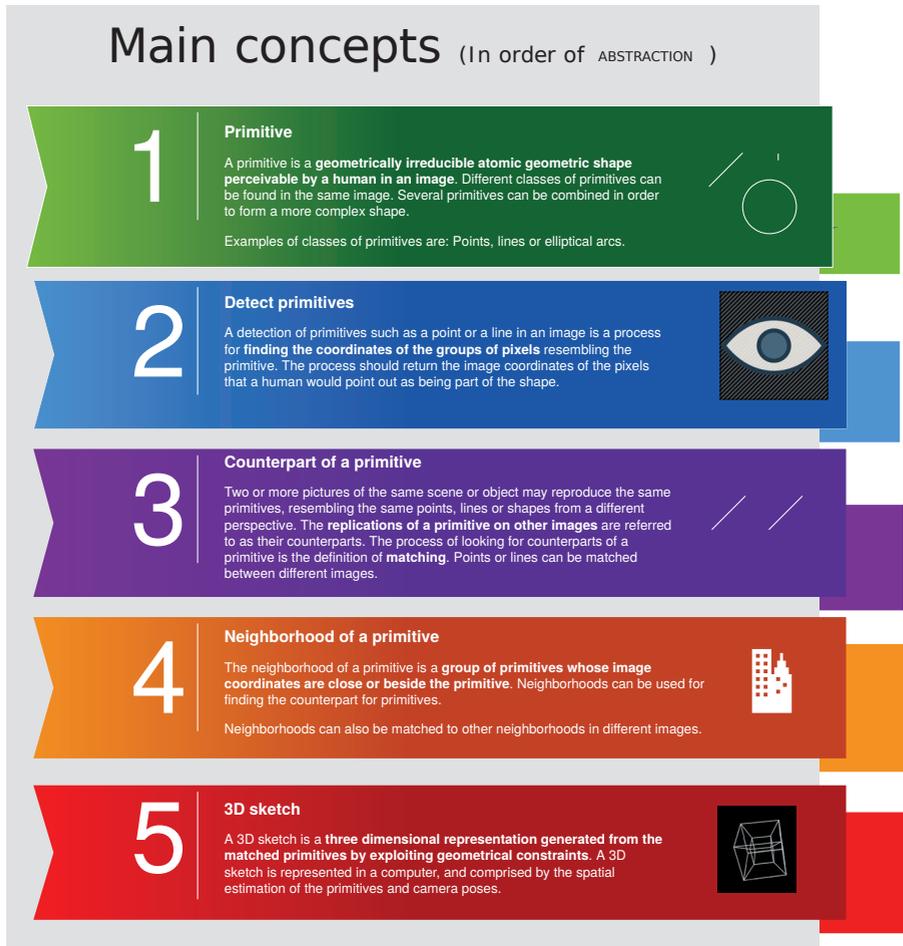
Images are processed for fetching usable pieces of information. In order to understand the represented scene or elements shown in the image, it is necessary to reduce the image to **pieces of information resembling geometric shapes that are perceivable by humans**. These pieces of information are referred to as **primitives**, and this is the first concept to understand for the reader of this thesis, as shown in Fig. 2.1. The pixel coordinates comprising the primitives are fetched during the primitive detection process. The detection of primitives in an image is an initial procedure towards the understanding of a scene.

Two or more images may reproduce the same primitives under different conditions. Relating the coordinates of these primitives is a process named **matching**. Two primitives in different images can be matched attending to characteristics in the images, such as their appearance in the image, the appearance of other detected primitives, or geometric constraints. After a successful matching, **a primitive is considered counterpart of other one in a different image if, based on these characteristics, the matching algorithm has found evidences that relate each other**. The definition of counterpart is highlighted in Fig. 2.1.

A group of detected primitives of a predefined kind, with predefined characteristics, lo-

cated in the same image and at a predefined distance from a primitive is defined as the **neighborhood** of the latter. The neighborhood of a primitive can be used as a source of evidences to relate the primitive to a different primitive in other image. This concept is included in the forth point in Fig. 2.1.

Primitives can be matched between different images, so it is possible to create an instance of a real spatial shape that links to every counterpart on their images. This spatial shape *projects* onto each picture as the detected primitive. It is possible to estimate the location of this spatial shape relative to the cameras that took the pictures, by using geometrical constraints. This estimation has three dimensions, includes representations for camera poses and shapes, and is referred to as **3D sketch**. The 3D sketch is the last main concept pointed out in Fig. 2.1.



**Figure 2.1:** Definitions for the main concepts included in this thesis, in order of complexity.

## 2.1 Detection of primitives

The detection of primitives on an image is the process aimed for **separation of the groups of pixels resembling the underlying human-perceivable primitive**. It generally starts by applying different convolutions on the original image in order to detect the target primitives. The result of these operations highlight groups of pixels resembling the primitives attending to predefined **criteria over appearance characteristics**, such as pixel color, intensity, contrast or phase congruency [71, 36]. Every preference or restriction on the characteristics of the shape of interest is implemented in the selection criteria attending to predefined descriptors. These descriptors are engineered as matrices that feature metrics into a predefined scheme, and these metrics are measurements over the appearance characteristics. Finally, the result of an operation(convolution) between the descriptor and the region of the image is used to decide whether the region includes a shape matching a predefined pattern that resembles the primitive. This detection process ends with the creation of a 2D map of pixels including the detected primitives. Detected primitives are then characterised attending to their own appearance attributes and the ones of groups of other detected primitives in the same image. The resulting description and classification is necessary in order to distinguish two primitives that may be related by a common measure or condition. This is the first step towards finding counterparts of a detected primitive in other images or regions, process which is referred to as matching.

## 2.2 Matching of primitives

A primitive matching method comprises a set of algorithms to find counterparts for segments across different images showing common elements, environment or regions of interest. The processes that interrelate the detected primitives are applicable if these can be found on more than one image, or when one image features any kind of redundancy for a pattern, so differ-

ent instances of the same region of interest can be observed in the unique frame. Therefore, a matching process can be performed either on a single image, independently over pairs of images, or by reducing the detected primitive to a descriptor and using it for locating correspondences on other images. The detected primitives are described and characterised attending to their apparent attributes, the ones of their neighboring primitives in the captured scene, and their geometrical distribution. This is the first process in which relations are created between primitives. Prior to the search for relating primitives and redundancies, an analysis of the degrees of freedom of the system has to be done. Therefore, this analysis of the whole system takes into account characteristics of the scene whose imply a constraint or restriction in the degrees of freedom. In the frame of line matching, this is the case of a predefined preference for the alignment of features named in Computer Vision as Manhattan World Assumption[17, 63] and aimed for urban scenes. Preferred directions are set for straight lines, in such a way that they correspond to the projection on the camera plane of parallel or orthogonal directions. These restrictions are exploited both for primitive matching and for subsequent stages of the spatial reconstruction process. No analogy can be done with points under the Manhattan World Assumption.

The proposed approach is framed in the group of line matching methods aimed for 3D reconstruction from pictures of objects built by humans, buildings, urban structures, industrial elements or computer generated models.

### **2.3 3D reconstruction**

A 3D reconstruction is the result of an estimation for the position of singular primitives matched among several images. In order to generate a proper 3D representation of the environment, primitives detected in the input images must be put in correspondence among the rest of images. Secondly, the whole set of matched primitives is employed to estimate the most probable poses for the cameras that took the pictures, by exploiting geometric con-

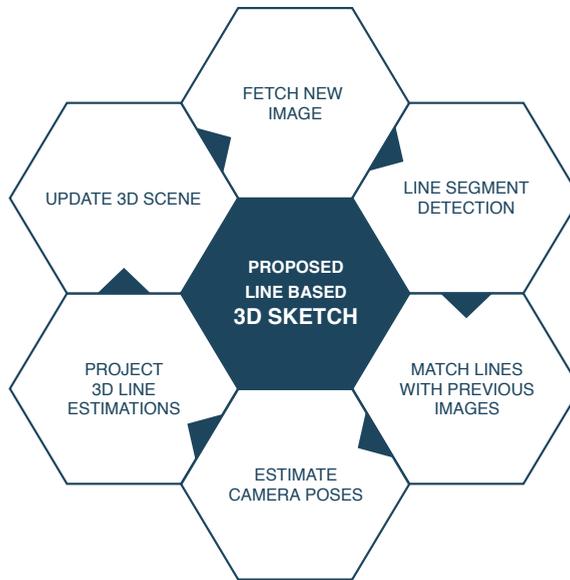
straints.

Cameras can be geometrically parameterized, and these parameters condition the final spatial abstraction. The group of characteristics of a camera defining its position and orientation in space are referred to as **extrinsic parameters**. The measures related to the construction of the optical geometry of the camera that will affect the final captures, such as the focal length, are embed in the **intrinsic parameters**.

Structure-From-Motion (SfM) is the first step towards the generation of a 3D reconstruction of a scene or object. **It estimates the unknown relative poses for the camera at the instant when each photo was taken, altogether with the location of primitives relative to these poses. The set of matched primitives is required as the only piece of geometrical information that relate the different views of the scene.** The geometric problem of estimating the unknown camera extrinsics is solved differently depending on the configuration for the cameras taking the pictures. If the setup comprises a set of cameras, each one with known relative translation and orientation respect to the other cameras, the number of degrees of freedom for the geometrical problem can be reduced due to the physical constraint between the views. This is the case of a stereo system comprised of two parallel cameras attached to the same rigid support. On the other side, the estimation of camera extrinsic parameters will be more complex in the case of freely moving cameras. In this case, the extrinsic parameters have to be estimated from cameras must be incrementally added to the spatial abstraction. An example of this using points as primitives is shown in Fig. 2.3.

There are available SfM bundles that reproduce objects and scenes from photos, some of these are free[147, 129, 1] and some others are commercial, like Agisoft Photoscan or Autodesk ReCap. These software applications are usually computationally expensive. Nevertheless SfM is getting very popular and is usually teamed with new desktop 3D printers and high definition cameras. These allow to build a reconstruction of the shape of an object in the real world by using just a smartphone with camera.

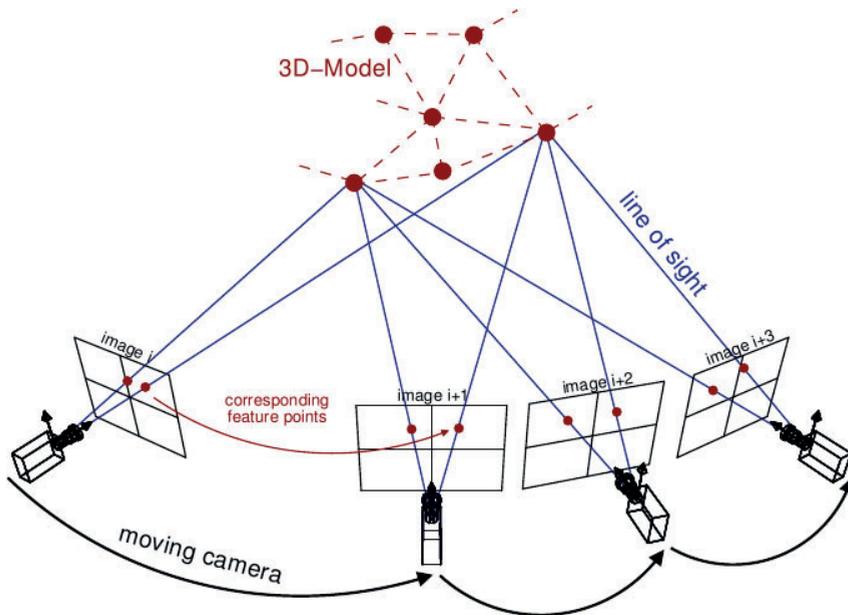
3D abstraction is included in this thesis about lines in order to **exploit the line matching**



**Figure 2.2:** Real-time generation of 3D abstraction

**results for the estimation of the location of a camera and their surrounding structures.**

It can help as assistance for the guidance of autonomous robots or unmanned vehicles. The generation of a line-based 3D sketch can be performed in real-time, using the newly captured images to update the reconstruction, as represented in Fig. 2.2.



**Figure 2.3:** Incremental addition of cameras to the 3D abstraction [133]

## Chapter 3

### Related works

In recent years, low-level image features have been proven useful for achieving reliable correspondences among images. Several feature point based methods have been proposed such as Moment Invariants [95], Steerable Filters [40], Differential Invariants [96] or multi-scale feature detection and description algorithms. The most popular multi-scale based approaches are the Scale Invariant Feature Transform (SIFT [82]), the Speeded Up Robust Features (SURF [8]) and KAZE [3].

SIFT [82] and SURF [8] features changed the way machine can **extract relations** between images and regions of images. The first one obtains candidate keypoints as the maxima and minima of the result of a difference of Gaussians through a scale-space. Keypoints are described as invariant to both scale and rotation. It was shown that SIFT-based descriptors, that match keypoints by using a scale invariant region detector and a descriptor based on the gradient distribution, outperform the other mentioned approaches [93]. However, the drawback of these SIFT-based methods is that by relying on properties like texture or local structure, they become deficient with low-textured objects and homogeneous surfaces, which are typical in industrial environments. In these cases, it is essential to use line segments as matching

features due to their abundance on man-made objects, and also because they bring us greater structural information. SURF [8] exploits the fact that Gaussian derivatives at different scales can be approximated by simple box filters, and a sum of Haar wavelet responses is computed per region. This allowed SURF to obtain better repeatability, distinctiveness, robustness and performance than SIFT. More recently, KAZE [3] substituted Gaussian scale space by non linear diffusion filtering in order to obtain much higher repeatability and distinctiveness than the previous algorithms of this kind.

The above mentioned multi-scale feature point detection and description algorithms create relations between points in images, and one of the main applications is to build **3D abstractions** of the captured scenes. The vast majority of the current approaches for 3D scene reconstruction are based on point clouds[151]. Hartley[50] solved the problem of triangulation of matched points among different views, by assuming Gaussian noise model for perturbation of the image coordinates, while knowing the camera intrinsic parameters. The same work proposes different ways to estimate camera poses from a set of triangulated points. Few years later, Triggs et al.[137] overviewed the Sparse Bundle Adjustment (SBA) optimization to improve clouds , as a least-squares minimization. Commonly, points are matched between pairs of views based on their descriptors, then triangulated to make an initial estimation of their relative rotation and translation in 3D space, and finally their poses are adjusted by using least squares minimization. A number of efficient point detectors and descriptors have made possible to generate robust and detailed 3D reconstructions based on feature point clouds [83] [8] [114] [3]. These algorithms made possible to evolve from simple 3D reconstructions of the surfaces[108] to dense point reconstructions of extensive landscapes and cities[129].

The photogrammetric technique SfM has been used for mapping forested landscape features from Unmanned Aerial Vehicles. This technique has similar operational requirements than airborne laser scanning (ALS) and is capable of providing a representation with equivalent level of accuracy[141]. In this study the main difference found was related with the lack penetration of SfM through the upper canopy of the dense vegetation compared with the

laser. The SfM technique applied to images acquired by a low-altitude UAV system has also been proved comparable to GNSS survey data, with a low RMS error for the SfM result, and just with some difficulties over small areas with sudden changes on topographic slopes [86].

### 3.1 Detection of lines in images

Straight line segments have been included in Computer Vision since back in early 1980s. The applications at the time were line matching between pairs of views[89, 90] or tracking the line segments through images[20]. The first works about Structure-from-Motion (SfM) based on lines were focused on the mathematical bases[49] and showed experimental results from low resolution and simple scenes[134]. The first works published in this century added new applications for the detection of line segments, like crack detection in materials by analyzing microscopic images[85]. Other application introduced during the change of century was improving the compression of images[39]. With the popularization of the Internet, some works employed lines for image indexation[46]. Matching lines between images is used to find similarities and common geometries between views[80]. Popular applications for line matching now are line based 3D abstractions[54]. A more specific application employing 3D abstraction is the guidance of robots, referred to in the literature as odometry[151].

#### Edge detection

Kovesi[71] defines an edge as a location in the image where the local autocorrelation function has a distinct peak. An operator that detects edges is usually based on convolutions to local derivative filters on the original image.

The detection of edges is the process that returns a map of pixels marking where these structures can be perceived by a human in the image. The detection of straight lines in images is following the detection of edges, and it can not be made without the latter. The author wants

to differentiate methods for edge detection in images and the ones of straight line segment detections.

The first works about edge detection in images dates of the 1970s, and were focused on the theory of color changes[111], intensity changes, scales, image filtering and derivatives[87]. The mentioned filters were used to discriminate orientations of edges and intensity changes.

Harris edge detector[47] is based in the local auto-correlation function. Canny edge detector[15] classifies a pixel as an edge if the gradient magnitude of the pixel is larger than those of pixels at both its sides in the direction of maximum intensity change. The Gray-level edge detection[65] computes the curvature of edges altogether with the gradient of pixel value of grey-scale images. Cooper et al.[16] published two edge detectors based on image gradient magnitude. This magnitude is computed along different paths centered on a point which can be an edge.

Edge detectors based on local energy are also proposed: The method from Rosenthaler et al.[112] uses oriented filters with even and odd symmetry, and combines their convolution outputs to oriented energy, in order to detect edges and keypoints.

At the end of last century, biometric applications promoted the usage of edges extensively, for example for the analysis of human palmprints[155].

The origin of some of the problems to solve in the fields of application of straight line segments is related to the way the segments are detected. Firstly, the fragmentation of lines occurs when an uniquely perceived segment entity is detected as an array of shorter segments. These sections have to be merged into an unique entity before searching for segment matching hypothesis. A theoretical approach to obtain a more reliable detection is to deal an image as a 2D generalization of the analytic signal, so it is possible to perform a transformation to account for the local amplitude and local phase. In this case, edges are detected in the scale-space, comprised by the original image and several downsampled versions of it. Adding this third dimension allows to employ the monogenic signal[130] to detect edges, because the maxima of differential phase congruency resemble the main edges on every picture in the

scale-space. Another advantage of going into this frequency band for border detection is a higher resilience to changes in contrast and intensity in the image. Detecting line segments in different scales allows to access the frequency domain[71]. With this approach, edges are not just marked according to a greyscale map, but the locations where the image signal gets maximal phase congruency[130].

The revolution in the way we exploit data started in the present century, and it opened the gate to new kinds of techniques. It is possible to classify the methods attending to the source of the data employed for the edge detection. During the writing of this literature review of the state-of-the-art of edge detection the author found out that works devoted to edge detection often get close to the topic of segmentation. Despite learning-based methods for object segmentation is not the topic of this thesis, many of these works are rooted on their own method for edge detection. In this way, the author found no reason to exclude the works aimed for segmentation for this section. Therefore, it is possible to organize the existent works about edge detection into three different groups:

1. **Model based methods:** In this group we can fit the above mentioned early methods, characterized for being based on predefined thresholds, patterns and responses to operations. One example is the Sobel detector[66], which keeps being improved[45]. Other early methods are based on zero-crossing[87]. In this group it has to be mentioned the Canny detector[15] which is still very popular nowadays.
2. **Data driven methods:** These appeared with the change of century, and the detection is based on probability distributions with examples like Statistical Edges[68]. The method Pb[88] was aimed for natural scenes and was followed by gPb[4]. The former one was improved by its author, and named as Multiscale-Combinatorial-Grouping (MCG)[5]. It is based on bottom-up segmentation and object candidate generation. In addition to edge extraction, it segments hierarchical regions and object candidates. Pointwise-Mutual-Information (PMI)[59] is a method intended for unsupervised object

segmentation, but includes its own method to find pixel-level accurate boundaries that avoids to mark as edge the texture of the segmented objects.

3. **Learning-based methods:** For the methods in this group, learning is based on inputs provided by humans. A relevant method in this group is Structured Edges[23]. Other examples are BEL[22] and Sketch tokens[77].

In addition, there are methods based on Convolutional Neural Networks. The difference compared with to the latter group in the list is that in this case the learning is automatized. Examples are HED[149], N<sup>4</sup>[42], DeepContour[124] and DeepEdge[11].

Despite the new wave of works employing learning appeared during the last years, several works of the first group also appeared, and these worth mentioning. For instance, in 2012 the simple Edge Drawing (ED)[135] method ED was published. Firstly, the image is filtered with Gauss. Secondly a gradient magnitude and orientation is computed. The peaks of the gradient maps are the pixels with high probability of being part of the edge are marked. The resultant continuous edge skeleton is drawn by joining the marked pixels. In the same group and published in 2017, FENI[103] is aimed for edge detection in noisy images, and takes the challenge of estimating on how faint can an edge be and still be detected, based on the minimal detectable contrast in an image. Moreover it proposes separate methods for straight line and curvy edge detections. Another example from 2017 is PLineD[118], aimed for detection of power lines for Unmanned Aerial Vehicles, it follows a similar approach as ED[135] for obtaining the edges for then chopping them. In order to highlight the power lines in the images, it chops the edges according to its orientation, purge them according to their length, and finally groups the resulting candidates.

### **Detect straight segments**

The fragmentation of lines occurs when an uniquely perceived segment entity is detected as an array of shorter segments. These sections have to be merged into an unique entity

before searching for segment matching hypothesis. One approach is to fetch segments based on edges detected in the scale-space and employ the monogenic signal ([130]), because the maxima of differential phase congruency resemble the main edges on every picture in the scale-space. Another advantage of going into this frequency band for border detection is a higher resilience to changes in contrast and intensity.

Most classic methods employ Hough transform over the edges extracted by the Canny edge detector[15]. The extracted straight lines with this transform are the ones containing a number of edge points higher than a threshold. Then lines are chopped according to length and separation thresholds. One of the problems with the Hough transform is caused by the impossibility of adjusting these thresholds dynamically according to the level of texture. Therefore, when trying detection on textured regions, it leads to false detections. No matter how accurately thresholds are designed for a picture, there will be regions exceeding its optimal working range. This problem related with the global thresholds is also negatively affecting the results of other propositions of edge detection, like the work of Deriche et al. [20], which merges edge pixels into curves, then joins them together into lines, according to predefined thresholds. The process of joining is referred as chaining. The chaining method from Etemadi[28] is able to segment edges into circular arcs and straight lines. It avoids user-supplied thresholds, and is based on complex edge maps.

At the end of the last century, the work Meaningful Alignments (MA)[21] focused on extracting the significant geometric structures in a single image from the detected line segments. It established the properties that allow to characterize meaningful segments. For this task the authors assume that the gradient direction of all pixels in the image is an statistically uniformly distributed random variable. Secondly they set the conditions that pixels with aligned direction have to meet in order to resemble a meaningful segment.

The popular Line Segment Detector (LSD)[140] appeared in 2010 and obtains the straight segments by representing edges as regions of aligned orientation pixels. It was followed the next year by EDLines[2], based on edge detected with ED. It includes false detection control

due to the Helmholtz principle. An improvement over LSD aimed for robust line-based SfM was presented by Salaün et al.[115], and has taken profit from edge detections over multiple scales.

In 2015 Xu et al.[150] published a method for line segment extraction using minimum entropy with the Hough transform. In the same year, Zhang and Huang[156] came out with a simple method for straight line detection by using simple conditions and thresholds. In 2016 Budak et al.[14] improved LSD by teaming it with SIFT and the Fisher Vector[117] for airport detection from aerial images.

## 3.2 Line matching

Few methods for automatic line matching are reported in the literature, due to several inherent difficulties. These difficulties include the inaccurate locations of line endpoints, object occlusions leading to missing line counterparts, the fragmentation of lines often causing the loss of topological connections among line segments, and also the lack of a global geometric constraint such as the epipolar constraint in point matching. Existing approaches for line matching can be divided into three types: those that match individual line segments, those that match groups of line segments, and those that use some point correspondences or epipolar constraints of line endpoints, to perform line matching. The main drawbacks of the last kind of approaches, such as the ones of Schmid and Zisserman[119] and Fan et al.[30], are the requirement of knowing the epipolar geometry in advance, and the reliance on some point correspondences which makes them inappropriate for low-textured scenes, typical in industrial environments.

Most of the approaches to match individual line segments are based on appearance similarities of the line segments, such as Bay et al.[9] where color histograms are used to obtain an initial set of candidate matches which grows iteratively, or Wang et al.[144] which defines the MSLD line descriptor by using a SIFT-like strategy. However, the sole reliance on the

appearance of lines also makes these methods inappropriate for industrial scenes, with textureless surfaces. In this way, Zhang et al.[158] have presented a line matching algorithm which considers not only the local appearance of segments but also the direction of lines. Direction histograms are used to estimate whether there is an approximate global rotation angle among image pairs, and if so, this angle is used to reduce the candidate matchings. Although it has been shown that this method achieves better results, the problem remains when there is no accepted rotation angle among images.

The main advantage of the methods that match groups of line segments is that more geometric information is available for disambiguation. For instance, in the method proposed by Wang et al. and referred to as LS[142], line segments are grouped based on the spatial proximity and relative saliency, and then these groups are matched by using angles and length ratios to describe geometric configuration of their segments, and also average gradient magnitudes to describe appearance information. This strategy is shown to be useful to deal with large viewpoint changes, and non-planar scenes. However, in order to improve the repeatability of line signatures among images, a multi-scale scheme for line extraction is applied to divide all the curves in many consistent ways, so that each curved connected-edge is polygonized at various scales and all possible segments are kept. This overestimation in the number of line segments makes this method computationally expensive.

A classification of the related methods for line matching can be done in two groups: one implementing hypothesis generation for the correspondences using exclusively data from 2D observations while avoiding to use homography constraints, and a second group that roots the matching candidate selection criteria on homography constraints.

One option for obtaining hypothetical counterparts based on 2D is to use a descriptor, like the above mentioned LS[142]. It parameterizes different similarity measures for scale and rotation invariance, altogether with affine invariance attributes. MSLD[143] encodes a SIFT-like description of the different regions of a line into a description matrix. LBD[159] improved MSLD by adding geometric constraints and an outliers topological fil-

ter. SMSLD[138] added scale invariance to MSLD, by creating a scale-space from the images, and matching the regions close to each segment. Nevertheless it is required an ample region beside the lines for the algorithm to success. Other methods include the definition of junctions as intersections of line segments subject to some geometrical conditions, in order to broaden the measurable characteristics of groups of segments. It is the case of LJL [75], a method that categorizes detected lines touching each other in one of their endpoints, and compares intensity changes along the segments, the angle that the pair of segments is drawing, and takes in account other neighboring line junctions. Finally, sole lines are grouped and become matching candidates to the segments detected close to the junction counterpart on other images. It is also possible to match lines over pairs of images by employing Harris edge detection[47] and matching the areas around these edges for instance using normalized cross-correlation (NCC) of the gray levels of contained pixels. Line matching candidates are found according to proximity measures to these matched areas [104]. Zeng et al.[154] uses appearance and structure for hypothesis generation, but adding feature point descriptor SIFT for outliers detection and removal. Another approach is by drawing convex hull around the clusters of segments, and exploiting affine invariants in the hull[84]. The ratios of the areas of triangles drawn inside the hull are compared. This method has been improved for the proposed work. For this group of approaches, additional efforts are required for outliers filtering, in order to prevent a critical degradation of the final 3D reconstruction. These issues come up more relevantly if the experiments are conducted with datasets featuring a wide camera viewpoint change.

The second group of methods is employing homography constraints in order to obtain the relations among detected segments. The method LJL was evolved to VJ ([76]) by adding homography constraints from the intersections of the elongations of closely located pairs of line segments. It is possible to separate groups of hypothetical matches just by spatial proximity to other hypothetical 3D representations, meaning that it is more likely that a match is true if it is close to other 3D line hypothesis[53].

LPI[31] exploits the line-points affine invariants analogously to the methods of the first group, and takes advantage of the projective homography invariants by using four feature points beside the line. A recent evolution of this method is CLPI[62], that constructs the line-points projective invariant on the intersections of coplanar lines. The drawback is that for both exploitations they had to suppose that all points and the line are coplanar, and often the lines resemble the limits of two planar surfaces. The downside of these methods is that they require a previous result from a feature point based SfM pipeline.

The recent work from Shen and Dai[125] shows a line matching based on matching points along the segments. It is required to provide the homography for the views. Instead of matching lines directly, for each segment it gets sample points as representation to match the whole line, by using optical flow along the line.

### **3.3 3D abstraction based on straight segments**

Two different problems can be considered regarding 3D abstraction from primitive matching: The first kind of problems aims to obtain the 3D abstraction from known camera poses, being these sourced from ground-truth data. In the second type of problems, referred to as SfM, camera extrinsics are not provided, so both the camera poses and the location of scene primitives have to be estimated altogether. This differentiation is decisive for the quality of the resulting reconstruction. For instance, the case of a moving UAV stands for unknown camera extrinsics, and both camera positions and environment abstraction have to be retrieved from a set of captures from the camera.

Another differentiation can be made based on the used representation for the 3D lines. The most recent works build representations for spatial lines that employ just 4 parameters, the same number as the degrees of freedom of a 3D straight line. Nevertheless, some other works published during this century still used 6 parameters for the representations. These parameters are either the two 3D endpoint coordinates[127, 43] or the closest 3D point on the

spatial line plus the direction vector. Another classification for the 3D abstraction methods can be done attending to the adjustment method, geometrical, by the Extended Kalman Filter (EKF), or by using non-linear optimization. The recent ones which use an EKF and aimed for SLAM are taking account of the camera velocity [127] and model based tracking [43].

Point based SfM pipelines may be teamed with line based solutions. The objective is to use the point based pipeline to obtain the camera poses. The camera extrinsics can be used to match detected lines by exploiting homography constraints, and to forward project the matched segments. The implementation of a line projection function has been proven profitable to source the correspondences from the camera extrinsics computed by a point-based SfM pipeline, given that this point cloud is dense enough to provide sufficiently accurate poses for the cameras[54], or more directly when the ground truth camera poses are used as input[160].

The first published methods based on minimization of an objective function, reconstructed the infinite line instead of the segment limited by the endpoints, provided partial extrinsic parameters of the cameras and generated initial estimates for the rest of them [134]. Bartoli et al.[7] goes a step forward and brings up the Plücker coordinates 4-parameter representation of lines, for the will of a more suitable SBA optimization. They also include the trifocal tensor[109] because they just used lines as input. Same as did [160], whose in addition avoided enforcing Plücker constraints to improve the SBA efficiency, by employing the Cayley representation of the Plücker coordinates. Their cost function computes the squared re-projection error from the sum of the squared shortest distance from each observed endpoint to the reprojected infinite line.

In 2011 Elqursh and Elgammal[26] proposed a line-based pose estimation method based on choosing groups of three pairs of matched lines between views. Two of the three lines have to be parallel, and perpendicular to the third one. Under this condition the relative camera rotation is computed, and the relative translation is obtained from two intersection points of the mentioned lines.

The above mentioned work about straight line detection from Salaün et al.[115] was followed by their SfM method for wide baselines and just 3 images [116]. Another sophisticated method for line segment based SfM was published in 2016 by Micusik and Wildenauer[92]. They captured line segments in fisheye lens images, based their matching on SIFT descriptors for the segment endpoints, estimated relative camera poses similarly to Elqursh and Elgammal[26],and built the 3D model incrementally: First computed relative camera orientations for all image pairs using the segment correspondences, then they simultaneously estimated their translation and built the 3D model by a simple RANSAC procedure.

After putting primitives in correspondence among views, we can distinguish three common further stages of a reconstruction process: The first stage for the spatial reconstruction is the estimation of camera poses by Structure-from-Motion (SfM), which outputs the estimated camera poses and reconstructed features. The second step is the computationally more expensive Multi-View Stereo (MVS)[41, 113] that can add up to millions of points into a dense cloud. The third stage is intended for post processing, and based on fitting more complex 3D elements to the estimated point clouds, including planes in a Manhattan World[63], planes reconstructed employing a stereo camera rig[110] or lines[160, 92]. However, these approaches are heavily dependent on the dense point clouds, and will not permit a feasible solution when a dense point cloud is not available. That applies, for example, to the case of UAVs, when an on-flight real time reconstruction is required, because expensive adjustments can not be delivered on time, when the video stream lacks high definition, the digital noise is persistent, or the received images have to be converted from an analog video source. Even with the appropriate setup, conditions related to poor texture of the surrounding objects, a poor environment illumination, blurring or the lack of high definition pictures of the scene available when the vehicle is moving fast, might compromise the building of a dense cloud.

In order to classify the observations, line clustering has been proven to be profitable at grouping lines before entering the pose estimation algorithms [120, 6]. The spatial position of clustered inlier matchings have to be recovered. This spatial dimension retrieval was

done classically by linear algorithms [79, 131], including a closed-form solution [145]. The addition of matrix factorization [136, 97] increased the applicability but did not solve the lack of support of matching outliers. Despite the Extended Kalman Filter (EKF) is not an optimal estimator [58], in the early 1990s some works[19, 139] used it. A minimum of six line correspondences in three images are required to solve this SfM problem [32]. The first nonlinear iterative approaches [78] were limited to a few lines in three views, but were later followed by methods using minimization of an objective function [134].

This clustering increases its relevance when the application is intended for UAV real-time video streams and other low quality image sources, because the number of matched lines drops to sparse densities. From point-based SfM pipelines it is expected to obtain low density point clouds, and a high rate of feature point matching outliers due to low image resolution, digital noise and changes in illumination conditions. Hence, matching algorithms exclusively based on straight segments detections avoid the source of uncertainties tied to the use of nearby feature points for matching. The major difference in performance is where the textures of surrounding objects are not available, and the line information is the only way to reach the convergence of the final adjustment.

A method to reconstruct a room from a single panorama image[162] was published in 2018. It used Manhattan World assumption and the vertical lines resembling the edges of the room are employed to shape its main geometry. After the geometry of the room is estimated, the texture is finally plotted on the sides of the model.

For large sets of images, the approach of perform line matching every view to the other ones can be problematic because the number of matching operations grows exponentially, so does the computational cost. In order to overcome this issue a common approach is to estimate the extrinsics for all the cameras by employing a feature-point based SfM pipeline, and use the homography constraints to match lines[55]. Moreover, it is easy to set a distance threshold for the cameras in order to decide whether two images get to be line matched. If they are too far apart the algorithm may decide that it will not be beneficial to match both im-

ages. Alternatively, the line matching algorithm can be executed just between images by following the ant colony optimization algorithm, like in the line-based SfM method from Shen and Dai[125] published in 2018. For this method, camera poses are estimated by matching the representative points lying on the lines, with the OpenMVG[100] pipeline.



## Chapter 4

# Straight segment detection

A straight line segment in 2D is a single primitive that can be defined by the coordinates of its two endpoints, or alternatively by just one point plus the angle and the length of the segment. Straight line detection is commonly rooted on edges detected in the image. A straight line detection algorithm generally works altogether with an edge detector that provides it with an edge skeleton comprised by curvy and straight sections. Every branch in the edge skeleton needs to be chopped into sections and fitted to straight segments. Therefore, in an image, several straight segments can be fitted to a curvy edge.

The early and basic detectors of edges in images are gradient based, meaning that the edge detection algorithm marks the points in the image with the highest contrast of pixel values, then draws the edge as a patch through them, adding joints when necessary. The problem with these are that changes in the global contrast will directly affect the edge detection, and moreover, filters can also affect the shape of edges and joints.

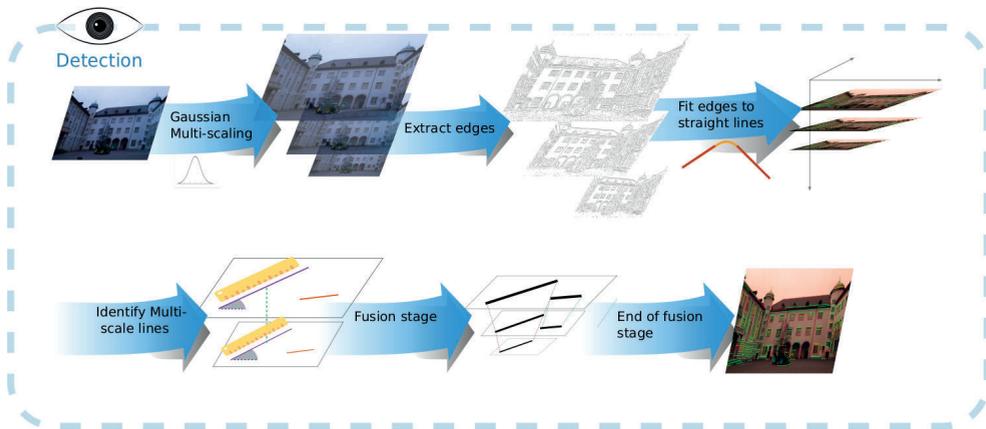
Fragmentation of lines occurs when an uniquely perceived segment entity is detected as an array of shorter segments. The generated problem is that these sections have to be merged into an unique entity before searching for segment matching hypotheses. One approach to

minimize fragmentation is to fetch segments in a way resilient to changes in contrast and illumination. It was proved that this can be accomplished by rooting straight segment detection on edges detected in the scale-space and employing the monogenic signal[130], because the maxima of differential phase congruency resemble the main edges on every picture in the scale-space.

The straight line segment detection algorithm described in this thesis is rooted on an edge detection method that exploits local energy and local phase. This approach has been proved profitable by the line detection method EDLines[2], and is also the same underlying idea as the approach used by Zhang and Koch[158] to source the line segments for their line matching method. Nevertheless, for the latter method edge detection is performed in several scaled versions of the original image. These downscaled versions of the image are referred to as octaves.

The first hypothesis of the presented line detection method is that a straight segment detection method will profit of detecting edges from the maxima of phase congruency in the scale-space[71], specially in sets of images with variable illumination conditions. The second hypothesis is that these edges can be reliably fitted to straight line segments by using linear regression, like performed when fitting data to lines in a plot. The third hypothesis is that additional appearance information can be extracted from the scale-space after merging the straight lines detected in different octaves. Therefore lines detected in the original image can be classified according to the depth it was detected in the scale-space, meaning in how many octaves the line segment has been observed. The goal of the method is to input an image, and without additional information detect solely straight line segments in the whole area of the image, avoiding fragmentation, and resiliently to changes in contrast and illumination.

The remaining of the chapter follows the Section 4.2, that sets the theoretical bases for the energy based method employed to extract the edges. The description of the method is in Section 4.2. The method is proved experimentally in Section 4.5.



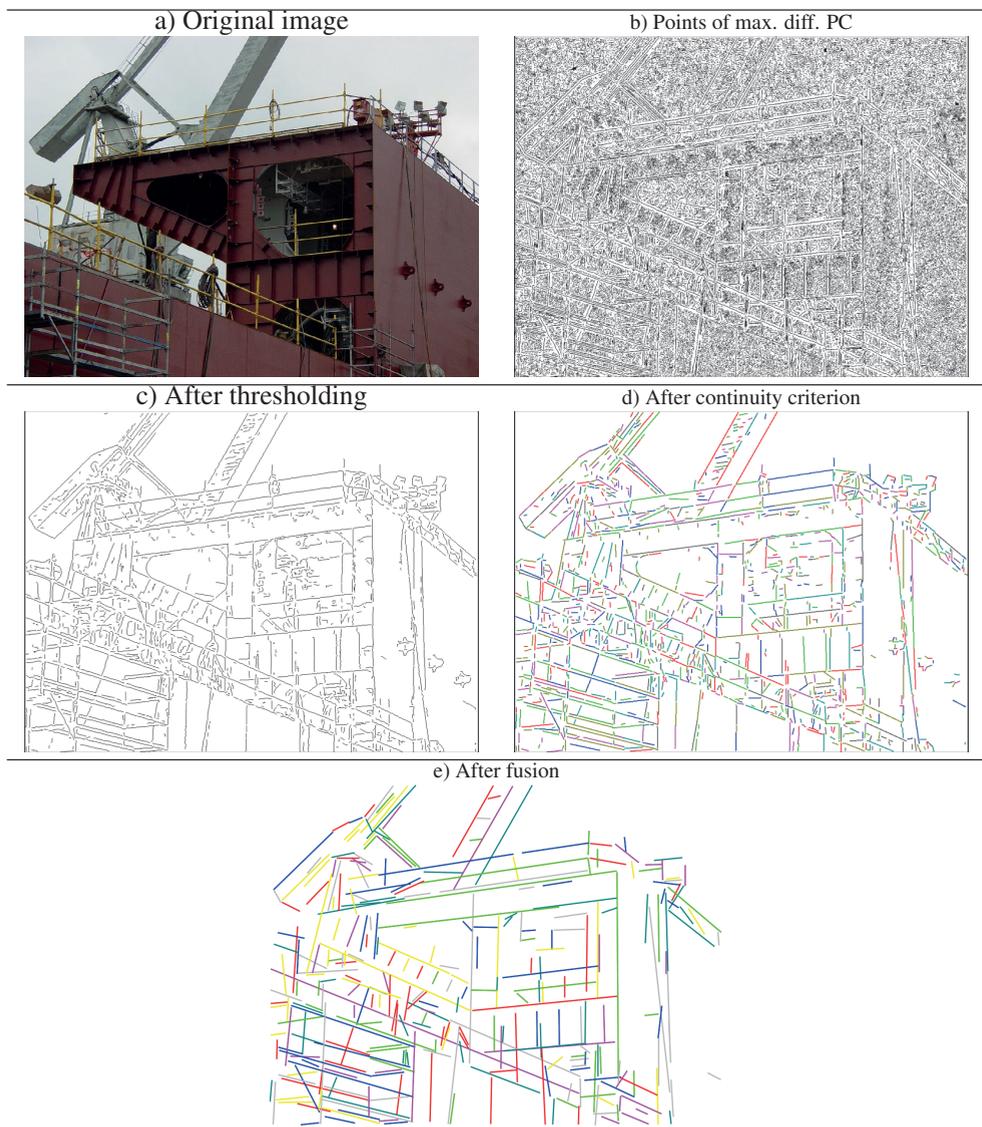
**Figure 4.1:** Representation of the different steps of the line detection method described in this chapter.

## 4.1 Overview of the Line Detection method

Fig. 4.1 is a graphical representation of the step-by-step process for straight line segment detection in an image. In this section the different steps of this detection process are reviewed: First the creation of a scale-space by downsizing the input image and applying a Gaussian filter on the octaves for successively smoothing them. Secondly, edges are detected in the octaves by looking for the extrema of differential phase congruency. Thirdly, these curvy edges are fit to straight lines by linear regression. The fifth step relates the straight segments between the different octaves according to their appearance. Finally these segments detected in the octaves are fused with the ones detected in the original image. These extracted lines are classified according to the number of octaves they were found in.

Table 4.1 shows an example of the process by plotting intermediate images as result of the steps that lead to the straight line detection: a) shows the original image. The points where the Fourier components of the image function get maximum phase congruency are marked in black in the image b), and are referred to as points of *differential phase congruency*.

This remainder skeleton of points is filtered by applying a threshold attending to its *phase congruency*, by removing the ones with slope lower than 0.2. Furthermore, the connected components drawing an angle greater than 5 degrees are removed, same as done with the ones featuring a length under 5 pixels. This let us with the image c). Next, the remainder of points are fitted to straight lines by linear regression, like explained in Subsection 4.4. This leaves us with the image d), which shows the separated line segments. Finally, an algorithm fuses straight segments into unique entities attending to the distance between endpoints and its inclination, coming up with image e).



**Table 4.1:** This table follows the process of straight line detection in an image, and it is better viewed in a computer monitor with a 400%. a) Original image. b) Marked in black, the points of differential phase congruency. c) Points are filtered using a threshold attending to the Phase Congruency, length and inclination. d) The remainder of points are fitted to straight lines by linear regression. e) After the fusion algorithm for fragments of segments.

## 4.2 Energy based edge detection

There are three concepts that are referred to in the description of the method, and these must be understood separately: the *Monogenic Signal*, the *Gaussian Scale-Space* and the *Phase Congruency*. The applicability and advantages of the presented edge detection method are dependent on how these concepts are teamed.

### The Monogenic Signal

A 2D image can be thought as a two dimensional signal. The energy based edge detection approaches use this abstraction, and for each point in the image the signal has an orientation, a spectra of frequencies and phases. In order to leave space to the frequency dimension to live, we have to think about an abstract extension of the images. We can think this third dimension as space in-between scales, filled by waves created by the image signal in the image. The Riesz transform for two dimensions is a generalization of the Hilbert transform [35] for these two dimensions. Let the vector  $\mathbf{H} = (\mathbf{H}_1, \mathbf{H}_2)$  be the transfer function of the Riesz transform, and represent filters in the frequency domain:

$$\mathbf{H}_1(\mathbf{u}_1, \mathbf{u}_2) = i \frac{\mathbf{u}_1}{\sqrt{\mathbf{u}_1^2 + \mathbf{u}_2^2}} \quad \text{and} \quad \mathbf{H}_2(\mathbf{u}_1, \mathbf{u}_2) = i \frac{\mathbf{u}_2}{\sqrt{\mathbf{u}_1^2 + \mathbf{u}_2^2}}. \quad (4.1)$$

The convolution kernel for the filter  $\mathbf{H}$  respect to the coordinates in the image is:

$$(f_{o1}(x_1, x_2), f_{o2}(x_1, x_2)) = \left( -\frac{x_1}{2\pi(x_1^2 + x_2^2)^{\frac{3}{2}}}, -\frac{x_2}{2\pi(x_1^2 + x_2^2)^{\frac{3}{2}}} \right). \quad (4.2)$$

If we combine the image signal  $f$  with its Riestz transform, we can obtain a multi-dimensional generalization of the analytic signal, called the *monogenic signal*:

$$f_M(x_1, x_2) = (f(x_1, x_2), (\mathbf{f}_{o1} * \mathbf{f})(x_1, x_2), (\mathbf{f}_{o2} * \mathbf{f})(x_1, x_2)) \quad (4.3)$$

The local amplitude of  $\mathbf{f}_M$  is the vector norm of  $\mathbf{f}_M$ , and denoted as  $\mathbf{A}_f$ :

$$|\mathbf{f}_M| = \mathbf{A}_f = \sqrt{\mathbf{f}^2 + (\mathbf{f}_{o1} * \mathbf{f})^2 + (\mathbf{f}_{o2} * \mathbf{f})^2} \quad (4.4)$$

For a triple we need two phases. Using the standard spherical coordinates:

$$\mathbf{f} = \mathbf{A}_f \cos(\phi) \quad (4.5)$$

$$(\mathbf{f}_{o1} * \mathbf{f}) = \mathbf{A}_f \sin(\phi) \cos(\theta) \quad (4.6)$$

$$(\mathbf{f}_{o2} * \mathbf{f}) = \mathbf{A}_f \sin(\phi) \sin(\theta), \quad (4.7)$$

where  $\phi \in [0, 2\pi)$  and  $\theta \in [0, \pi)$ .

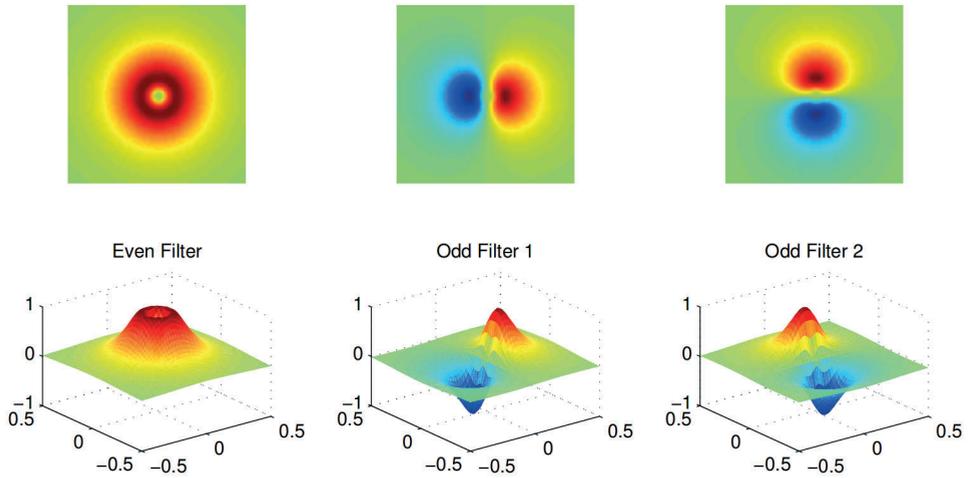
In order to filter the Riesz transform to reduce its infinite pulse to a local filter mask, a triple of spherical quadrature filters (SQF) is applied. This triple comprises the radial band-pass filter and the radial bandpass filtered kernels. In this case the bandpass filter employed is a lognormal filter, which is a Gaussian filter if considered in logarithmic scale.

$$G_e(\mathbf{u}_1, \mathbf{u}_2) = \mathcal{F}\{f_e(x_1, x_2)\} = \exp\left(-\frac{(\log(\sqrt{\mathbf{u}_1^2 + \mathbf{u}_2^2}/2^k))^2}{2(\log(\omega))^2}\right), \quad (4.8)$$

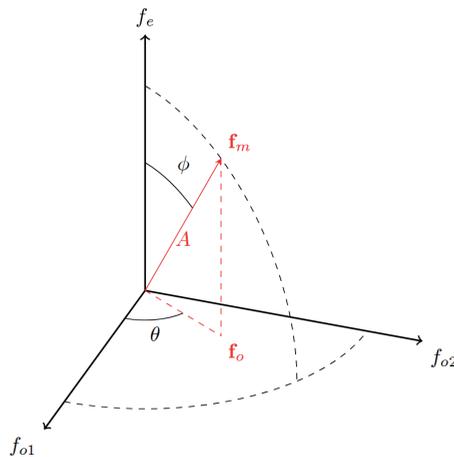
where  $k$  indicates the octave (the center frequency) and the bandwidth-parameter  $\omega$  is either 0.55 for two octaves, or 0.41 for three octaves. Therefore, the triple of SQF that is applied is:  $(\mathbf{G}_e, \mathbf{H}_1 \mathbf{G}_e, \mathbf{H}_2 \mathbf{G}_e)$ . As explained by Felsberg and Sommer[35], the response of these SQF is drawn in Fig. 4.2. The components of the monogenic signal are represented in Fig. 4.3, as described by[35]. An example of the application of each one of the components of the filter is shown in Fig. 4.4.

### The Gaussian scale-space

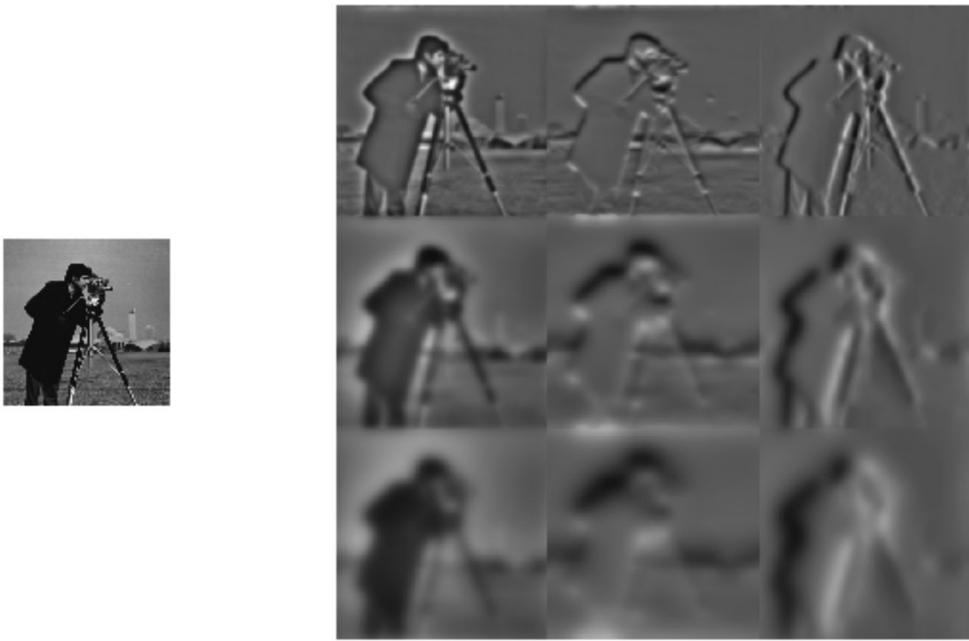
The Gaussian is the only function that is located on both space and frequency domains. The natural scale parameter to vary in the calculation of phase congruency is the size of the analysis window over which to calculate local frequency information. Scale is varied using high-pass filtering rather than low-pass or band-pass filtering, in order to avoid losing significance of the edges with more relevance in the image.



**Figure 4.2:** Frequency-domain representations of a set of spherical quadrature filters. Figure from [13].



**Figure 4.3:** The decomposition of the components of the monogenic signal. Figure from [13].



**Figure 4.4:** On the left, test image. On the right its monogenic signal. It is presented on increased scale from up to down. From left to right it is decomposed on their even and both odd components. Figure from [13].

Line detection starts by constructing a Gaussian scale-space pyramid of the input image by repeatedly smoothing it with a Gaussian and subsequently downsampling [35, 71, 70] the original image into octaves. For each scaled image, edge features are extracted by means of *phase congruency* [71], which is a dimensionless quantity that is invariant to changes in image brightness and contrast. These edge features are then locally approximated to line segments according to a continuity criterion, and finally, the multi-scale information is combined to perform a more meaningful merging of those line segments which may correspond to fragments of the same real line.

Given a grayscale image, Gaussian filtering is applied in order to suppress image noise and smooth out the image. Gaussian filters of different sizes are applied in order to downscale

the original image into different octaves.

The Gaussian scale-space of an image is a family of functions  $L(x, y; s)$  defined by the convolution of a Gaussian  $G(x, y; s)$  with the input image  $I(x, y)$ :

$$L(x, y; s) = G(x, y; s) * I(x, y) \quad (4.9)$$

where  $*$  is the convolution operator in  $x$  and  $y$ , and  $s$  is the scale parameter.

The Gaussian scale-space is sampled in both space and scale to obtain a pyramid representation. Our results are obtained by using three images:  $I(x, y)$ ,  $L(x, y; s_1)$  and  $L(x, y; s_2)$ , using a convolution kernel of size  $5 \times 5$ , where  $s_1 = 1.3$ ,  $s_2 = 1.6$ , and where the last two images are downsampled by a factor of  $M_1 = \sqrt{2}$  and  $M_2 = 2$ , respectively.

### Phase congruency

An application of the spherical quadrature filters is the detection of edges and lines in images. If we take the local amplitude as measurement for setting the edges in images, the problem that rises is how to fix the threshold. If it is too low, a lot of false-positives are raised. On the other hand, if the threshold is set too high, many edges will be missed with low local contrast. Nevertheless, since the phase information is independent of the local amplitude, it is possible to use the *phase congruency*: Edges in images are ideally described by dimensionless quantities, and these should be independent of illumination and magnification. In order to predict edges using such parameters it is not possible to use the gradient magnitude, but instead the local energy model of feature detection[99]. Phase congruency is a dimensionless measure of feature significance. This model postulates that features are perceived at points of maximum phase congruency in the image. The phase congruency function is developed from the Fourier series expansion of a signal[71]:

$$F(x) = \sum_n A_n \cos(n\omega x + \phi_n), \quad (4.10)$$

where  $\omega$  is a constant. The phase congruency function is defined as

$$PC(x) = \max_{\theta \in [0, 2\pi]} \frac{\sum_n A_n \cos(n\omega x + \phi_n - \theta)}{\sum_n A_n}. \quad (4.11)$$

The value of the added phase  $\theta$  that maximizes  $PC(x)$  in the equation 4.19 is the weighted mean phase angle of all the Fourier terms at the point being considered. Finding the values of  $x$  where phase congruency is a maximum is approximately equivalent to finding where the weighted standard deviation of phase angles is a minimum.

Phase congruency is usually weighted by some measure of the spread of the frequencies that are present at each point in an image. In order to extrapolate the concept of phase congruency to two dimensions, a spreading function has to be applied across the filter, perpendicularly to its orientation. A good choice for this function is the Gaussian function, because the convolution does not corrupt the phase data in the image. In the frequency domain, any function smoothed with a Gaussian suffers amplitude modulation of its components, while phase remains unaffected.

### 4.3 Edge Detection

Classical approaches to edge detection involve the localization of significant local changes in image intensity. However, these methods are sensitive to scene illumination, blurring, and magnification[161]. Moreover, most of them are optimized to detect only step-edges and will give spurious responses when applied to real image edges, which are a combination of step, peak, and roof profiles[107]. In contrast, rather than assuming that a feature is a point of maximal intensity gradient, the local energy model postulates that features are perceived at points in an image where the Fourier components are maximally in phase, equivalently, points of maximal *phase congruency*[98, 99], which is a dimensionless quantity that provides a contrast invariant way of identifying features[72]. Due to practical difficulties in estimating phase congruency, most of the existing methods use a related quantity, the local energy, for

feature detection[12], but the problem is that this quantity is not invariant to illumination changes and contrast. In last years, a few methods to directly compute phase congruency have been developed by Kovessi[69]. However, these methods require to compute the responses of a bank of filters over many orientations and scales, with the consequent high computational cost.

In this thesis, *differential phase congruency* proposed by Felsberg and Sommer[34] is used to achieve robustness to different illumination conditions, which has the consistency of Kovessi approach while solving its drawbacks. Differential phase congruency is derived on the basis of the monogenic scale-space, which arises from the combination of the monogenic signal with a Poisson scale-space representation[34, 37]. The monogenic signal  $I_M(x)$  is a 2D generalization of the analytic signal, that uses the Riesz transform instead of the Hilbert transform.

$$I_M(x, y) = (I(x, y), (f_{o1} * \mathbf{I})(x, y), (f_{o2} * \mathbf{I})(x, y))^{\top} \quad (4.12)$$

where  $I(x, y)$  is the image signal, and  $(f_{o1}, f_{o2})$  are the components of the Riesz transform, which is a scalar-to-vector signal transformation with an impulse response given as:

$$\mathbf{h}(x, y) = (f_{o1}(x, y), f_{o2}(x, y))^{\top} = \left( \frac{x}{2\pi\|x, y\|^3}, \frac{y}{2\pi\|x, y\|^3} \right)^{\top} \quad (4.13)$$

The Poisson scale-space representation of the monogenic signal forms the monogenic scale-space  $\mathbf{I}_M(x, y; s)$ :

$$\mathbf{I}_M(x, y; s) = (u(x, y; s), \mathbf{v}(x, y; s))^{\top} \quad (4.14)$$

where  $\mathbf{v}(x, y; s)$  is the Riesz transform of  $\mathbf{u}(x, y; s)$ , which is the image convolved with the Poisson kernel  $p_s(x, y) = s \cdot (2\pi(s^2 + \|(x, y)\|^2))^{-3/2}$ . In this framework, points of differential phase congruency are defined by the zeros of the scale derivative of the local phase-vector  $\partial_s \mathbf{r}(x, y; s) = 0$ , defined as:

$$\mathbf{r}(x, y; s) = \frac{\mathbf{v}(x, y; s)}{\|\mathbf{v}(x, y; s)\|} \cdot \arctan \left( \frac{\|\mathbf{v}(x, y; s)\|}{u(x, y; s)} \right) \quad (4.15)$$

The advantage is that for intrinsically one-dimensional neighborhoods (i.e. neighborhoods where the signal varies only in one direction), this derivative can be expressed analytically, and points of phase congruency can be computed directly as:

$$\partial_s \mathbf{r}(x, y; s) = \frac{u(x, y; s) \partial_s \mathbf{v}(x, y; s) - \mathbf{v}(x, y; s) \partial_s u(x, y; s)}{u(x, y; s)^2 + \|\mathbf{v}(x, y; s)\|^2} = 0 \quad (4.16)$$

In practice, similarly to the use of quadrature filters to estimate the analytic signal[12], the monogenic signal is estimated by convolving the image signal with a spherical quadrature filter (SQF)[33], which is a triplet consisting on a band-pass filter and its Riesz transform. The band-pass filter used is the Difference of Poisson (DOP), which is combined with the corresponding Difference of Conjugate Poisson (DOCP) filters to form the SQF. Their impulse responses are:

$$b(x, y; s) = \frac{s}{2\pi(s^2 + \|(x, y)\|^2)^{3/2}} - \frac{\hat{s}}{2\pi(\hat{s}^2 + \|(x, y)\|^2)^{3/2}} \quad (4.17)$$

$$\mathbf{c}(x, y; s) = \left( \frac{1}{2\pi(s^2 + \|(x, y)\|^2)^{3/2}} - \frac{1}{2\pi(\hat{s}^2 + \|(x, y)\|^2)^{3/2}} \right) (x, y) \quad (4.18)$$

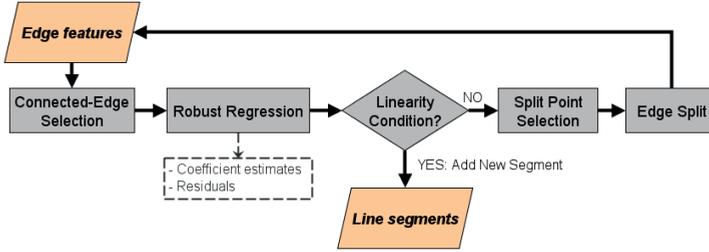
where  $s$  is the scale parameter, and  $\hat{s} = s + \Delta s$ . Results are obtained with  $s = 1$  and  $\hat{s} = 3$ . And then, for intrinsically one-dimensional neighborhoods, points of differential phase congruency are obtained as those  $(x, y)$  that satisfy:

$$\begin{aligned} & (b(x, y; s) * I(x, y)) \cdot (\partial_s \mathbf{c}(x, y; s) * I(x, y)) \\ & - (\mathbf{c}(x, y; s) * I(x, y)) \cdot (\partial_s b(x, y; s) * I(x, y)) = 0. \end{aligned} \quad (4.19)$$

By linear regression these zeros are detected with subpixel accuracy.

#### 4.4 Straight line segments detection

Edge features may correspond to straight lines, but also curved lines, which can be locally approximated to straight line segments. Consequently, a linearity criterion is required to



**Figure 4.5:** Continuity criterion for the extraction of line segments from edge features of an image.

decide whether a connected-edge fits well to a straight line, but also a split criterion is needed to select the break point to divide those edges that do not satisfy the linearity condition.

### From edges to straight lines

Fig. 4.5 shows a scheme of the applied process, which is carried out iteratively until all the connected-edges are fitted to line segments. As seen in this figure, a **robust regression** is performed for each connected-edge to estimate the corresponding straight line segment, which is done to reduce the influence of possible outliers due to image noise. This robust regression is implemented by using iteratively reweighted least squares (IRLS) with a bisquare weighting function[57], which involves the following steps: Consider an ordinary robust regression with the model:

$$y_i = x_i' \beta + \sigma \varepsilon_i, \quad (4.20)$$

where  $\beta$  is the regression parameter or slope of the line, the errors  $\{\varepsilon_i\}$  are independent and identically distributed, and  $\sigma$  is the scale parameter.

1. Fit the model by weighted least squares, solving the normal equations:

$$\left( \mathbf{W}^{1/2} X \right)^\top \left( \mathbf{W}^{1/2} X \right) \beta = \left( \mathbf{W}^{1/2} X \right)^\top \left( \mathbf{W}^{1/2} y \right) \quad (4.21)$$

where  $\mathbf{W} = \text{diag}(w_1, \dots, w_N)$  is the diagonal matrix built from the weights (which are set to 1 at the first iteration),  $\beta$  is a column vector with the two regression coefficients, and  $y = [y_1, \dots, y_N]^\top$  and  $\mathbf{X} = \begin{bmatrix} 1 & | & (x_1, \dots, x_N)^\top \end{bmatrix}$  are a column vector and a  $2 \times N$  matrix respectively, with  $(x_i, y_i)$  the image coordinates of the  $N$  points forming the connected-edge. These normal equations are solved by means of a **QR** decomposition of the matrix  $(\mathbf{W}^{1/2}\mathbf{X})$ , given by:  $\mathbf{W}^{1/2}\mathbf{X} = \mathbf{QR} = [\mathbf{Q}_1 \mathbf{Q}_2] [\mathbf{R}_1 \mathbf{0}]^\top = \mathbf{Q}_1 \mathbf{R}$ .

2. Compute the adjusted residuals given by:

$$r_i^{\text{adj}} = \frac{r_i}{\sqrt{1 - h_i}}, \quad (4.22)$$

where  $r_i = y_i - \sum_j \mathbf{X}_{i,j} \beta_j$  are the residuals, and  $h_i$  are the leverage values from a least-squares fit (the diagonal elements of the hat matrix,  $\mathbf{H} = \mathbf{Q}_1 \mathbf{Q}_1^\top$ , of the first iteration), which adjust the residuals by reducing the weight of high-leverage data points.

3. Compute the standardized adjusted residuals as:

$$u_i = \frac{r_i^{\text{adj}}}{K \cdot \sigma}, \quad (4.23)$$

where  $K = 4.685$  is a tuning constant, and the scale parameter  $\sigma = 1.4826 \cdot \text{MAD}$  represents an estimate of the standard deviation of the error term, estimated by using the *median absolute deviation* (MAD) of the adjusted residuals. The constant 1.4826 makes the estimation unbiased for the normal distribution.

4. Compute the bisquare weights as:

$$w_i = \begin{cases} \left(1 - (u_i)^2\right)^2 & , |u_i| < 1 \\ 0 & , |u_i| \geq 1 \end{cases} \quad (4.24)$$

5. Return to the first step until the fit converges, or until it reaches a maximum number of iterations (set to 50).

### Linearity and splitting criteria for edges

Once the robust fit is performed, the **linearity criterion** accepts a line segment as valid if:

1. The dispersion in the orientation (computed following[73]) of the connected-edge is less than  $\pi/12$ . This threshold roughly corresponds to the error of 1% for the approximation of  $\sin \theta \approx \theta$ , which corresponds to  $\theta = 0.244$  radians ( $14^\circ$ ).
2. The maximum of the residuals is less than 5 pixels, in order to chop ample arcs. It has been proved empirically to be a safe perpendicular distance from the fitted straight line to the edge pixels, by letting through thick or blurred segments that are still considered valid, while being a good compromise by considering to bisect long arcs.

The connected-edges that are not accepted as valid segments are split in two. The **splitting criterion** selects points on the edges that did not meet the *linearity criterion* to be candidate points where the edges are chopped. These are the points on the edge whose derivative of the local orientation meet any of the following conditions:

1. It is a local maximum, because the high peaks of the derivative of the orientation coincides with the points of inflection of a curve.
2. It exceeds the predefined angular threshold of  $\pi/12$ , because the edge might be reaching an inflection there, or a change of direction of the line.

For each splitting point candidate, the total dispersion in the orientation on both sections of the edge is computed, so the splitting point candidate with the least total dispersion is selected as the final splitting point. Edges shorter than a threshold of 5 pixels are discarded because are considered too small for being practical with the proposed applications. The image c) in Table 4.1 shows the line segments extracted from the edge detector response for an image.

### Multi-scale fusion stage

The previous subsection went through the extraction of straight line segments for the original image and each downscaled octave. Now, at this fusion stage, all these multi-scale sets must be combined to provide a single set of lines that corresponds with the perceived segments in the original input image as precisely as possible. Thus, the purpose of this stage is to reduce the presence of fragmented and overlapping segments which may correspond to the same perceived line, and this is done through the use of the multi-scale information provided by the previous stages. It is clear that the lines extracted from the smallest octaves (lower resolution) will be located with a more inaccurate pose due to Gaussian smoothing and downsampling. But on the other hand, the lines in these small octaves will be more significant due to the reduced noise. The line set extracted from the original image will be located more precisely, but it may contain spurious lines caused by noisy edges, which could lead to erroneous merging. For this reason, the scale-wise merging is spreading from the smallest octaves to the original image, looking for merging the significance given by the segments in the smallest octaves with the precision in the location of the segments in the original image. Two steps are required: the first one is to identify homologous lines along scales for providing the possibility of extending a merge from high to low scales, and the second is to conduct merging when the appropriate conditions are met.

Two lines at different scales are identified as homologous based on four parameters, which are assigned with their respective distance thresholds:

1. The difference in the orientations of both lines  $\omega$  is set to be less than  $\pi/12$ , preventing the error for the approximation  $\sin \omega \approx \omega$  of exceeding 1%.
2. The distance between midpoints in the normal direction is configured to be no greater than 14 pixels. This distance might seem excessive at first sight, but has been proved profitable for short segments that might be detected with high uncertainty in their

slope for the smaller scales. This is mainly due to Gaussian smoothing or noise. The slight difference in slope plus the uncertainty in the point location after scaling can be combined with a shift of location of the segment along the line, bumping the normal distance between endpoints further than expected for the same actual segment in different scales.

3. The maximum distance between endpoints has to be less than three times the length of the longer segment:  $3 \cdot \max(L_i, L_j)$ . This similarity condition applies to ensure the lengths of the segments are roughly proportional between scales.
4. The minimum distance between endpoints has to be less than 24 pixels, because the minimum possible line for detection for every scale has been defined as 5 pixels, so 24 is three times the length of this segment in the higher scale. By losing restrictions between the location of endpoints we allow incomplete detections in smaller scales to still add in with information about the importance of the line segment detected in the original scale.

If there are multiple candidates for the same line, it will take the closest based on the distance between endpoints. Moreover, the algorithm may encounter two aligned segments located close together in a scale. In this case, the *merging criterion* is applied in order to discern if both segments resemble the same real segment that was detected as two fragments in the scale, and merge them accordingly:

**Scale-wise merging criterion:**

Two aligned segments are selected as merging candidates based on three parameters:

1. The difference in the orientations of both aligned segment to be less than  $\pi/12$ , for the same reason that in the *linearity criterion*.

2. The distance between midpoints in the normal direction will be no greater than just 2 pixels because here we are just measuring alignment.
3. The minimal distance between endpoints is fixed to  $0.02L$ , where  $L$  is the largest dimension of each image.

The iterative process forces the new merging candidate segment to undergo the *linearity condition* described in the previous subsection, in order to be finally accepted. Then, newly merged lines are introduced again into an iterative process, which scans through all scales until no mergings are left.

The final output is composed just by segments detected in the the original scale, whose length exceeds  $0.025L$ .

## 4.5 Experimental results for line detection

Line detection methods are evaluated employing Ground Truth assessment, based on results against public datasets. A human observer will mark the detected lines and assess if they represent a true segment and if it is completely addressed by the detected feature. Detection has been evaluated against synthetic and real public images.

Several aspects of the proposed approach are evaluated in this section. Regarding the line detection method we will focus on evaluating its robustness against different illumination conditions and levels of noise, with the aim of quantitatively measure the benefits of an energy-based edge detector. In addition, we also want to check the accuracy reached in minimizing the number of fragmented and overlapping lines by means of the proposed line extractor. On the other hand, for the line matching method we will focus on evaluating the performance of line neighborhoods to deal with low-textured scenes, containing objects with homogeneous surfaces and many repetitive patterns, which are common in industrial environments. Moreover, we want to test its accuracy under a wide range of differences in perspective

between the pair of views, and also the quality of the resulting matching information from a photogrammetric point of view.

In order to test line detection method we used a set of **synthetic images** subject to different illumination conditions and levels of noise. These images contain 48 perceived lines with different contrast settings on both sides, and a closed curve to observe its effect on the straight line detection. The illumination conditions are not global brightness changes, but rather different settings of lights and shadows which try to simulate a more realistic scene. For each one of these images with different illumination conditions, four different levels of Gaussian noise are added with zero mean and variances 0.002, 0.006, 0.010 and 0.014 (intensity values of the images are normalized in the interval  $[0, 1]$ ). Furthermore, the line detection method is also tested on a real image under different conditions of illumination, level of noise, and average gray levels. On the other hand, a set of wide-baseline image pairs containing **real industrial environments**, specifically in the field of shipbuilding, were used for the quantitative comparison. These images were taken by workers without stopping the production activities, under uncontrolled conditions. The images contain a set of objects, characterized by containing low-textured surfaces and many repetitive patterns. There are also many of them containing objects like temporary marks made by the industry workers, clothes, cables, etc. which lack clear lines, and they are therefore classified as non-relevant for line matching. In addition, a set of images taken from public databases[122, 30] are used. The selected images are of buildings because they share similar characteristics with industrial parts, either for being images taken under uncontrolled illumination conditions, or for containing many repetitive patterns of lines, and also for including views with small and moderate view-point changes, different scale, and occlusions. The images are represented in Fig. 4.12 and Fig. 4.12.

Our results are obtained with the aforementioned fixed parameters, and are compared with two state-of-the-art line matching approaches: *Line Signatures* (LS)[142], and the method proposed by Zhang et al. (AG)[158]. Both implementations are supplied by their authors,

and are selected due to their good performance to deal with a wide range of image transformations.

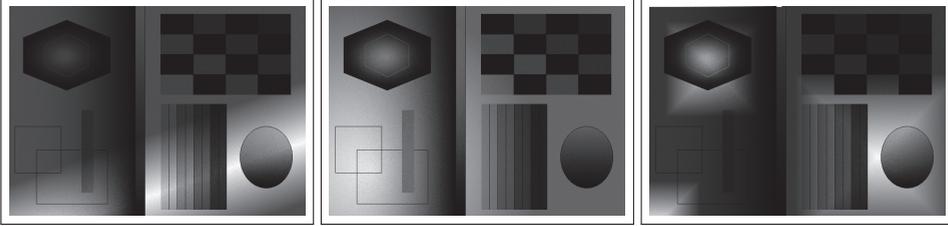
### **Line Extraction Results**

Results of line detection with the proposed method are shown in Fig. 4.6 and Fig. 4.7, altogether with the results of two methods that output line detections: LS[142] and AG[158]. In these results it can be noted the different segment lengths obtained for each method. For instance LS[142] is often marking non relevant lines, and the fragmentation of both LS[142] and AG[158] is more severe than in the proposed method. Therefore a comparison of number of extracted lines does not reflect the performance of methods against Ground Truth.

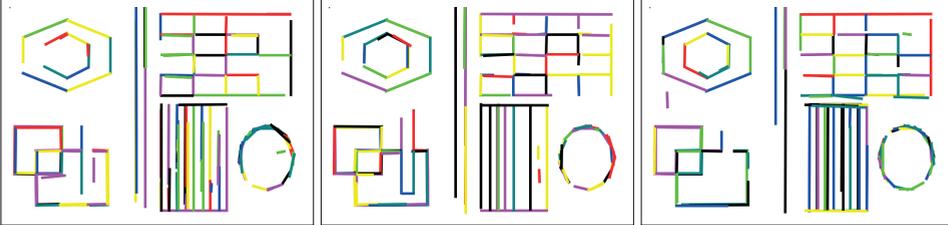
Fig 4.6 shows the line extraction results for three of the synthetic images under different illumination conditions. It can be seen that both AG and LS overestimate the total number of lines on the images, exceeding in all of them a minimum of 221 and 147 respectively, which is more than three times the number of perceived lines (48). Moreover, for one of the images, LS extracted 288 lines which is more than five times the number of perceived lines. As seen in the figure, this overestimation is due to many overlapping segments are extracted from the same single lines, and also because many straight lines are fragmented into several segments. In contrast, the number of lines extracted by our approach in the three images (58, 59 and 60 lines) is very close to the real number of perceived lines (48). Furthermore, it can be seen that there are no overlapping detected segments, even in the polygonal approximation of the ellipse, and also that almost no straight line is fragmented, which is due to the proposed fusion stage.

Table 4.2 shows in a quantitative way the dependence of both LS and AG on illumination conditions and noise level. Especially note that the number of lines extracted by LS varies very strongly among different conditions, becoming more than twice in some cases. This is because both LS and AG use gradient-based methods for line extraction, which are very

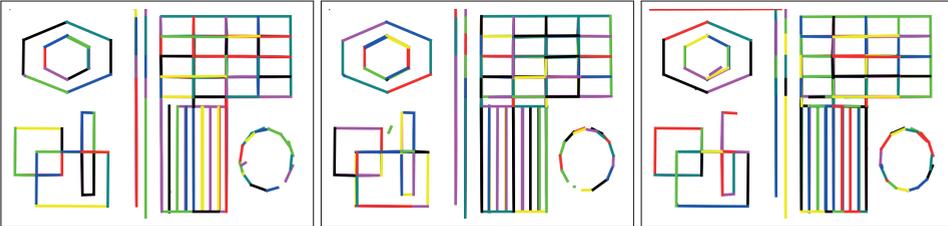
[Three synthetic images with different illuminations (48 perceived lines).]



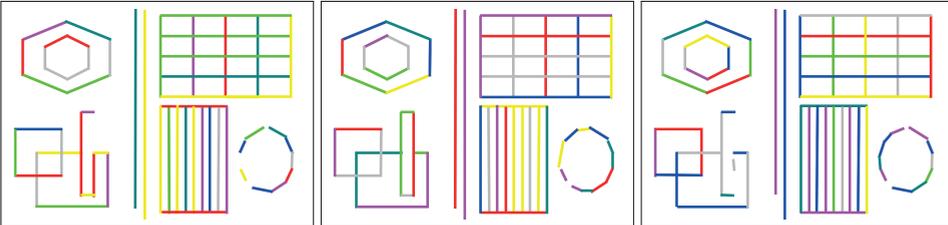
[Detection results with the method AG: 147, 136 and 137 extracted lines.]



[Detection results with the method LS: 236, 221 and 288 extracted lines.]



[Detection results with the Proposed method: 58, 59 and 60 extracted lines.]

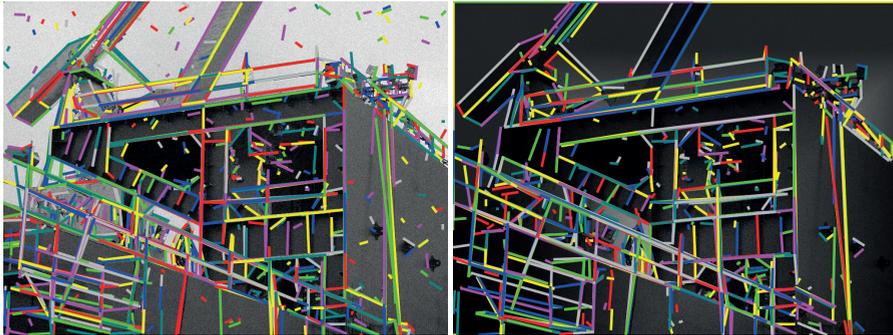


**Figure 4.6:** Line segment detection results. Lines extracted from synthetic images under different illumination conditions. Colors were randomly assigned to segments in order to distinguish them from their neighbors. Quantitative results are shown in table 4.2.

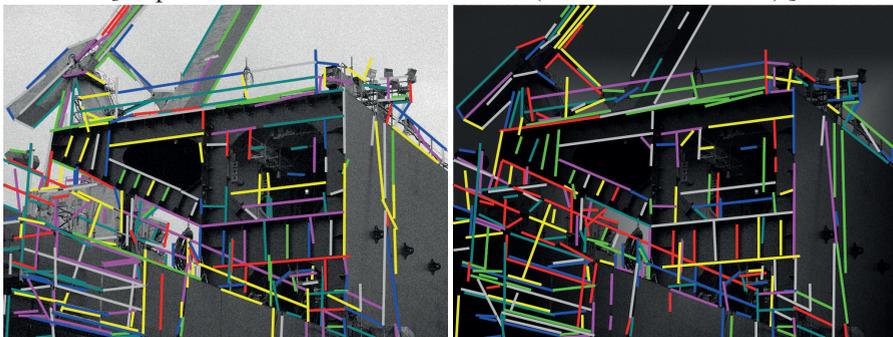
[AG: 975 and 778 lines extracted (a difference of 197 lines).]



[LS: 2632 and 2015 lines extracted (a difference of 617 lines).]



[Proposed: 292 and 283 lines extracted (a difference of 9 lines).]

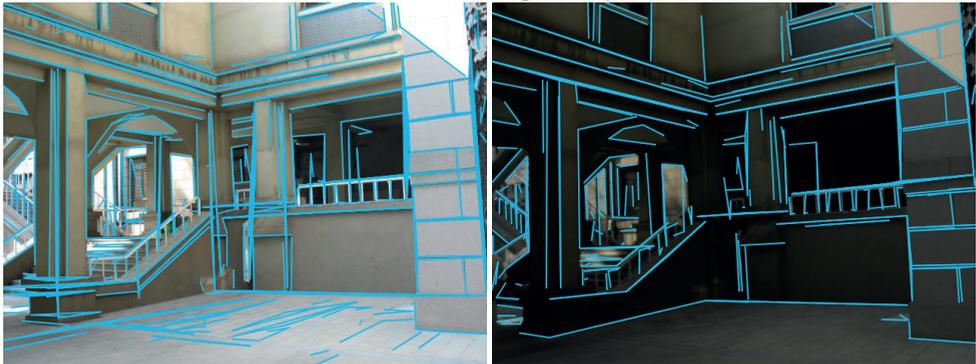


**Figure 4.7:** Line extraction results under different illumination conditions for the same real image (the same image in Fig. 4.7).

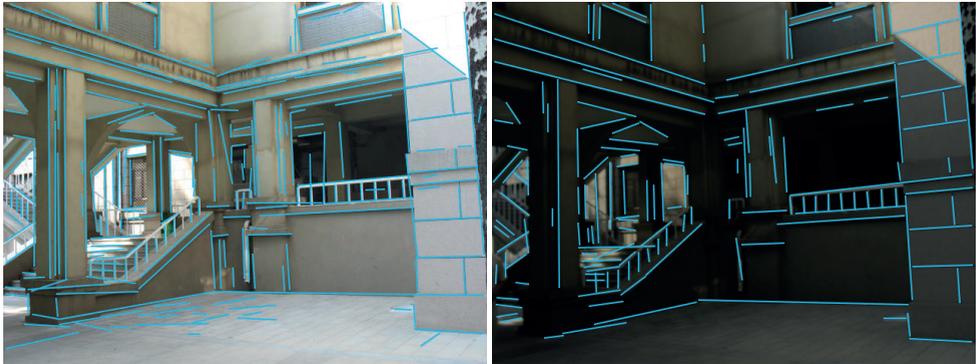
[Original images]



[LS: 757 and 405 lines extracted respectively. Times: 1s and 0.8s.]



[result of the SALM method: 373 and 386 lines respectively. Times: 3.5s and 3.9s.]



**Figure 4.8:** Line extraction results for the images in the dataset *outdoorlight*[74].

[Original images]



[LS: 622 and 938 lines extracted respectively. Times: 0.8s and 0.9s.]



[result of the SALM method: 346 and 231 segments respectively. Times: 3.6s and 3.4s.]



**Figure 4.9:** Line extraction results for the images in the dataset *drawer*[74].



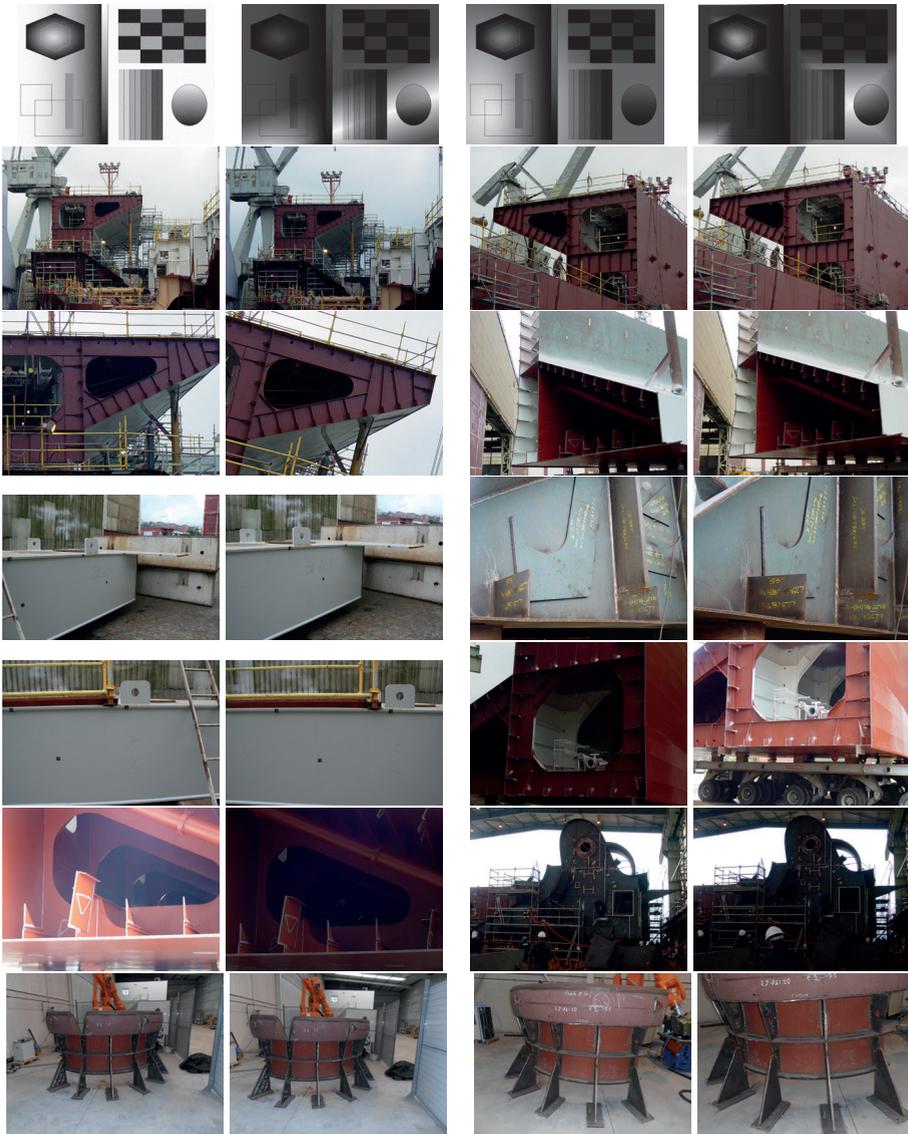
**Figure 4.10:** Line extraction results for the images in the dataset *lowtextureI*[74]. On the left, the result of LS with 194 lines extracted and cost of 0,7s. On the right, the result of SALM featuring 56 segments and obtained in 3.8s.

**Table 4.2:** Total number of lines extracted for the synthetic images. On the left: fixed illumination conditions and different levels of Gaussian noise  $G(\mu, \sigma^2)$ . On the right: different illumination conditions and same noise level. The number of perceived lines in the images is 48.

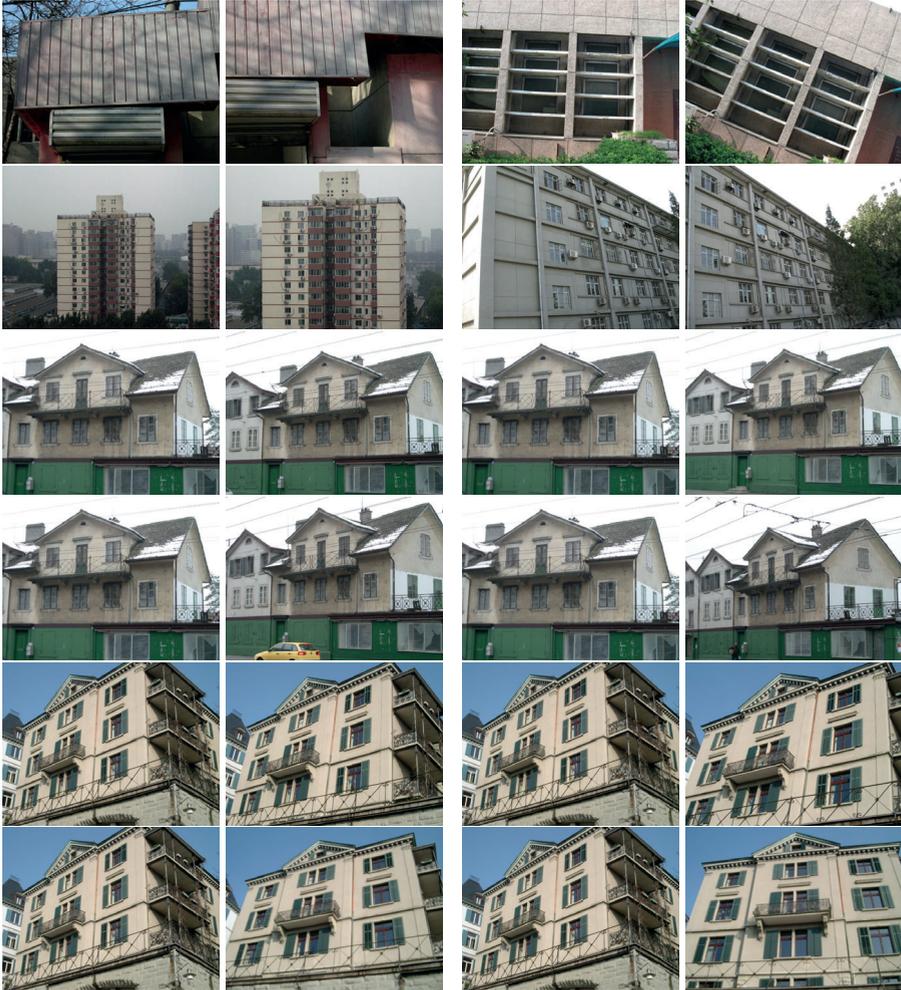
Noise	AG	LS	Proposed	Illumination	AG	LS	Proposed
$G(0, 0.002)$	175	147	61	Level 1	175	147	61
$G(0, 0.006)$	193	290	63	Level 2	147	236	58
$G(0, 0.010)$	205	189	59	Level 3	136	221	59
$G(0, 0.014)$	185	368	64	Level 4	137	288	60

**Table 4.3:** Total number of lines extracted from a real image (the one in Fig. 4.7), under different transformations. On the left: fixed illumination conditions and different levels of Gaussian noise  $G(\mu, \sigma^2)$ . On the right: different illumination conditions and same noise level.

Noise	AG	LS	Proposed	Illumination	AG	LS	Proposed
$G(0, 0.002)$	975	2632	292	Level 1	975	2632	292
$G(0, 0.006)$	813	2161	233	Level 2	821	2041	278
$G(0, 0.010)$	746	2410	210	Level 3	778	2015	283
$G(0, 0.014)$	673	2471	206	Level 4	714	1985	283



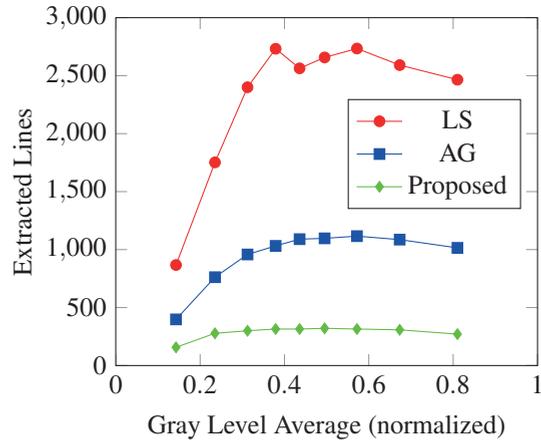
**Figure 4.11:** Images used for evaluation: The first row is comprised by synthetic images and the rest of images are grouped in pairs, taken from real industrial environments.



**Figure 4.12:** Images from public databases used for evaluation: the first four image pairs under various transformations are taken from [30]; the remaining two image sequences are taken from ZuBuD database [122].

sensitive to these conditions. In contrast, our approach gives practically identical results for all images, exhibiting robustness to noise level and illumination changes.

The performance of the line detectors for a real image in addition of Gaussian noise under two different illumination conditions is shown in Fig. 4.7. As seen, the overestimation in the number of lines performed by AG and LS becomes more evident on real images. While our approach extracts about 285 lines, AG reaches more than 750, and LS more than 2000, which is a very high number of lines, considering that the image resolution is not very large, but only  $1024 \times 768$  pixels. Closely observing the images one can see the large number of overlapping and fragmented lines extracted by both gradient-based methods, and also the large number of detection failures performed by LS, where many lines are extracted where none exists. In contrast, our approach extracted much less fragmented lines and no overlapping segments, achieving a much clearer detection with less lines extracted. It is important to note that a large number of overestimated lines will increase the computational cost of any post-processing, which may be impracticable for very high resolution images in which the number of perceived lines can be huge. In Fig. 4.8, Fig. 4.9 and Fig. 4.10 additional comparisons has been performed for the interest of this thesis. These comparatives has been performed against LS because this line detection method is used as base of a line matching method subject to quantitative comparison in the following chapter. In this way, the comparisons has been performed for the same datasets that will be employed for matching comparisons in the following chapter. In general, the number of lines and their plot highlight the high redundancy and fragmentation of the segments in the results by LS. For instance, Fig. 4.10 features very low number of perceived lines, but still LS marks 193 segments, while SALM shows a more reasonable number of 56, while both method are showing virtually the same human perceivable lines. Also Fig. 4.8 and Fig. 4.9 are examples of how the lines extracted by *SALM* present more accuracy in the direction and location of the actual human perceived line. In The nature of LS, based on gradient, is the responsible of its lower processing cost than *SALM*.



**Figure 4.13:** The variation of the total number of lines extracted by the methods for the same real image with different average gray levels.

It can be also observed in Fig. 4.7 the large variability in the number of lines extracted by AG and LS under the two different illumination conditions for the same image. There are  $\sim 200$  lines of difference between the results given by AG, and this number becomes in more than 600 lines for LS, whereas our approach gives only 9 lines of difference, providing almost the same detection for the same image under different illumination conditions. Table 4.3 gives a more complete information about the dependence of the methods on illumination conditions and noise levels. As seen, AG results vary in more than 300 lines along the different levels of Gaussian noise, and this difference is greater than 250 lines along the different illumination conditions. For its part, LS varies in more than 450 lines across noise levels and in more than 650 for the different illumination conditions. Meanwhile, our approach is practically invariant to illumination conditions (the biggest difference is only 14 lines), and also exhibits robustness to noise level with a maximum difference of 86 lines.

In Fig. 4.13 it can be seen the total number of lines extracted for the real image with different average gray levels. This figure shows very clearly the huge difference in the number of lines extracted by the methods for a real image, and also shows that even global brightness variations in the image cause significant variations in the total number of lines extracted by AG, and specially by LS, while our results remains quite stable.



## Chapter 5

# Straight line segment matching

Matching creates relations between elements of an image or several images. It is a basic tool in Computer Vision and the root for many applications in manufacturing industry, robotics or autonomous vehicles.

A basic problem in matching of primitives takes two different images showing the same scene, common elements or the same object. Both images may be different in image filtering, illumination, contrast, pose of 2D shapes or objects, different viewpoint or coloring. Additionally, the algorithm has to be provided with a set of detected primitives of the same kind in the images, which may be obtained from a detection algorithm on the images, other physical perception methods, or provided by humans. The challenge of a line matching method is to relate every primitive in one image to its counterpart in the other view.

Matching between edges on a pixel-by-pixel basis works in stereo systems with known epipolar camera geometry. However for the problem of freely moving cameras it is not possible to obtain the exact correspondence for each pixel between images without knowing the extrinsic and intrinsic parameters of the cameras. Representing curvy edges as a set of straight line fragments might not be a good approach for trying to match them. These ample

arcs can be detected or decomposed as discrete straight lines that the matching algorithm can handle. Nevertheless, the segment matching might not work properly in this case, because curvy edges and textured edges are expected to be decomposed differently for each different capture of the same object or scene.

The matching method we are proposing is referred to as *SALM*, which is the acronym of *Structure and Appearance Line Matching*. It integrates an outliers removal extension that employs 3D projections to group lines according to their coplanarity. The remainder of the chapter is organized as followed: Section 5.1 goes through a comprehensive short overview of the method. A detailed description is followed in Section 5.2, and the process of going from the images to the final straight line segments is mathematically explained. Section 5.3 expose the experimental set-up, the quantitative results, and compare them with several other state of the art line matching methods.

## 5.1 High level overview

A line matching method comprises a set of algorithms which put in correspondence segments across different images showing common elements, environment or regions of interest. A 3D reconstruction is the result of an estimation of the position of singular primitives captured in several images. The approach followed by *SALM* is framed in the group of line matching methods aimed for 3D reconstruction from pictures of objects built by humans, buildings, urban structures, industrial elements or computer generated models.

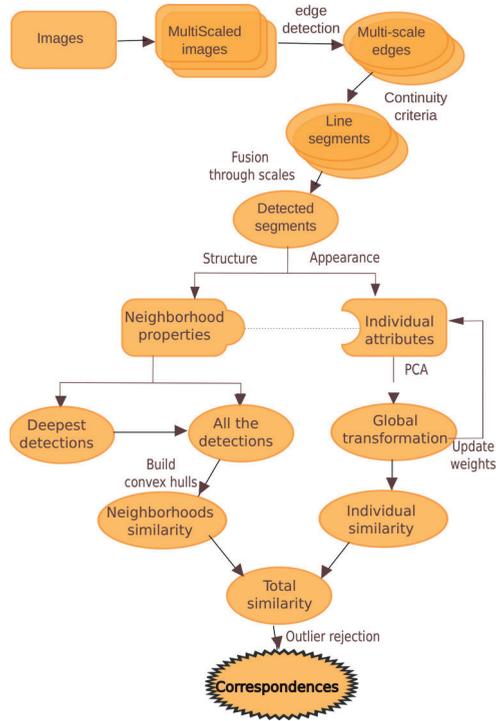
The *SALM* line matching algorithm is aimed for application altogether with 3D line based abstraction. It employs an iterative voting algorithm running in groups of lines with the same structural distribution[84], and the outliers rejection algorithm exploits 3D structures to discriminate potential outliers. *SALM*'s approach is based on three core elements: The first one is that a segment matching method will profit of detecting the segments based on edges obtained from the maxima of phase congruency in the scale-space, specially in sets of

images with variable illumination conditions. Secondly, a blend of descriptions of individual line appearance and the structure of groups of neighboring segments is the best approach for finding counterparts of the straight segments detected on different images. The last core element introduces an outliers detection algorithm based on coplanar line intersections of the lines put in correspondence. The inputs for the segment matching method are both the images and the intrinsic parameters of the camera, being the output the relation of matched lines among the images and the potential matching outliers.

## 5.2 Low level description

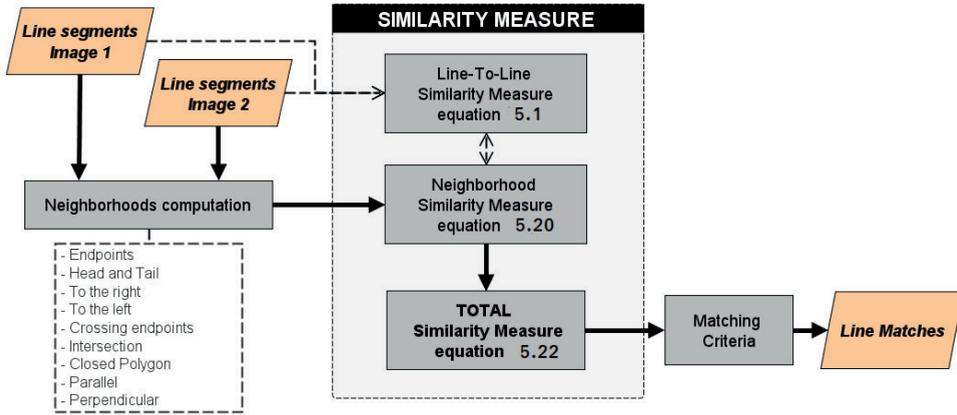
Fig. 5.1 shows a diagram of blocks with the main steps of the process from the images.

A two-stage iterative algorithm is designed for line matching, and the set of matching lines increases at each iteration. Both stages are executed sequentially, and both ones share the same iterative design. The inputs to the first stage are those lines that have been robustly detected along scale-space on both images, whereas the input of the second stage consists of all unmatched lines. These line sets are introduced in an iterative process for matching, which is performed in four steps. First, several kinds of line neighborhood are computed for each line to obtain local structure information. Secondly, a similarity measure is computed for each pair of line segments by taking into account their local structure, their geometric properties (such as orientation, length, location of endpoints) and also their local appearance (by using intensity correlations, and phase congruency averages). Thirdly, the strongest correspondences are added to the set of matched lines on the basis of a matching criteria. Finally, the weakest correspondences are broken by checking several conditions over the whole set of matched lines, ensuring that it grows robustly. The loop ends when all the matching lines remain paired with the same partner for at least  $T$  iterations. Results are obtained with  $T = 5$  which is found to be a balanced choice between the stability of line matches and time consumption.



**Figure 5.1:** Diagram of blocks for the SALM method. The upper region represents the integrated line detection method disclosed in Chapter 2.

The similarity between two lines relies on comparing their geometric properties and local appearance, but also the structural context where they are framed by taking into account the geometrical arrangement of their neighboring segments. Thus, two similarity measures are needed: a line-to-line measure to compare two lines purely, and a neighborhood measure to compare each of its neighborhoods. Figure 5.2 shows an overview of the steps involved in calculating the similarity between two lines.



**Figure 5.2:** Computation of the similarity measure for each pair of lines. The total similarity between two lines (equation 5.22) depends on the structural similarity of their neighborhoods (equation 5.20), which integrates the line-to-line similarity (equation 5.1) between the neighboring segments.

### Computation of individual similarity.

Given an unordered set of detected lines, related to one of the images:  $\mathcal{L}=\{l_1, l_2, l_3, \dots, l_A\}$ , the measure of similarity of  $l_i$  to an hypothetical counterpart on the other image  $l_j$  is stored into a similarity vector  $\mathbf{d}$ , comprising the similarity attributes both lines have in common. These components are:

### Line-To-Line Similarity Measure

The similarity  $z(i, j)$  between two lines  $i$  and  $j$  is measured based on geometric relationships and local appearance. Geometric relationships are constructed by using properties such as orientation, length, and location of endpoints. Local appearance is estimated by comparing gray level intensity averages, phase congruency averages, and also intensity correlations in the local neighborhoods of the segments.

A feature vector  $\mathbf{d} = (d_{cr}, d_{cl}, d_{PC}, d_{\rho}, d_{\theta}, d_1, d_x, d_y)$  is built for each pair of segments  $(i, j)$ ,

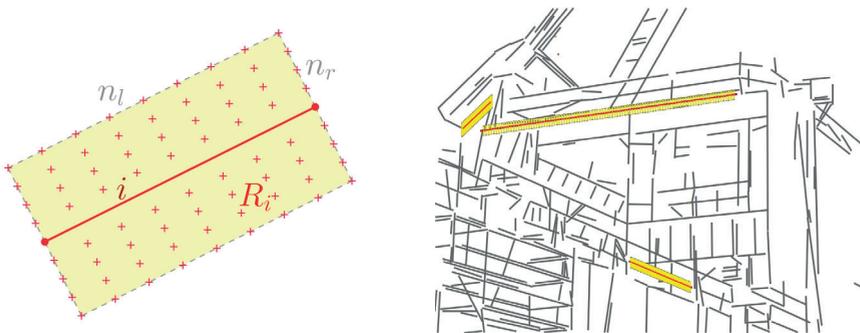
whose components are in  $[0, 1]$  according to their similarity, where all of them vanish for a segment with itself. The feature vector components are:

1. **Contrast:** average gray level intensity of the image.  $d_{cr}(i, j) = (I_i^r - I_j^r) / \max(I_i^r, I_j^r)$ , where  $I^r$  is the ratio of the average gray level intensity on the right side of the segment to the average gray level intensity of the image. The component  $d_{cl}$  is defined analogously on the left side.
2. **Phase Congruency,**  $d_{PC}(i, j) = \frac{|PC_i - PC_j|}{\max(PC_i, PC_j)}$ , where  $PC_i$  and  $PC_j$  are the phase congruency averages of the pixels of the segments.
3. **Intensity correlation:**  $d_\rho(i, j) = 1 - \rho^2(R_i, R_j)$ . For each segment, a local region  $R$  is defined, normalized by an invariant resampling under rotation and length (Fig. 5.3).  $\rho(R_i, R_j)$  is the correlation coefficient among gray level intensities of the two segments regions. This measure is invariant to global illumination changes.
4. **Angular distance,**  $d_\theta(i, j) = \frac{2}{\pi} \cdot |(\theta_i - \theta_j)|$  where  $\theta$  is the segment orientation,  $\theta \in [-\pi/2, \pi/2]$ .
5. **Difference in length:**  $d_l(i, j) = \frac{|l_i - l_j|}{\max(l_i, l_j)}$ , where  $l$  is the segment length.
6. **Distance between midpoints:**  $d_x(i, j) = |m_x^i - m_x^j| / \max(m_x^i, m_x^j)$ , where  $m_x^i, m_x^j$  are the  $x$  components of the midpoints of each segment, analogously for  $d_y(i, j)$ .

The line-to-line measure is computed as:

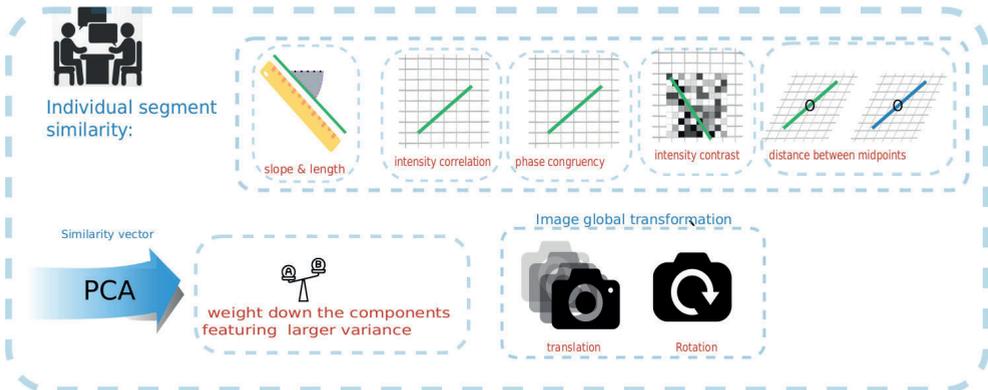
$$z(i, j) = \sum_{k=1}^8 w_k \cdot d_k(i, j) \quad (5.1)$$

where  $d_k$  are the components of the feature vector, and  $w_k$  are a set of normalized weights to take into account the relative importance of each component for matching. These weights are set at the beginning of the algorithm for each pair of input images by using a Principal Component Analysis (PCA), as followed in the next paragraph. The reason for introducing the



**Figure 5.3:** Local region  $R_i$  defined for a line segment  $i$ , consisting of  $n_l \times n_r$  samples uniformly distributed along the segment, with  $n_r = 61$  and  $n_l = 100$ . Examples of these regions for three different lines on a real image.

weights is to adapt the algorithm to each different image pair, according to the discriminative power of each component of the feature vector. The attributes for the individual similarity of lines are graphically represented in the upper half of the Fig. 5.4.



**Figure 5.4:** Computation of the individual similarity for the line segments, in the SALM method.

### Initial estimation of the weights of the feature vector.

The statistical procedure PCA creates an orthogonal linear transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables (principal components), in such a way that the first principal component has the largest possible variance, and each subsequent component in turn has the highest possible variance, fulfilling orthogonality to the previous components[105]. This analysis is used for the computation of lines' individual similarity to obtain an estimation of the weights of the feature vector.

In order to set the initial weights  $w_k$  of the feature vector, the line-to-line similarity measure for each possible line pairing is calculated at the beginning of the algorithm, obtaining a data matrix  $\mathbf{X} \in \mathbb{R}^{P \times 8}$ :

$$\mathbf{X} = [d_{cl}(i_n, j_m) | d_{cr}(i_n, j_m) | d_{PC}(i_n, j_m) | \dots | d_y(i_n, j_m)] \quad (5.2)$$

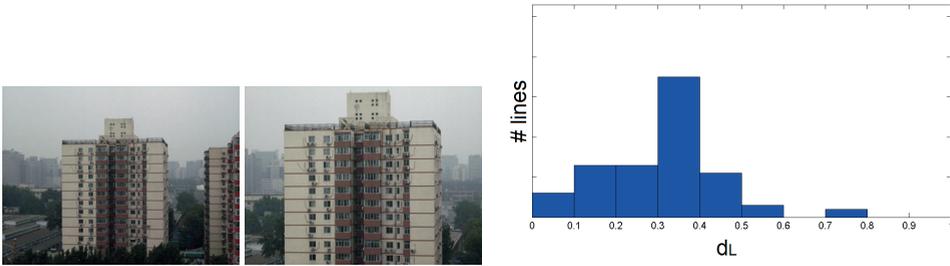
where  $i_n, j_m$  are all lines on both images ( $n = 1 \dots N, m = 1 \dots M$  and  $P = NM$ )

Then, PCA is performed over this data matrix, obtaining a linear transformation that let us write each element of the data matrix  $d_{(i)k}$  in the new coordinate system as a linear combination of the principal components  $p_{(i)t}$ .

$$d_{(i)k} = \sum_{t=1}^8 \alpha_{kt} \cdot p_{(i)t} \quad , k = 1 \dots 8 \quad (5.3)$$

where  $\alpha_{kj}$  are obtained directly from the analysis, and satisfy  $\sum_t \alpha_{kt}^2 = 1$ .

Since matchings are made between all possible line pairs, most of them will be wrong. So, ideally, all the components of the feature vector should return a value of  $d_k = 1$  because it implies the highest possible dissimilarity. Therefore, those components  $d_k$  that contribute less to the variability of the data will be the ones of most interest for the line matching since they are able to discern that almost all matches are wrong, getting almost all values close to



**Figure 5.5:** Histogram of the component  $d_l$  of the similarity vector, showing the change of length of the matched lines for an image pair. The dominant value in the histogram corresponds to the dominant scale transformation between the two images.

1. The weights  $w_k$  are assigned as follows:

$$w_k = \frac{1}{\sum_{t=1}^8 \alpha_{kt}^2 \cdot S_t}, k = 1 \dots 8 \quad (5.4)$$

where  $S_t$  is the total percentage of variance associated with the principal component  $p_t$ , which is obtained directly from the analysis. Thus, for example, the vector components whose projection on the  $p_1$ -direction (the direction of maximum variance) is large will be assigned to a small weight, because they are responsible for a large variance in the data, and are not distinguishing that almost all matches are wrong.

### Estimating a global transformation between images.

The line-to-line similarity of the matched lines at each iteration is used to estimate whether the images are related by a global transformation: global illumination changes, rotations, scalings and displacements. Thus, the algorithm is able to infer information about the image transformation, which in turn is used to correct the line-to-line similarity measure in the next iterations.

The histograms of the components  $\{d_{cr}, d_{cl}, d_{\theta}, d_l, d_x, d_y\}$  are computed for the matched lines at each iteration, and the dominant values (those exceeding 33%) are associated with the existence of a global transformation. Fig 5.5 gives an example of a length histogram of

an image pair obtained from the matched lines at a fixed iteration, in which it can be seen that the dominant value is related to the global scale transformation. These dominant values are then used to shift the feature components in the next iterations, in such a way that if there are some dominant values of the component  $d_k$ , then it is recomputed as the minimum of the difference between dominant values and old value. The idea was inspired by [158], where line direction histograms were used to deal with global rotations.

A similarity vector  $\mathbf{d}$  is computed for every possible pairwise combinations, and embedded into the matrix  $\mathbf{X}^{P \times 8}$ :

$$\mathbf{X} = \begin{bmatrix} d_{cl}(i_1, j_1) & d_{cr}(i_1, j_1) & d_{pc}(i_1, j_1) & \dots & d_y(i_1, j_1) \\ \dots & \dots & \dots & \dots & \dots \\ d_{cl}(i_n, j_m) & d_{cr}(i_n, j_m) & d_{pc}(i_n, j_m) & \dots & d_y(i_n, j_m) \\ \dots & \dots & \dots & \dots & \dots \\ d_{cl}(i_A, j_B) & d_{cr}(i_A, j_B) & d_{pc}(i_A, j_B) & \dots & d_y(i_A, j_B) \end{bmatrix}, \quad (5.5)$$

where  $A$  and  $B$  are the numbers of detected segments in the first and second images respectively, therefore the pair of indexes  $(i_n, j_m)$  represents all possible pairwise combinations of segment between views ( $n = 1 \dots A$ ,  $m = 1 \dots B$ ).

A PCA is applied over the rows of the matrix  $\mathbf{X}$ , obtaining a linear transformation that lets us write each element of the data matrix  $d_k(i_n, j_m)$  in the new coordinate system as a linear combination of the principal components  $pc^t$ :

$$d_k(i_n, j_m) = \sum_{t=1}^8 \alpha_k^t(i_n, j_m) \cdot pc_k^t(i_n, j_m) \quad , k = 1 \dots 8 \quad , \quad (5.6)$$

where  $\alpha_k^t(i_n, j_m)$  is the set of  $k$  variables that are obtained directly from the analysis, and satisfy  $\sum_{t=1}^8 \alpha_k^t(i_n, j_m)^2 = 1$ . This procedure discriminates the hypothetical correspondences with the lowest dissimilarity. The weight applied to each component of a similarity vector is intended to give more relevance if it has low variability, and to damp the ones with higher dissimilarity through the whole set of hypotheses. In addition, this set of similarity vectors

is drawing the most probable global transformation between both images. Since the correspondences are made between all possible line pairs, most of the hypotheses will be wrong, meaning that the majority of the components should have a value close to 1, i.e. the highest dissimilarity. Those  $d_k$  that contribute less to the variability of the data are the components that allow to discern the hypothetical correct matches, consequently weights  $w_k$  are assigned as follows:

$$w_k = \frac{1}{\sum_{t=1}^8 \alpha_t^2 \cdot S_t}, k = 1 \dots 8, \quad (5.7)$$

where  $S_t$  is the total percentage of variance associated with  $pc_t^1$ . Thus, vector components whose projection on the direction  $\mathbf{pc}^1$  are large, will be assigned to small weights. This is done because this direction features a large variance in the data, and does not allow to distinguishing the most probable matches. The appearance similarity measure is therefore computed as:

$$z(i, j) = \sum_{k=1}^8 w_k \cdot d_k(i, j), \quad (5.8)$$

Distances between geometric and appearance features of matched lines are used to estimate whether the images are related by a global transformation in illumination or affine geometry [159]. A tenth of the range of values of a similarity component is considered to resemble a global transformation when more than 33% of the total number of hypotheses falls into it.

A group of neighboring line segments can be enclosed into a convex hull polygon[152], the smallest convex polygon containing all the vertices of the neighbors and the line segment itself. By making use of the area invariance property associated with the affine transformation, a set of convex hull affine invariants are constructed by taking ratios between the areas of the polygons formed by connecting consecutive vertices on the convex hull. For a line  $l_i$  with a neighborhood  $N^{(i)} = \{l_1^i, l_2^i, \dots, l_R^i\}$ , the set of affine invariants  $\Omega^{N^{(i)}} = \{\Omega_1^{N^{(i)}}, \dots, \Omega_R^{N^{(i)}}\}$ , stores the ratios of the areas of every possible triangle in the quadrangle to the area of the quadrangle itself. In order to establish the correspondence between two hypothetically match-

ing neighborhoods  $N^{(i)}$  and  $N^{(j)}$ , the following measure is used:

$$c(N^{(i)}, N^{(j)}) = \min_{e=1, \dots, R} \min_{f=1, \dots, S} \|\Omega_e^{N^{(i)}} - \Omega_f^{N^{(j)}}\|^2, \quad (5.9)$$

where  $\Omega^{N^{(i)}}$  and  $\Omega^{N^{(j)}}$  are the set of the affine invariants of the two neighborhoods,  $n$  and  $m$  are the number of vertices of their convex hulls respectively. For a neighborhood  $N^{(i)}$ , the quantity  $c(N^{(i)}, N^{(j)})$  gets minimum for the pair of neighborhoods with the highest probability of resembling affine-regular polygons. A maximum threshold  $c_{TH} = 0.02$  is set to decide whether the transformation is affine or not, and discard hypotheses.

The final structural measure  $Z(N^{(i)}, N^{(j)})$  for two neighborhoods  $N^{(i)} = \{i_1, \dots, i_R\}$  and  $N^{(j)} = \{j_1, \dots, j_S\}$  adds to the equation the appearance similarity, and is computed as:

$$Z(N^{(i)}, N^{(j)}) = \begin{cases} 1 & , c(N^{(i)}, N^{(j)}) > c_{TH} \\ \frac{1}{(A+1)\tau} \sqrt{z^2(i, j) + \min_{\sigma \in S_N} \sum_{k=1}^N z^2(i_k, j_k)} & , \text{otherwise} \end{cases}, \quad (5.10)$$

where  $S_N$  is the symmetric group including all permutations among neighboring segments, aimed to achieve invariance against the order they were taken.  $\tau$  is the number of elapsed iterations when all the neighborhoods  $N^{(i)}$  and  $N^{(j)}$  are matched, or  $\tau = 1$  otherwise, and  $z'(\cdot, \cdot)$  is a refinement of  $z(\cdot, \cdot)$  in Eq. 5.8, defined as:

$$z'(i, j) = \begin{cases} 0 & , (i, j) \text{ match with each other} \\ 1 & , i \text{ matches } k \neq j, \text{ or vice versa} \\ z(i, j) & , \text{otherwise} \end{cases}, \quad (5.11)$$

$z'(i, j)$  favors a matching of two lines if their neighbors are already matched.

The scores obtained for both the appearance and geometric properties between two hypothetical matching segments need to be merged into a single final distance measure. This measure is noted  $D(i, j) \in [0, 1]$  and given as:

$$D(i, j) = \frac{\sum_{k=1}^9 W_k \cdot Z(N_k^{(i)}, N_k^{(j)})}{\sum_{k=1}^9 W_k}, \quad (5.12)$$

where  $N_k^{(i)}$  and  $N_k^{(j)}$  are the neighborhoods of the matching lines and  $W_k$  a set of weights. These weights are initiated to 1 and are updated at each iteration by using PCA, analogously as described in Eq. 5.7 for the appearance similarity weights  $w_k$ , but in this case the data matrix is obtained from the structural similarity  $Z(N_k^{(i)}, N_k^{(j)})$  for each neighborhood of the matched lines. Distances get close to 0 for most of the neighborhoods, and therefore we should assign a higher weight to those neighborhoods that have so far contributed less to the variability of the data, since they are able to discern that almost all matches are correct. Consequently, weights  $W_k$  are updated as follows:

$$W_k = \frac{1}{1 - \sum_{t=1}^9 \alpha_k^t \cdot S_t} \quad , k = 1 \dots 9 \quad , \quad (5.13)$$

where  $\alpha_k^t$  are a set of coefficients obtained directly from the analysis, and  $S_t \in [0, 1]$  resembles the total percentage of variance associated with each component. Therefore the weights are dynamically updated during the execution of the algorithm.

A pair of segments  $\{l_i, l_j\}$  is matched based on a nearest/next distance ratio criteria. Let's consider a line  $l_i$  with the highest similarity to hypothetical counterpart  $l_j$ , and followed by another candidate  $l_q$ . On the other hand,  $l_j$  has the highest similarity to  $l_i$ , and is followed by  $l_p$ :

$$D(l_p, l_p j) / D(l_p i, l_p j) > \delta \quad (5.14)$$

$$D(l_p i, l_p q) / D(l_p i, l_p j) > \delta \quad (5.15)$$

$$D(l_p i, l_p j) > D_{th} \quad , \quad (5.16)$$

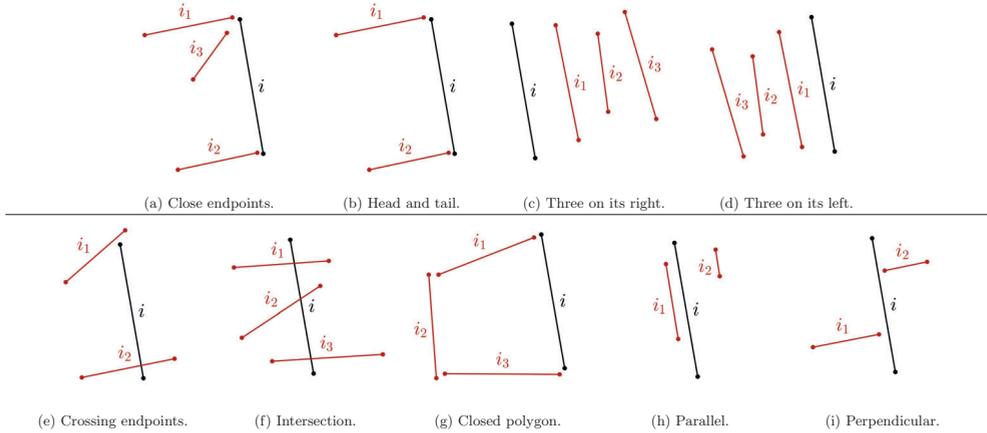
being  $\delta$  the distance ratio threshold, and  $D_{th}$  a global distance threshold.  $\delta = 1.5$  and  $D_{th} = 1$  for the first stage of the algorithm, that takes just lines that are robustly detected in the scale-space and ending after collecting the best  $N = 10$  line correspondences. For the second stage

of the algorithm, that takes all the detected lines on both images,  $\delta = 1.2$  and  $D_{th} = 1$ . This second stage runs until all the lines with candidate counterparts are paired. The weighting of the components of the feature vector for the computation of the individual similarity is graphically represented in the lower half of Fig. 5.4.

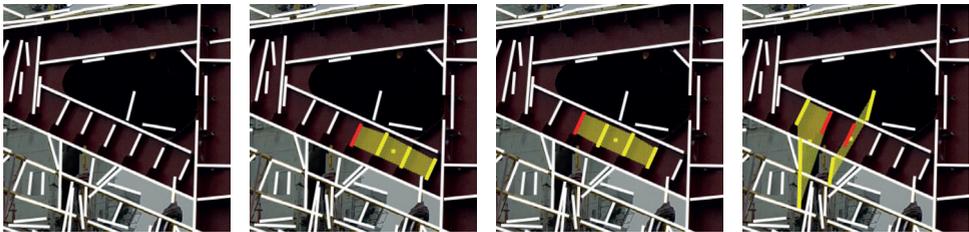
### Neighborhood computation.

Line neighborhoods provide structural information around each line in the images. For each line  $i$ , several neighborhoods  $L_k^{(i)}, k = 1 \dots 9$ , are computed. The usage of several kinds of line neighborhoods has the advantage of making the method more robust to occlusions and differences in perspective, fragmentation of lines and misdetections between views. The reliance on a single type would cause the method to fail in many situations where the selected neighbors of a line and its homologous does not match for that type of neighborhood, and also in situations where multiple lines have an identical local structure (repetitive patterns) for which more structural information is required for disambiguation. Fig. 5.7 illustrates these difficulties and Fig. 5.6 shows a representation of each one of the neighborhoods, which are described below:

1. **Endpoints:** This neighborhood is formed by the three closest segments to  $i$  based on the minimum distance between endpoints. The neighbors must be closer than a threshold  $D_{th} = 14$ .
2. **Head and tail:** Formed by the closest segment to one of the endpoints of the line  $i$ , and another segment which is the closest to the other endpoint of  $i$ . The neighbors must be closer than  $D_{th}$ .
3. **Three on its right:** Formed by three segments  $i_1, i_2$  and  $i_3$ . The segment  $i_1$  is the closest to  $i$  that has their endpoints at the right of  $i$ , fulfilling  $d_\theta(i, i_1) = \frac{2}{\pi} \cdot |(\theta_i - \theta_{i_1})| < \frac{1}{4}$  and  $d_l(i, i_1) = \frac{|l_i - l_{i_1}|}{\max(l_i, l_{i_1})} < \frac{1}{2}$ , where  $\theta$  is the segment orientation, and  $l$  is the segment length.



**Figure 5.6:** Classification of the different kinds of neighborhoods for a line  $i$  attending to the relative pose of its neighboring lines  $i_j$ .



**Figure 5.7:** Example of a real situation which illustrates the need of several line neighborhoods. The images show three simultaneously coexistent neighborhoods. The image on the left shows the original region of the low-textured scene, in which the six central lines are very similar. The middle left and middle right images show line neighborhoods of the type 'Three on its right' for two different lines. Both neighborhoods are too similar to be discerned by the algorithm when matching to a different view. Therefore, more structural information has to be provided in order to distinguish the counterparts of both segments in other image. In the right hand image, another kind of neighborhood ('Head and Tail') is represented for both lines. Note that this latter neighborhood solves the disambiguation, meaning that the votes of the segments in these neighborhoods will produce different scores that will be used to distinguish both segments.

The segment  $i_2$  is chosen in the same way with respect to  $i_1$ , and equivalently for  $i_3$  with respect to  $i_2$ .

4. **Three on its left:** Analogously, but to the left.
5. **Crossing endpoints:** Formed by the two closest segments to  $i$  based on the distance between segment and midpoint. Those segments that have one endpoint to the right and the other endpoint to the left of  $i$  are preferred.
6. **Intersection:** This neighborhood is formed by the three segments the closest to  $i$  based on the euclidean distance between midpoints, under the constraint that both intersect  $i$ .
7. **Closed polygon:** Formed by a chain of segments,  $\{i_1, i_2, \dots, i_N\}$ , where the segment  $i_1$  is the closest to the head endpoint of  $i$ , the segment  $i_2$  is the closest to the endpoint of  $i_1$  which is the farthest to  $i$ , the segment  $i_3$  is the closest to the endpoint of  $i_2$  which is the farthest to  $i_1$ , and so on. The neighborhood is only considered if the tail endpoint of  $i$  is closest to the endpoint of some  $i_N$ , closing the chain. It is only allowed  $N < 4$ , and the neighborhood must be the same when exchanging the 'head' and 'tail' conventions of  $i$ .
8. **Parallel:** The two closest segments are parallel to  $i$ :  $d_\theta(i, j) < \frac{1}{4}$ .
9. **Perpendicular:** The two closest segments are perpendicular to  $i$ :  $d_\theta(i, j) > \frac{3}{4}$ .

These line neighborhoods have been chosen to provide enough structural information around each line, trying not to add too many to avoid slowing the process.

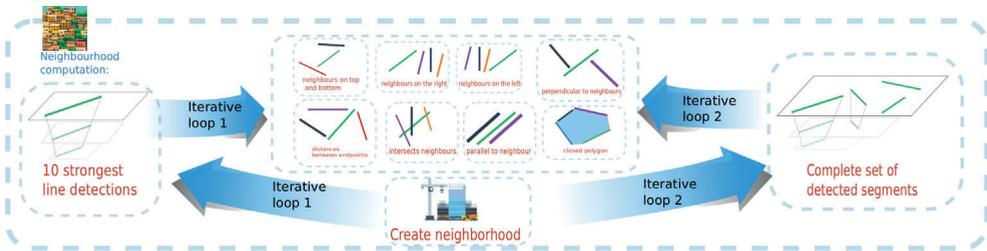


Figure 5.8: Process of neighborhood computation for the SALM method.

**Neighborhood Similarity Measure.**

The neighborhood similarity measure estimates the likelihood that two line neighborhoods are the same under an affine transformation. The measure uses the previous line-to-line measure to compare the neighboring segments of two line neighborhoods, as well as a set of affine invariants for taking into account the geometrical arrangement of the segments.

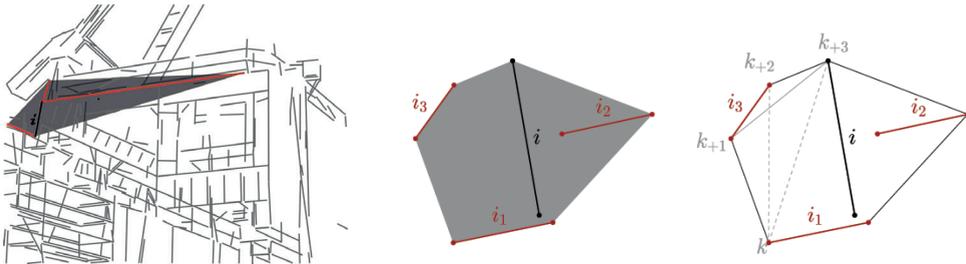


Figure 5.9: Image on the left: example of the convex hull (shaded region) from a line neighborhood for a real image. In the center: representation of the convex hull of the line neighborhood  $L = \{i_1, i_2, i_3\}$  of the line  $i$ . Image on the right: construction example of an affine invariant vector  $\Omega_k^B$ , which is obtained by taking the ratio of the areas of the four triangles  $\{k, k_{+1}, k_{+2}\}$ ,  $\{k_{+1}, k_{+2}, k_{+3}\}$ ,  $\{k_{+2}, k_{+3}, k\}$ ,  $\{k_{+3}, k, k_{+1}\}$  to the area of the quadrangle  $\{k, k_{+1}, k_{+2}, k_{+3}\}$ .

To obtain the set of affine invariants, the convex hull of each line neighborhood is constructed as the smallest convex polygon containing all the vertices of the neighboring segments as well as of the line itself. Then, by making use of the area invariance property associated with the affine transformation, a set of convex hull affine invariants are constructed by taking ratios between the areas of the polygons formed by connecting consecutive vertices on the convex hull, as explained in [153]. In such a way that, for a line  $i$  with a neighborhood  $L = \{i_1, i_2, \dots, i_N\}$ , the set of affine invariants is  $\Omega^L = \{\Omega_1^L, \Omega_2^L, \dots, \Omega_n^L\}$ , with  $n$  being the number of vertices of the convex hull, and where:

$$\Omega_k^B = \left\{ \frac{S_1(k)}{S(k)}, \frac{S_2(k)}{S(k)}, \frac{S_3(k)}{S(k)}, \frac{S_4(k)}{S(k)} \right\}, k = 1 \dots n, \quad (5.17)$$

where  $S(k)$  is the area of the quadrangle  $k$  formed from the four consecutive vertices  $(k, k+1, k+2, k+3)$  of the convex hull, and  $S_1(k), S_2(k), S_3(k), S_4(k)$  are the areas of the 4 sub-triangles of the quadrangle  $k$ . Fig. 5.9 shows an example of the construction of an affine invariant vector.

Due to occlusions and overlappings, the number of vertices of the convex hull may vary for the same line neighborhood from one image to another, and consequently, the number of affine invariants may be different even for homologous neighboring segments. So, to establish the correspondence between two neighborhoods  $L$  and  $M$ , the following measure  $c(L, M)$  is used:

$$c(L, M) = \min_{j=1, \dots, n} \min_{k=1, \dots, m} c_{LM}(j, k, w), \quad (5.18)$$

with

$$\begin{aligned} c_{LM}(j, k, w) = & \|\Omega_j^L - \Omega_k^M\|^2 + \|\Omega_{j+1}^L - \Omega_{k+1}^M\|^2 \\ & + \dots + \|\Omega_{j+w-1}^L - \Omega_{k+w-1}^M\|^2, \end{aligned} \quad (5.19)$$

where  $\Omega^L$  and  $\Omega^M$  are the set of the affine invariants of the two neighborhoods,  $n$  and  $m$  are the number of vertices of its convex hulls respectively, and  $w$  is a parameter to balance between relying on local shape through the use of local invariants to establish correspondence versus

relying on more regional shape. Results are obtained with  $w = 1$ . Thus defined,  $c(L, M)$  vanishes for two neighborhoods  $L$  and  $M$  when their convex hulls are affine-regular polygons. In practice, an affine threshold  $c_{TH} = 0.02$  is set such that  $c(L, M) < c_{TH}$  implies that  $L$  and  $M$  are related by an affine transformation.

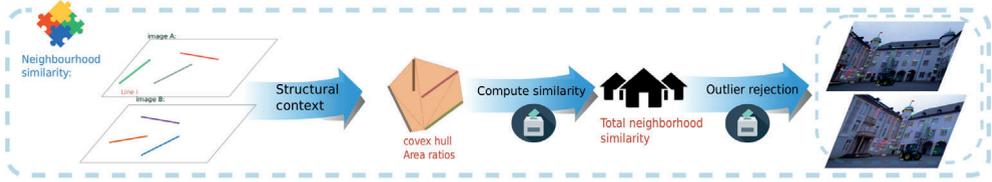
The final neighborhood measure  $Z(L, M)$  for two neighborhoods  $L = \{i_1, i_2, \dots, i_N\}$  and  $M = \{j_1, j_2, \dots, j_P\}$  of the lines  $i$  and  $j$  respectively, is computed as:

$$Z(L, M) = \begin{cases} 1 & , c(L, M) < c_{TH} \\ 1 & , N \neq P \\ \frac{1}{(N+1)\tau} \left( z^2(i, j) + \min_{\sigma \in S_N} \sum_{k=1}^N z'^2(i_k, j_{\sigma(k)}) \right)^{1/2} & , \text{otherwise,} \end{cases} \quad (5.20)$$

where  $z(\cdot, \cdot)$  is the line-to-line measure defined previously,  $S_N$  is the symmetric group for checking all permutations among neighboring segments to achieve invariance against the order they were taken,  $\tau$  is the number of elapsed iterations when all the neighbors of  $L$  and  $M$  are matched with each other, or  $\tau = 1$  otherwise, and  $z'(\cdot, \cdot)$  is a refinement of the line-to-line measure for taking into account the matching information from previous iterations of the algorithm, defined as:

$$z'(i, j) = \begin{cases} 0 & , (i, j) \text{ matched with each other} \\ 1 & , i \text{ matched with a line different from } j, \text{ or vice versa.} \\ z(i, j) & , \text{matched with no line.} \end{cases} \quad (5.21)$$

Therefore, by using  $z'(i, j)$ , the neighborhood similarity measure takes advantage of the matching information obtained in previous iterations, producing a better score for those pairs whose neighbors are already matched, and by using  $\tau$  this information becomes more reliable as the iteration progresses. The computation of the neighborhood similarity is graphically represented in the left hand side of Fig. 5.10.



**Figure 5.10:** Graphical representation for the computation of the similarity of neighborhoods, the integration with the individual similarity to obtain the total similarity, and last steps towards obtaining the final segment matchings.

### Total Similarity Measure:

The similarity measure between two lines  $D(i, j) \in [0, 1]$  is given as:

$$D(i, j) = \frac{\sum_{k=1}^9 W_k \cdot Z(B_k, M_k)}{\sum_{k=1}^9 W_k}, \quad (5.22)$$

where  $B_k$  and  $M_k$  are the neighborhoods of the lines  $i$  and  $j$  respectively, and  $W_k$  a set of weights. These weights are set to 1 at the beginning, and are updated at each iteration by using PCA, in a similar way to that described above (Eq. 5.7) for the weights  $w_k$ . In this case, the data matrix is obtained from the neighborhood similarity measure  $Z(B_k, M_k)$  for each neighborhood of the matched lines. The process followed is identical to the one already described, but here the way to assign the weights is different, since we are now using mostly matches that are correct (and not wrong ones as in the previous case). So, ideally, we should now get the highest similarity measure ( $Z(B_k, M_k) = 0$ ) for all neighborhoods, and therefore we should assign a higher weight to those neighborhoods that have so far contributed less to the variability of the data, since they are able to discern that almost all matches are correct, getting almost all values close to 0. The weights  $W_k$  are assigned as follows:

$$W_k = \frac{1}{1 - \sum_{t=1}^9 \alpha_{kt}^2 \cdot S_t}, \quad k = 1 \dots 9, \quad (5.23)$$

where  $\alpha_{kj}$  are a set of coefficients obtained directly from the analysis, and  $S_i \in [0, 1]$  is the total percentage of variance associated with each principal component. Thus, we assign a higher weight to those neighborhoods which are choosing consistent neighbors according to the current matching information, becoming more significant in the following iterations. In this way, the importance of each kind of neighborhood is dynamically weighted through the execution of the algorithm.

### Matching criteria.

Once the total similarity measure is computed, the most similar lines are added to the set of matched lines. A pair of segments  $(i, j)$  are matched based on a NNDR (nearest/next distance ratio) criteria, i.e.:

$$D(p, j)/D(i, j) > \lambda, \quad (5.24)$$

$$D(i, q)/D(i, j) > \lambda, \quad (5.25)$$

$$D(i, j) < D_{th}, \quad (5.26)$$

where  $p$  and  $q$  are the next nearest to  $j$  and  $i$  respectively,  $\lambda$  is the distance ratio and  $D_{th}$  a global threshold. Results are obtained with  $\lambda = 1.5$  and  $D_{th} = 1.0$  for the first stage of the algorithm, and  $\lambda = 1.2$  for the second one to achieve a greater number of matchings. Besides, for the first iteration, only the  $N = 10$  lines with the best distance ratio are matched, even without satisfying the thresholds, ensuring that the algorithm starts with an initial seed of a suitable size.

### Voting hypothetical counterparts.

At each iteration, the weakest correspondences are broken to ensure that the set of matched lines grows robustly throughout the iterative process. The weakest correspondences are selected by means of an affine-regular voting method, in which each pair of matched lines votes

to the other pairs. A matched pair  $(i, m_i)$  gives a positive vote ( $p = 1$ ) to the pair  $(j, m_j)$  when the line neighborhoods  $B = \{i, j\}$  and  $M = \{m_i, m_j\}$  are related by an affine transformation ( $c(B, M) < c_{TH}$ ), and gives a negative vote ( $p = 0$ ) otherwise. Once all pairs have voted, those matches with a low number of votes ( $\sum_{N-1} p_i < 0.4N$ ) are broken. Therefore, this strategy removes most dubious matches (although they can be correctly matched), under the assumption of a locally affine transformation. The voting mechanism for the final matchings in the method *SALM* is graphically represented in the middle region of Fig. 5.10.

A flow chart embedding both the line detection method and *SALM* is included in Fig. 5.11.

### Outliers detection

The outliers in the line matching process are lines that get matched to an incorrect counterpart, meaning that a human would not mark that line as being the same on both images. The assessment for counting matching outliers has to be performed by a human. Line matching outliers can occur either originated by segment misdetections on some images, line fragmentation, overlapping, or be triggered by a weak understanding of line neighborhood structures by the matching algorithm. The latter mentioned source of outliers can be overcome by a proper description of the 2D structure of groups of detected lines.

The *SALM* method includes two different approaches to detect incorrect matches. The first one is simple and is integrated into the matching algorithm, it is based on a voting method and runs at the end of each iteration, after computing the structural similarity. Each pair of matched segments votes other pairs according to their presence on their respective set of convex hulls: The hypothetically matched line that is voting  $\{l_i, l_j\}$  will give a positive point to  $\{l_p, l_q\}$  just if  $l_p$  and  $l_q$  are included in the convex hulls habited by  $l_i$  and  $l_j$  respectively. As a line can simultaneously be part of multiple convex hulls, several different structural relations are built within the neighborhood of each segment. Every neighbor segment votes other hypothetical correspondences located in their respective convex hulls. The correspondences

between segments that are voted positive by less than 40% of their neighbors, are considered as outliers and are rejected.

A more sophisticated outliers detection method runs over the final matches, to double-check the set of line correspondences by performing geometric relations among their mutual intersections. Matched segments are extended to intersect neighbors within the image boundaries. This intersection and its counterpart are stored if they are apart from the each segment a distance less than two times the length of the shortest originating segment, and the inner angle drawn by the intersecting lines is greater than  $\pi/6$ . These requirements are implemented because intersections will carry the uncertainty in the direction and location of both crossing lines. Firstly, the set of corresponding 2D intersections on both images  $\mathcal{I} = \{\mathcal{I}^i, \mathcal{I}^j\}$  is divided into groups according to their coplanarity:  $\mathcal{I}^i = \{\mathcal{I}_1^i, \mathcal{I}_2^i, \dots, \mathcal{I}_V^i\}$  being  $V$  the total number of planes with more than 10 3D points. A triangulation of these points is performed. Although for this problem the camera matrix  $\mathbf{K}$  is provided, the camera poses  $\mathcal{P} = \{\mathbf{P}^i, \mathbf{P}^j\}$  are unknown, therefore we have chosen to estimate them from the line crossing correspondences. The Essential Matrix  $\mathbf{E}$  is estimated by using the Five-Point Algorithm[102]. Having  $\mathbf{E} = \mathbf{R}[\mathbf{t}]_{\times}$  and the set of 3D points  $\mathcal{Y}$ , the relative camera rotation and translation among the first pair of cameras  $\mathbf{P}^j = \mathbf{K} \times [\mathbf{R}|\mathbf{t}]$  are estimated by using cheirality check and discarding the triangulated points of  $\mathcal{Y}$  that are not in front of the cameras.

The grouped line intersections are used to double-check groups of matched lines for consistency. A line crossing several intersections in different order than its counterpart is prone to be a matching outlier. The output of the algorithm is a set of line correspondences flagged as not trustworthy. This algorithm is depicted in Algorithm 1. At the start of this algorithm the triangulated 3D points are fit to different planes. RANSAC is employed for the generation of hypothetical groups of 3D points. A minimum threshold of 10 points is required to resemble a valid plane, and the 3D points that are not related to any plane after 100 iterations will be discarded for the rest of the algorithm. Therefore, each fitted plane gets related to a group of known corresponding intersections. Within each group, a search for neighboring coplanar

intersections is performed by a k-Nearest-Neighbors (kNN) algorithm, as shown in the main loop of Algorithm 1 and drawn on Fig. 5.12.

---

**Algorithm 1:** Line matching outlier detection
 

---

**Data:** Set of matched lines  $\{l_1^i, l_1^j\}, \{l_2^i, l_2^j\} \cdots \{l_L^i, l_L^j\}$  and their intersections on both images  $\{\mathcal{S}^i, \mathcal{S}^j\}$ ;  $\mathbf{K}$

**Result:** Most probable matching outliers  $\mathcal{L}$   
initialization;

Selection criteria for intersections  $\rightarrow \{\mathcal{S}^i, \mathcal{S}^j\}$

Linear Triangulation for  $\{\mathcal{S}^i, \mathcal{S}^j\} \rightarrow$  3D points  $\mathcal{Y}$ , camera poses  $\{\mathbf{P}^i, \mathbf{P}^j\}$

Fit  $\mathcal{Y}$  to  $V$  different planes;  $v \in V$

**for**  $(\{c_{AB}^i, c_{AB}^j\}) \in \{\mathcal{S}_v^i, \mathcal{S}_v^j\}$  **do**

    k-NN: Find 5 Nearest Neighbors of  $\{c_{AB}^i, c_{AB}^j\} \rightarrow \{d^i, d^j\}$

$d^i = \{d_1^i, d_2^i, d_3^i, d_4^i, d_5^i\} \in \mathcal{S}_v^i$

$d^j = \{d_1^j, d_2^j, d_3^j, d_4^j, d_5^j\} \in \mathcal{S}_v^j$

**for**  $u \in [1, 5]$  **do**

**if** Counterpart of  $d_u^i \in d^j$  **then**

$Score(\{c_{AB}^i, c_{AB}^j\}) = Score(\{c_{AB}^i, c_{AB}^j\}) + 1$

**end**

**end**

**if**  $Score(\{c_{AB}^i, c_{AB}^j\}) < 2$  **then**

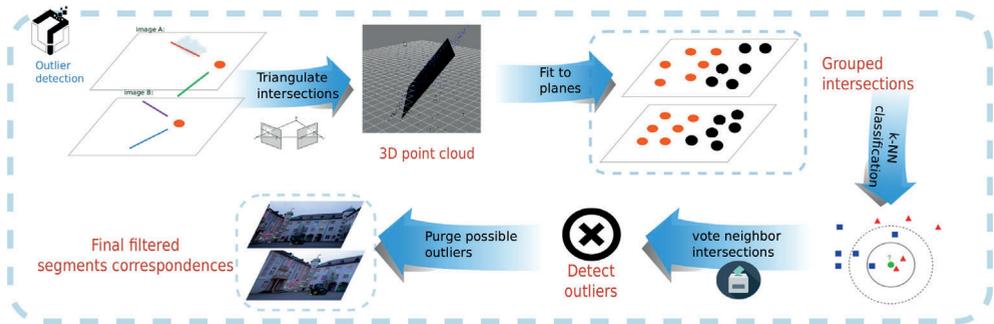
$\{l_A^i, l_A^j\}$  and  $\{l_B^i, l_B^j\}$  are potential outliers and stored in  $\mathcal{L}$ .

**end**

**end**

---

The relative 2D position of the intersections within these groups are compared on both images, to unveil a subset of  $\mathcal{Y}$  comprised by the intersections that are most likely to be outliers. Any matched segment that originates four or more suspicious intersections is quarantined, as written in the last condition of the Algorithm 1. A graphical representation for the outlier rejection algorithm depicted in this subsection is drawn in Fig. 5.13.



**Figure 5.13:** Graphical representation for the algorithm that exploits planes to obtain possible line matching outliers. This algorithm can be run after a line matching process.

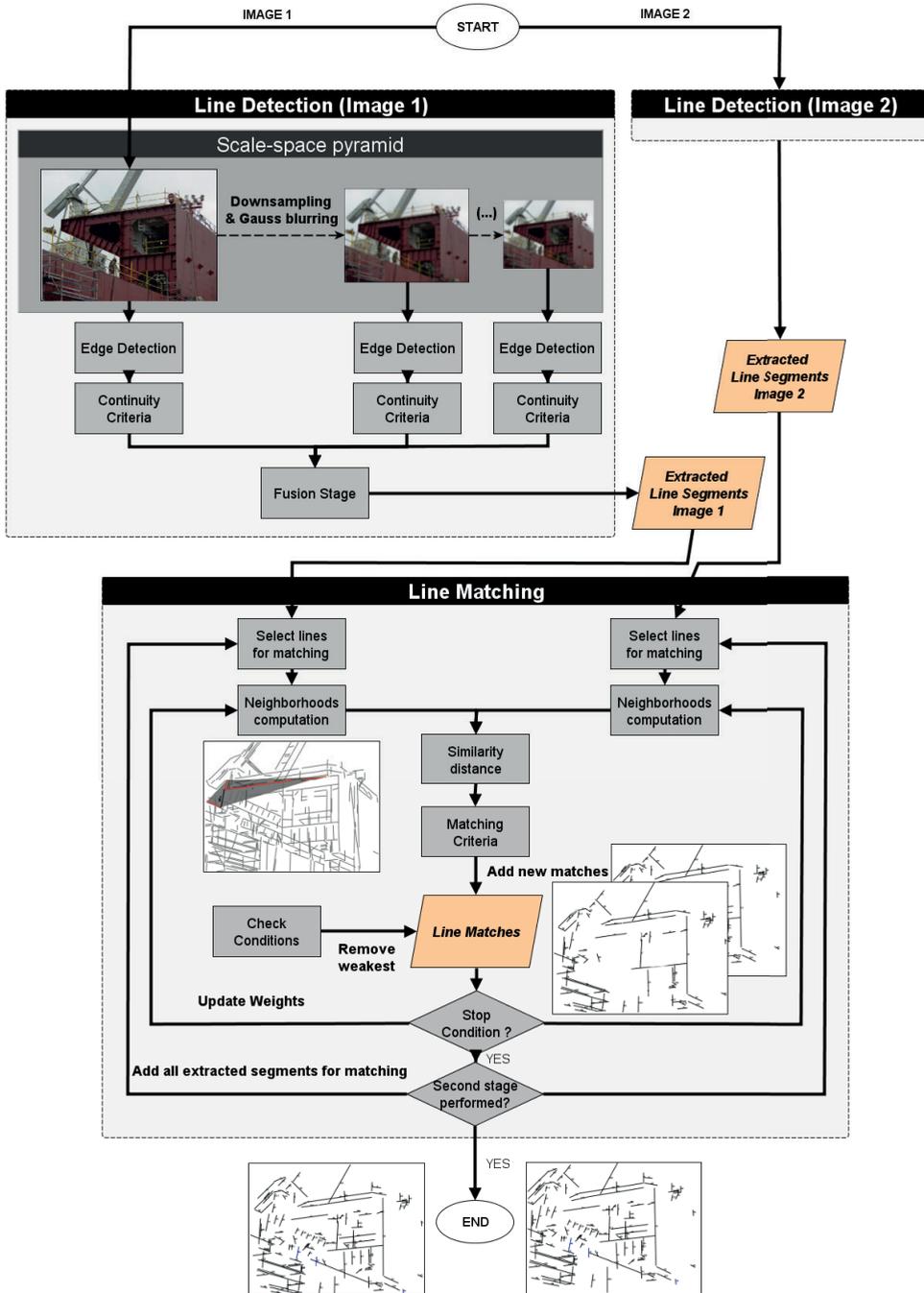
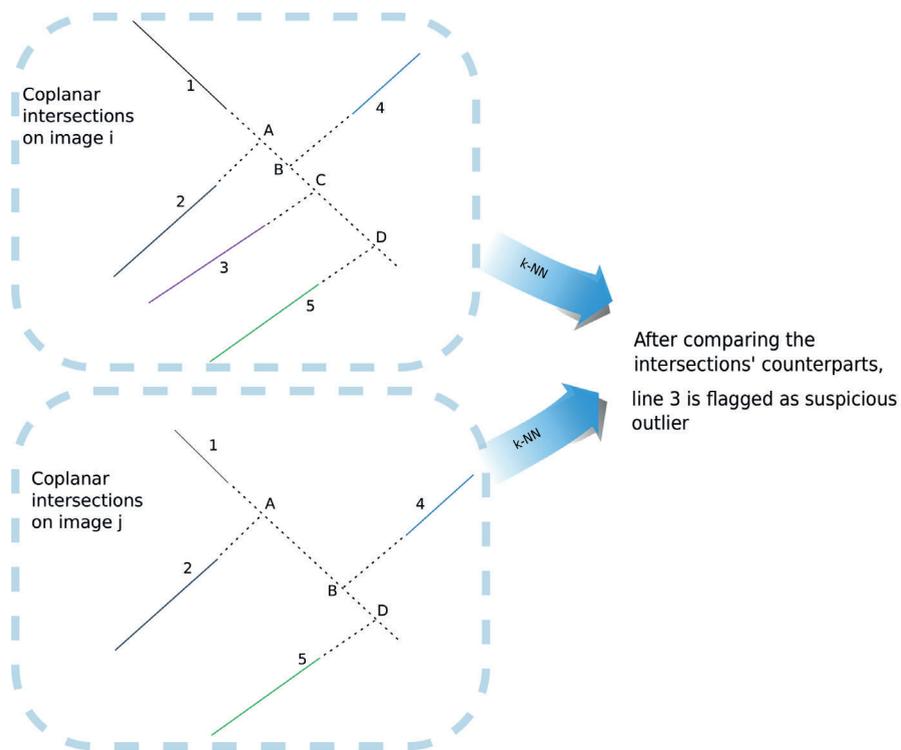


Figure 5.11: Flow chart embedding both the line detection algorithm and SALM.



**Figure 5.12:** Visual example of the k-Nearest-Neighbors search for close coplanar intersections.

### 5.3 Experimental results

Public datasets are selected looking for a fair compromise of scenes with and without texture, transformations that include camera translation, moderate global rotations, and changes in illumination conditions. The dataset "Castle" comprises the pictures  $\{0,1\}$  of the dataset [94]. It features a viewpoint change with a camera rotation, unveiling repetitive structures that can be tricky to identify. It was chosen in order to evaluate the structural cohesion of the line neighborhoods. The rest of datasets were extracted from [74]. The pairs "Low Texture" and "Textureless corridor" portrait a complicated classical interior of a building, featuring few observed long segments, and are selected to evaluate the resilience of the method to an absence of texture information. "Outdoor light" and "Leuven" are included to test changes of illumination in two different scenarios. Finally, "Drawer" combine changes of viewpoint and light exposition into a scene featuring repetitive similar line patterns. The *SALM* method is quantitatively compared against the state-of-the-art methods LS, CLPI, LBD and LJL. These implementations are provided by their respective corresponding authors, and its applicability is restricted to pairs of views.

The whole set of algorithms described in this thesis is implemented into a ROS node, which is executed by a notebook with Intel i7 3720QM Quad-Core and 16GB DDR3.

Both the line matching results for all the methods in the quantitative comparison and the processing times are shown in Tab. 5.1. The images with the experimental results are plotted in Tab. 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10. The averages of the most relevant variables are shown in Tab. 5.3. The Ground Truth evaluation of the methods adopted an approach similar to [138]: A line is marked as correct match if located 5 or less pixels apart from the human-perceived line in the orthogonal direction, and if the difference in line direction respect to the Ground Truth match of the counterpart is less than 5 degrees of rotation. Despite the ratio of matching inliers brings up meaningful information of the performance of each method, it is not possible to extract an unique global score to compare methods as a whole, as the average

segment length their similarity and redundancy might be more relevant in specific scenarios.

Dataset	Method	Line Correspondences	Correct counterparts	Redundant lines	Avg. length dissimilarity	Avg. segment length (pix.)	Total length (pix.)	Time (s)
Castle	CLPI	158	137	0	0.31	66.2	10459.6	45
	LJL	451	440	7	0.23	30.0	13559.6	704
	LBD	255	72	8	0.29	37.8	9643.2	4
	LS	183	171	ND	ND	ND	ND	8
	<i>SALM</i>	114	104	1	0.11	69.3	7897.3	125
Low-Texture	CLPI	0	0	-	-	-	-	20
	LJL	27	27	3	0.38	100	2699.9	4
	LBD	40	17	2	0.51	93.0	3722.8	1
	LS	44	40	ND	ND	ND	ND	7
	<i>SALM</i>	22	22	0	0.12	187.5	4124.9	38
Corridor	CLPI	0	0	-	-	-	-	50
	LJL	34	28	3	0.46	93.5	3177.8	3
	LBD	38	12	0	0.49	103.1	3919.3	1
	LS	53	40	ND	ND	ND	ND	3
	<i>SALM</i>	13	11	0	0.24	161.4	2098.1	45
OutdoorLight	CLPI	0	0	-	-	-	-	60
	LJL	210	202	1	0.27	36.2	7615.6	73
	LBD	193	25	7	0.37	49.8	9614.9	3
	LS	190	190	ND	ND	ND	ND	6
	<i>SALM</i>	85	80	2	0.10	68.2	5804.6	66
Leuven	CLPI	151	116	0	0.15	68.4	10328.4	54
	LJL	320	315	1	0.25	34.0	10889.5	100
	LBD	328	42	7	0.34	48.1	15774.5	1
	LS	190	190	ND	ND	ND	ND	7
	<i>SALM</i>	108	106	0	0.10	70.6	7624.8	85
Drawer	CLPI	1	0	-	-	-	-	7
	LJL	52	41	2	0.38	62.2	3300	38
	LBD	24	8	0	0.38	67.7	1692.8	1
	LS	49	45	ND	ND	ND	ND	6
	<i>SALM</i>	42	33	0	0.16	73.9	3104.9	20

**Table 5.1:** Quantitative evaluation of methods for matching of lines across two images

The data on the column of redundant lines on the Tab. 5.1 is externally computed from the resulting line coordinates. Two pairs of matched segments are considered redundant if and only if the shortest distance between them is less than 2 pixels, and their angles differ in less than 5 degrees. Redundant line segments are considered matching inlier if they properly meet the above mentioned requirements of correct fitting to the Ground Truth matching counterpart.

The results in Tab. 5.1 show that the *SALM* method extracts line correspondences featuring longer, less fragmented lines than the competition. In addition these segments of are more

similar in length to their respective counterparts compared with the results of other methods. Full length and non fragmented matched lines profit when the method is applied to line-based 3D reconstruction from three or more images. Therefore, an average segment length has been extracted from all the results, and shown in the seventh column of the Tab. 5.1. Besides this data, it is shown the total number of pixels covered by the matched segments on one image. Another measure that is crucial for the success of spacial reconstructions is the similarity between features in correspondence. This measure is valuable if the zoom global transformation is not featured in the image datasets, like in the ones included in this study. It is computed as the absolute value of the difference of lengths of the lines in correspondence, divided by the length of the longer segment, and shown in the sixth column of the Tab. 5.1. A better score is given to a result if both segments in correspondence are of similar length. This mark penalizes correspondences of atomic short segments, as they return poor geometric information of the scene. These length and dissimilarity values could not be computed for the method LS because the authors just provided the binaries and these exclusively return images showing matched lines and it does not output the coordinates. The best average matched segment length is obtained by the *SALM* method, with an average of 105.1 pixels, as written in Tab. 5.3. It is distantly followed by the other methods, with resulting average lengths of less than two thirds the number of pixels covered by the segments put in correspondence by the presented method. *SALM* also returns the best average dissimilarity score of 0.14. This result shows a high advantage compared to the other methods in this mixed comparative, because the second on the line is CLPI with an average dissimilarity score of 0.23.

The last column in the Tab. 5.1 shows the processing times in seconds for each specific method on the evaluated dataset. The highest processing times were taken by LJJL.

The method CLPI failed to return any correspondence from pairs of images featuring low texture and repetitive patterns. The most severe fragmentation was observed on the results of the methods LBD and LJJL. The method LJJL performed very well, just with downs in the images that present the segments more isolated. On the other hand, LBD performed poorly in

almost all the scenarios, showing a lack of understanding of the structure cohesion. LS was the second on the line, just losing edge on the textureless corridor scene. The comparative uses a fair mixture of scenes, and the *SALM* method obtained an inlier ratio second to none.

The method LS stands out for its good performance, despite it returns some inexistent line matches. For instance, there are no perceived edges along the matched segments 78, 155, 157 and 158 in the dataset "Castle", despite they might loosely match their counterparts. Moreover these are crossing other real lines on the images. Although these segments are detected over the same texture on the counterpart, no real edge is perceived there by a human.

### **Trying to match unrelated scenes**

The highest average inlier ratio for the selected datasets was achieved by the method *SALM*, though the method LS performed very well. We wanted to try both methods against pairs of unrelated captures, images that does not feature the same scene or structures. In order to build the pairs, images were picked randomly among industrial and public[30, 122] datasets. This is a way to test the robustness of the structural description of each method. The ideal result would be 0 matches for all the image pairs. In all the pairs both scenes feature a high number of short segments in every possible pose, forming repetitive structures. If the segment matching relied exclusively on their individual appearance, there would be many correspondences. Nevertheless both matching methods feature a robust structural description, and therefore few segments are matched by the methods. The results show that *SALM* returns lower number of correspondences than the method LS for the six image pairs, and for some of them it returned 0 matches. This reveals that the structural description of the method *SALM* is more robust than the one of LS.

Image pair		LS	SALM
		8	5
		8	7
		11	0
		11	0
		8	0
		14	0

**Table 5.2:** Number of line matchings given by the methods on a set of image pairs of unrelated views. The images of each pair have been randomly selected from industrial and public[30, 122] datasets.

### Analyzing the outliers for *SALM* by a human

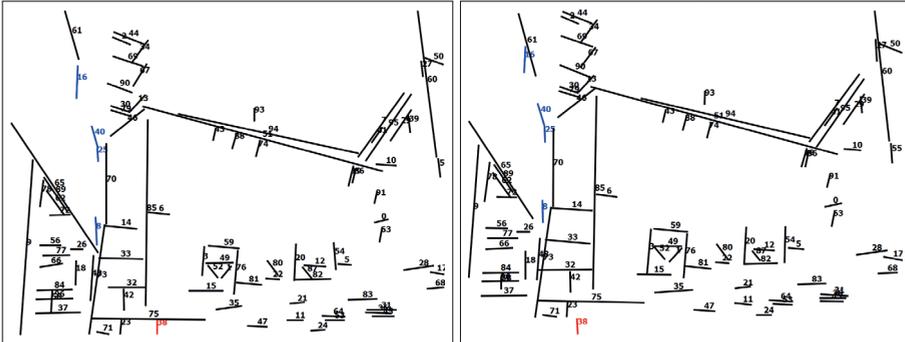
Fig. 5.14 shows the result of the Ground Truth assessment against the method *SALM*. Fig. 5.15 shows the matching results for public datasets[30, 122]. Firstly, a human notices that some outliers are located on regions of the images with high repeatability of patterns. The higher the number of repeated structures, the higher the probability of matching outliers. If the individual appearance of segments similar between many segments, the algorithm cannot exploit individual measures to discern matching hypotheses, so it has to rely mostly on the structural description. Secondly, the matching errors are more frequent in the outer regions of the neighborhoods of detected lines. In other words, the algorithm tends to fail more with segments that are not completely surrounded by neighbors. This happens as consequence of the nature of the structure based description of neighborhoods. The segments detected in these outer regions has less surrounding neighbors for the structural description, therefore they are more prone of missing their correct counterparts.

Method	Inlier ratio	Avg. length	Avg. dissimilarity	Processing time (s)
CLPI ([62])	27.25%	67.3 pix.	0.23	39
LJL ([75])	85.91%	59.3 pix.	0.33	154
LBD ([159])	24.27%	66.6 pix.	0.40	2
LS ([142])	91.94%	-	-	6
<i>SALM</i>	92.22%	105.1 pix.	0.14	39

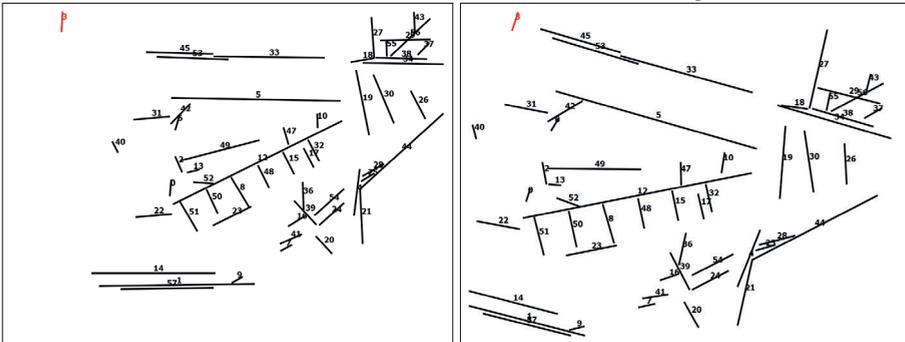
**Table 5.3:** Average line matching accuracy and processing times from the results shown in table 5.1



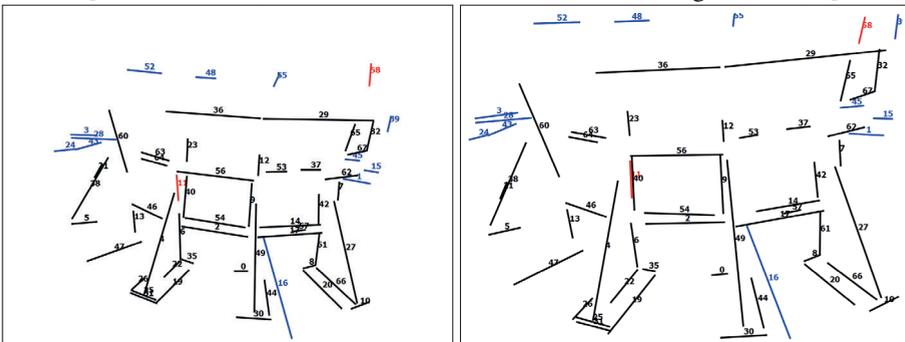
[Extracted: 220, 222; Matches: 97; Correct: 96; Matching score: 0,97]



[Extracted: 336, 170; Matches: 58; Correct: 57; Matching score: 0,96]



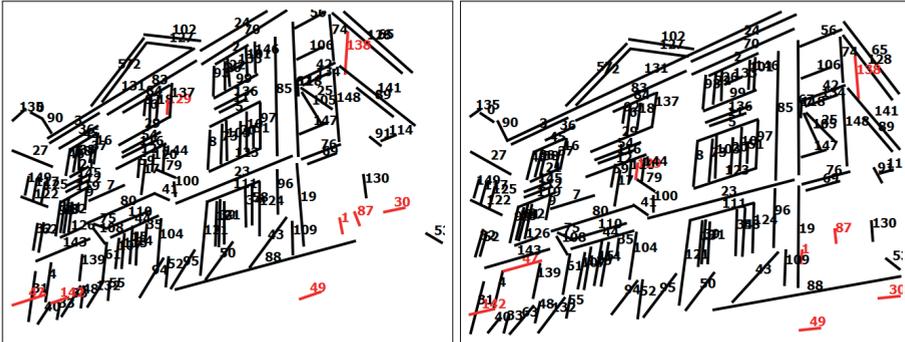
[Extracted: 297, 177; Matches: 68; Correct: 66; Matching score: 0,89]



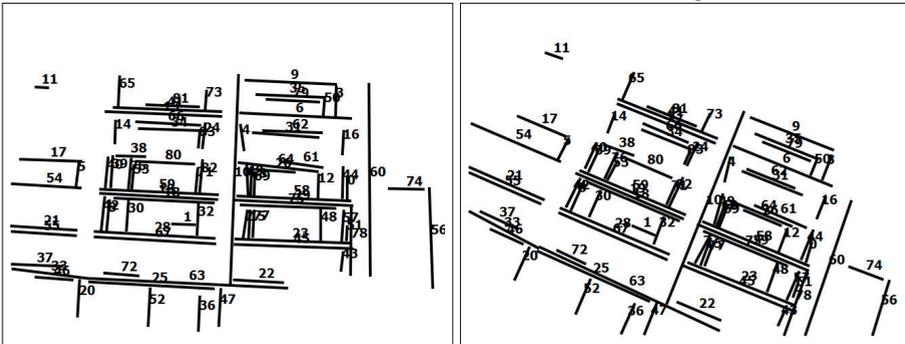
**Figure 5.14:** On the top, three pairs of industrial images. The rest of the figure shows the matching results by the method *SALM* for the three pairs of images. The coloring of the line segments corresponds to the human ground truth: Black lines are correct matches, red lines are wrong matches, and blue lines are non-significant lines.



[Extracted: 309, 282; Matches: 150; Correct: 142; Matching score: 0,92]



[Extracted: 196, 181; Matches: 84; Correct: 84; Matching score: 0,99]



[Extracted: 348, 307; Matches: 153; Correct: 151; Matching score: 0,96]

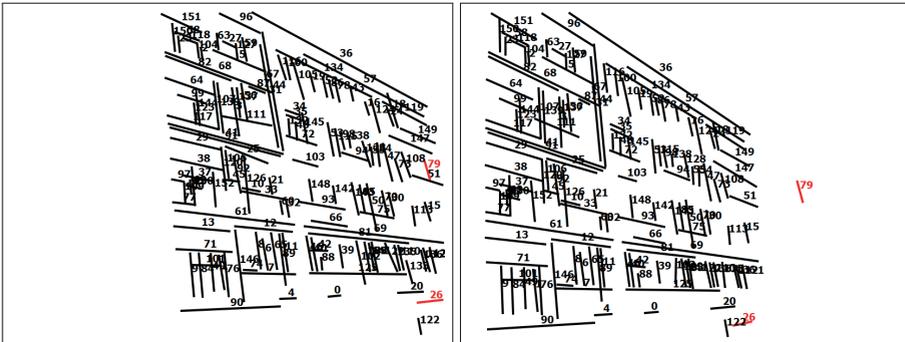


Figure 5.15: On the top, three pairs of images from public datasets[30][122]. The rest of the figure shows the matching results by the method *SALM* for the three pairs of images. The coloring of the line segments corresponds to the human ground truth: Black lines are correct matches, red lines are wrong matches, and blue lines are non-significant lines.

### Evaluation of the outliers detection algorithm

The outliers detection algorithm is only applicable when there is a noticeable change of view-point among both images. Therefore results have been extracted for both datasets featuring perspective change. These results shown on Tab. 5.4 brings up to the validity of the *SALM* outliers detection method. The addition of the outliers detection improves the results on both datasets, without increasing the processing times. On the dataset "Drawer", the outliers detection algorithm extracts the most noticeable segment correspondence outlier which is surrounded by lines visible on both images. There are other outliers on the figure, but the structural context does not contain minimum number of neighboring matching intersections. On "Castle", 13 suspicious line matches that are indicated, from whose just 4 are actual correct matches, and 9 are real outliers.

Dataset	Outliers detection	Correspondences	GT inliers	Inliers ratio	Processing time (s)
Castle	Without outliers detection	110	95	86%	46
	With outliers rejection	97	92	95%	46
Drawer	Without outliers detection	46	37	80%	20
	With outliers rejection	45	37	82%	20

**Table 5.4:** *SALM* method. Average line matching accuracy and processing times from the results shown in table 5.1

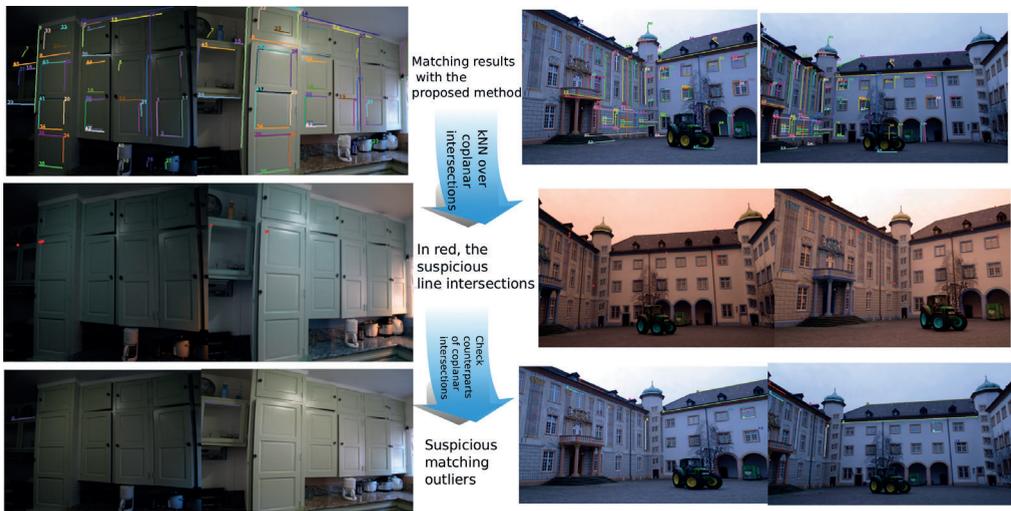
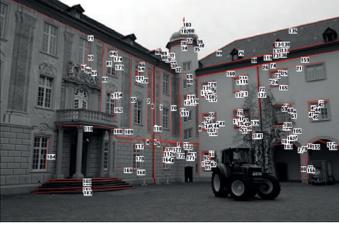
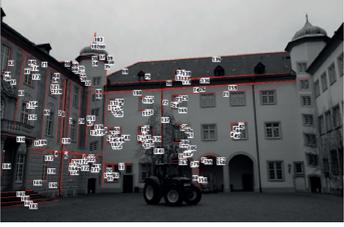
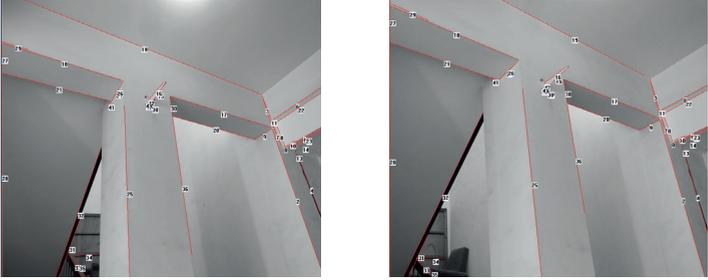


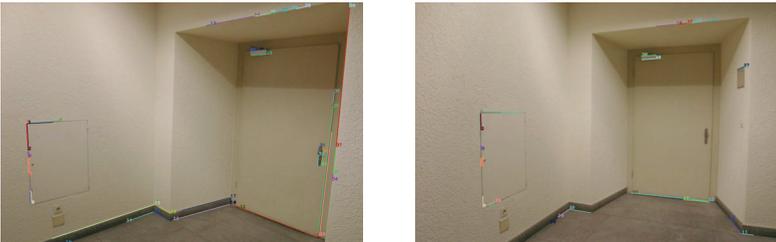
Figure 5.16: Examples of the search for outliers using coplanar neighbor intersections.

Method	Castle	
CLPI		
LJL		
LBD		
LS		
SALM		

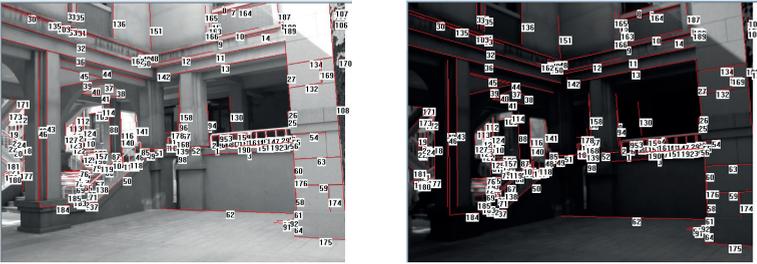
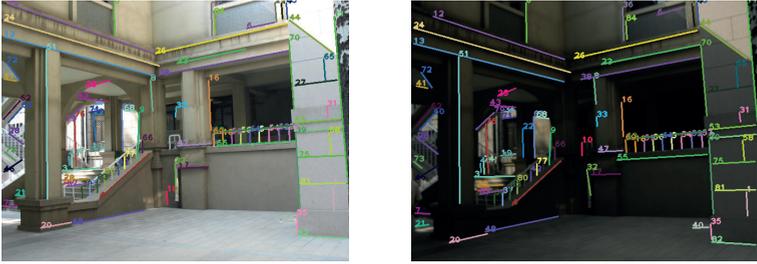
**Table 5.5:** Quantitative evaluation of methods for matching of lines against the dataset "Castle". The images are recommended to be displayed in a computer monitor with a 500% zoom.

Method	Low Texture	
CLPI	No result	
LJL		
LBD		
LS		
SALM		

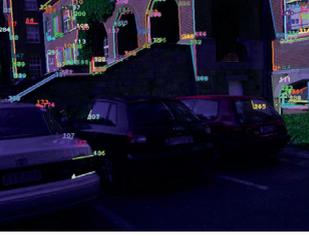
**Table 5.6:** Quantitative evaluation of methods for matching of lines against the dataset "Low Texture". The images are recommended to be displayed in a computer monitor with a 500% zoom.

Method	Textureless corridor	
CLPI	No result	
LJL		
LBD		
LS		
SALM		

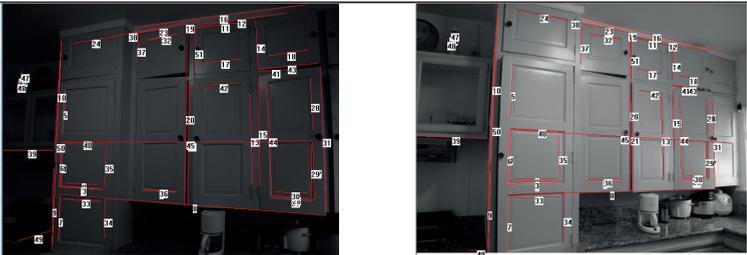
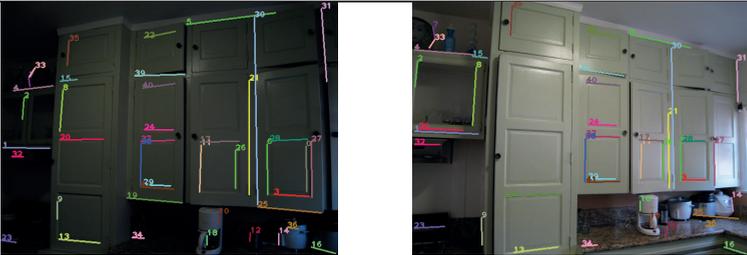
**Table 5.7:** Quantitative evaluation of methods for matching of lines against the dataset "Textureless corridor". The images are recommended to be displayed in a computer monitor with a 500% zoom.

Method	Outdoor light
CLPI	No result
LJL	
LBD	
LS	
SALM	

**Table 5.8:** Quantitative evaluation of methods for matching of lines against the dataset "Outdoor light". The images are recommended to be displayed in a computer monitor with a 500% zoom.

Method	Leuven	
CLPI		
LJL		
LBD		
LS		
SALM		

**Table 5.9:** Quantitative evaluation of methods for matching of lines against the dataset "Leuven". The images are recommended to be displayed in a computer monitor with a 500% zoom.

Method	Drawer	
		
CLPI		
LJL		
LBD		
LS		
SALM		

**Table 5.10:** Quantitative evaluation of methods for matching of lines against the dataset "Drawer". The images are recommended to be displayed in a computer monitor with a 500% zoom.

## 5.4 Applications for *SALM*

The presented method *SALM* is implemented in C++ for ROS Jade in Ubuntu 14.04, Mac OSX and also for Windows 7 and 10. This implementation has a wide range of applications in Computer Vision and engineering. A major part of *SALM* algorithm has been integrated in the method *3DwSkt* aimed for 3D abstraction for urban environments or man-made objects, which is followed in the next chapter of this thesis. The segment matching algorithm *SALM* can be used for indoor robot navigation, and also as complement for point based Simultaneous Localization and Mapping systems.

## Chapter 6

# 3D abstraction based on lines

This chapter goes through the engineering of a 3D abstraction method based on straight line segments. It employs the *SALM* line matching method for building correspondences between pairs of images. Then it groups the geometric relations and exploits them to generate the 3D sketch. The 3D sketch is a spatial representation based on lines that features estimations for the camera poses and for the 3D straight segments.

The logical evolution of the environment abstraction from multiple views is to incorporate line-based pipelines that do not require a detailed point-based description of the areas of interest. Beside, coplanar line primitives can be intersected to further reveal geometrical information. Likewise, groups of segments will also indicate the location of the most probable vanishing points from a camera plane[101]. These advantages make lines a good candidate to team with point feature detectors and descriptors[54, 160]. They offer the possibility of combining individual similarities of pairs of segments, related to strong constraints of parallelism and orthogonality [91, 18], and specially coplanarity constraints [64]. This chapter exploits the latter ones, and the potential outliers are determined based on the homography between different views. The exploited constraint is that the intersection of a pair of coplanar

lines, even if it might not resemble a physical point, is still geometrically invariant under perspective projection.

The remaining of this chapter is organized as follows: an enumeration of the headlines of the 3D sketching method, and its structure is discussed in Section 6.1, while a detailed description of the 3D devoted layers takes place in Section 6.2. An experimental quantitative study for the whole 3D sketching method is followed in Section 6.4.

## 6.1 High level overview for the *3DwSkt* 3D sketching method.

The *3DwSkt* 3D abstraction or sketching method features a set of improvements over the state-of-the-art algorithms based on lines. The contributions of the *3DwSkt* method are aimed for a more suitable 3D abstraction from images featuring low texture, and in scenarios in which is impossible to generate a dense point cloud:

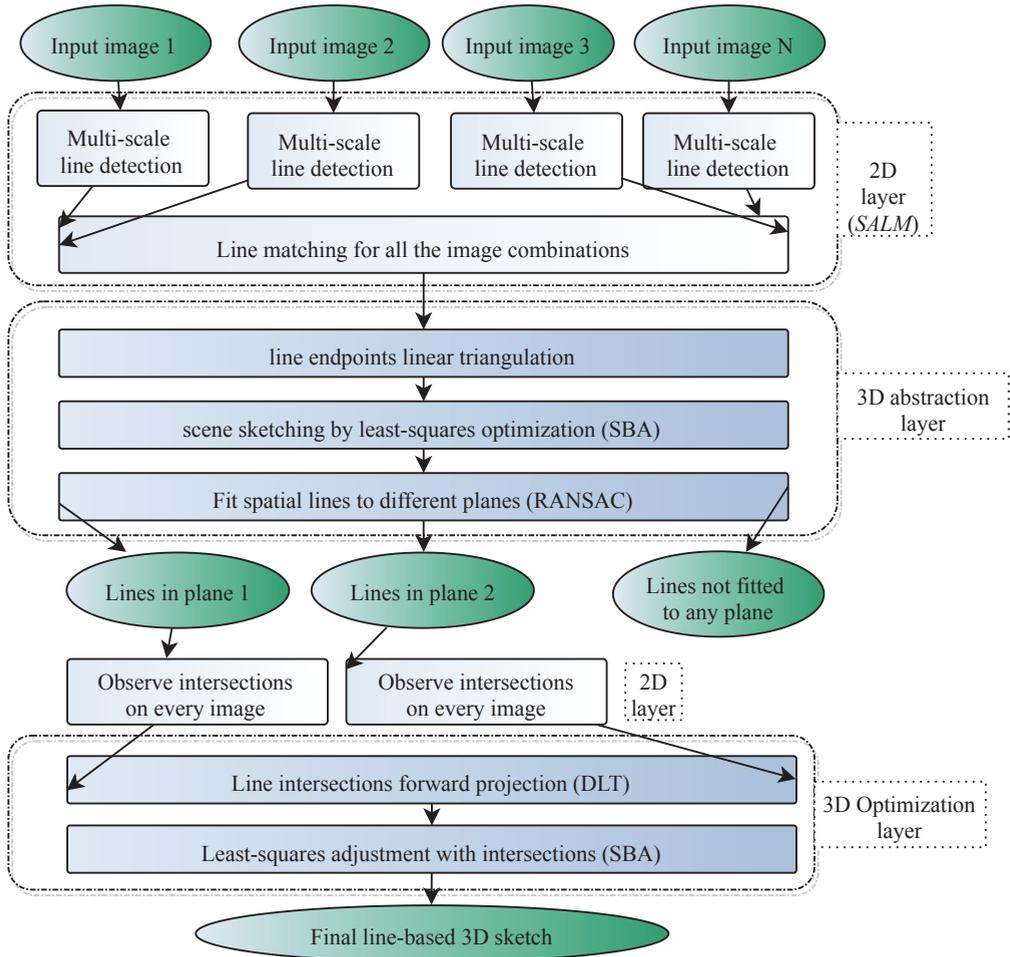
1. The *3DwSkt* method does not require to be provided with camera poses, nor a dense point cloud obtained from the input pictures. It makes feasible an abstraction based exclusively on line correspondences, performed independently over pairs of images. Our approach estimates camera extrinsics, but does not root the line matching on these spatial projections, preventing the uncertainty from SfM to propagate and merge with the uncertainty related to 2D line detection and matching. Previous reconstruction methods rely on a third party SfM pipelines[67][55], in order to source the camera poses and dense point clouds, to base line matching and 3D reconstruction on.
2. Our *Line Observation* is built by merging independent line matchings over pairs of images. The correspondences between segments for each pair of images are obtained with the *SALM* method. The groups of matched 2D lines define unique global entities, and every entity is defined before the 3D stages of reconstruction, in order to avoid the problem of redundancy of 3D lines and to reduce the number of matching outliers

when dealing with multiple views. On the other hand, some recent methods qualify the matching candidates based on their support on neighboring views[55], for later clustering them based on their spatial proximity, instead of performing a global matching of the observed lines individually. This may represent a source of uncertainty if the camera poses provided by the point-based SfM are not accurate enough, as the matching criteria is tied to the accuracy of every camera pose that was adjusted with point-based SfM, and used as input.

3. We propose to group the set of the spatial lines generated by the *3DwSkt* method, attending to coplanarity, by fitting them to different planes through RANSAC. The observed intersections of 2D coplanar lines are therefore described according to the observed matched lines. These segmented group of intersections are projected from every camera plane. They are finally included into the cost function for a second SBA run, taking advantage of this accurate source of observed points in correspondence. Most of the published methods are intended for urban environments, where many lines are coplanar. Nevertheless, they do not retrieve additional information from the images according to the spatial structure. Hence, the projected lines use to be the sole primitive input to the cost function for a least-squares minimization.

These contributions are intended for spatial abstractions of man made objects and environments. Our approach is fully automatic and only requires a set of pictures or video frames, and camera calibration parameters as the inputs. This complete automatic process is depicted will be followed in the different sections of this work.

The architecture of the line sketching model comprises three layers as shown in Fig. 6.1. i) The *2D layer* extracts and unifies the observed projections of edges from the set of views of a scene, where in general there might be large camera translations and rotations among the different views. ii) The *3D abstraction layer* builds stereo subsystems for every possible pair of views, which are fed into a first SBA, that outputs a 3D wireframe-based sketch and the



**Figure 6.1:** Flow for the *3DwSkt* method. The *SALM* method is integrated into the upper 2D layer.

camera poses, valid up to scale. iii) Finally, the *optimization layer* detects the intersections of the coplanar lines in the sourced images and add them to the least squares optimization for the sake of a more accurate reconstruction.

Line detection and matching between pairs of views are computed in the *2D layer* and based on the method *SALM*, described in Chapter 5. In this case the matching outliers detector based on planes is not teamed with *SALM*. Segments that remain unmatched after the described iterative matching process are definitely discarded, so all the line segments the 3D layer receives have a defined counterpart in other image of the input set. Scene 3D abstraction will be based on the pair-wise correspondences of line segments.

Line matching is performed by *SALM* between all the possible pairs of images available. The matching results are stored into **multi-view entities**. Each entity represent an unique 3D straight segment, and links to its counterpart in every image where it was matched. The first set of multi-view entities is created by the lines matched in the first pairwise combination of views. Relations between detected lines in the other images are incrementally built by matching among each other, and counterparts for the already created entities in other images are found. New multi-view entities are created for segments with known counterpart, if no previous record is found matching the segment. Line relations can be built even if line matching failed to flag them as counterparts. Moreover, a distance threshold can be set in order to minimize the number of pairwise line matching between images, and hence, the total computational cost. This is visually represented in Fig. 6.2: no direct matching has been done between views 1 and 6, though these are related because the matching among views 3 and 4 identifies the line matched between captures 1 and 3, to be the counterpart of the one related between views 4 and 6. Algorithms look for multi-view matching inconsistencies during this merging stage. These outliers rejection algorithm scan through all the counterparts of each multi-view feature entity, looking for potential outliers.

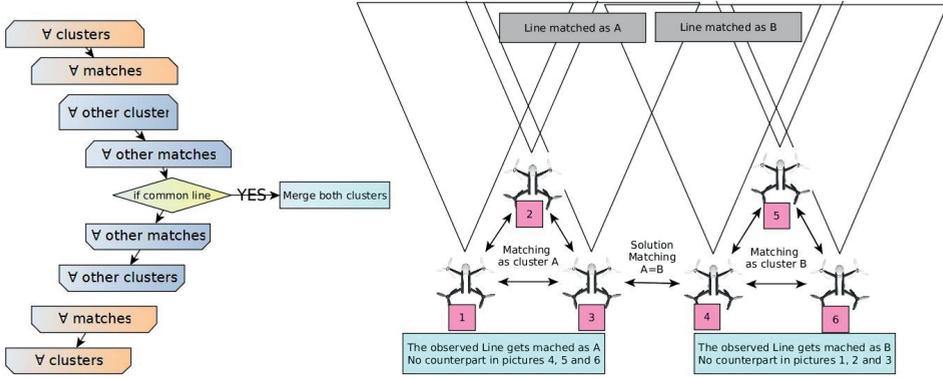


Figure 6.2: Example of merging of two groups of counterparts.

## 6.2 Low level description of the $3DwSkt$ method.

Given an unordered set of images, related to a set of unknown camera planes  $\Upsilon = \{\Upsilon^1, \Upsilon^2, \Upsilon^3, \dots, \Upsilon^M\}$ , the  $3D$  abstraction layer computes the initial estimations for each 3D line segment in the set  $\Gamma = \{\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_N\}$ . These are estimated from two or more observed projections, therefore  $\Gamma_i$  may firstly embed one or more estimations of the 3D line before they are unified into a single estimation. The set of line projections observed in  $\Upsilon$  is represented as  $l = \{l_1^1, l_2^1, \dots, l_N^1, \dots, l_N^M\}$ . A Line Feature is defined as a subgroup of projections from  $l$  of the same 3D line  $\Gamma_i$ . The set of Line Features is noted as  $L = \{L_1, L_2, \dots, L_N\}$ . The 3D lines  $\Gamma$  are obtained by forward projecting the endpoints of  $l$  from pairs of camera planes of  $\Upsilon$ , by using linear triangulation, analogously to Direct Linear Transformation (DLT)[109]. The cameras  $\Upsilon$  are sequentially bundled in the same reference frame, by stacking the new ones according to the  $L$ -to- $\Gamma$  correspondences computed in the previous stereo pair of cameras. The unification of all the estimations on each  $\{\Gamma_i\}$  is computed as the center of gravity of these estimations.

The 3D sketch  $\{\Upsilon, \Gamma\}$  generated by linear triangulation is used as input for the  $3D$  abstraction layer. The least-squares optimization named SBA is based on the Levenberg–Marquardt algorithm, and uses as input the estimated camera extrinsics  $\Upsilon$  and the set  $\Gamma$ , now containing

unique estimations for each 3D line.

The *optimization layer* fits the spatial line segments  $\Gamma$  to a set of different planes  $\mathcal{P}$ .  $\Gamma$  is therefore segmented into different groups according to the planes  $\mathcal{P}$ , and so is done with their projections  $L$ . The group of Line Features fitted to the plane  $\mathcal{P}_l$  is noted as  $F_l$ . The intersections of the coplanar lines  $F_l$  on the camera plane  $\Upsilon^j$  is the set of points  $\mathcal{T}_l^j$ . These intersections are observed on the input images, and described like a feature point, being the descriptor the pair of two coplanar lines drawing it. Like was performed in the *3D abstraction layer* with the endpoints of  $l$ , now the point correspondences in  $\mathcal{T}_l^j$  are fed into the linear triangulation algorithm, in order to create initial estimates for the 3D intersections by forward projecting  $\mathcal{T}_l^j$ . The set of estimations for the 3D intersections is a sparse cloud and is denoted as  $\mathcal{R}$ . The 3D intersections  $\mathcal{R}$  enter the least-squares optimization. These new iterations of SBA are based on a simple 3D point reprojection distance to the observations on  $\Upsilon$ . The outputs are the new optimized estimations for  $\Upsilon$  and the optimal 3D intersections  $\mathcal{R}$ . The line poses are corrected by forward projecting them from the newly estimated camera planes  $\Upsilon$ , generating the final sketch  $\{\Upsilon, \Gamma\}$ . The high level diagram on Fig. 6.3 shows the process described in this section.

The 3D line based sketch  $\{\Upsilon, \Gamma\}$  is built from the knowledge of correspondences among line projections  $l$  on camera planes and the intrinsics of all the cameras. The linear triangulation of these observations is performed from scale-space images, and this is a novelty of the method compared to other recently published ones[55, 160, 7]. This allows to discriminate and weight down lines that have been detected on two or more scales with a different slope. The practical consequence is that prior to any 3D extrapolation of the observed lines, matching inliers with inconsistent endpoint location among scales on both images can be avoided, as these lines might introduce uncertainty in the camera.

The first problem is that the camera poses  $\mathbf{P}$  are unknown. We have chosen to estimate it from the endpoint correspondences of  $l$ . For this task we first estimate the Essential matrix  $\mathbf{E}$  for the camera pairs, by using the Five-Point Algorithm[102] and RANSAC for hypotheses

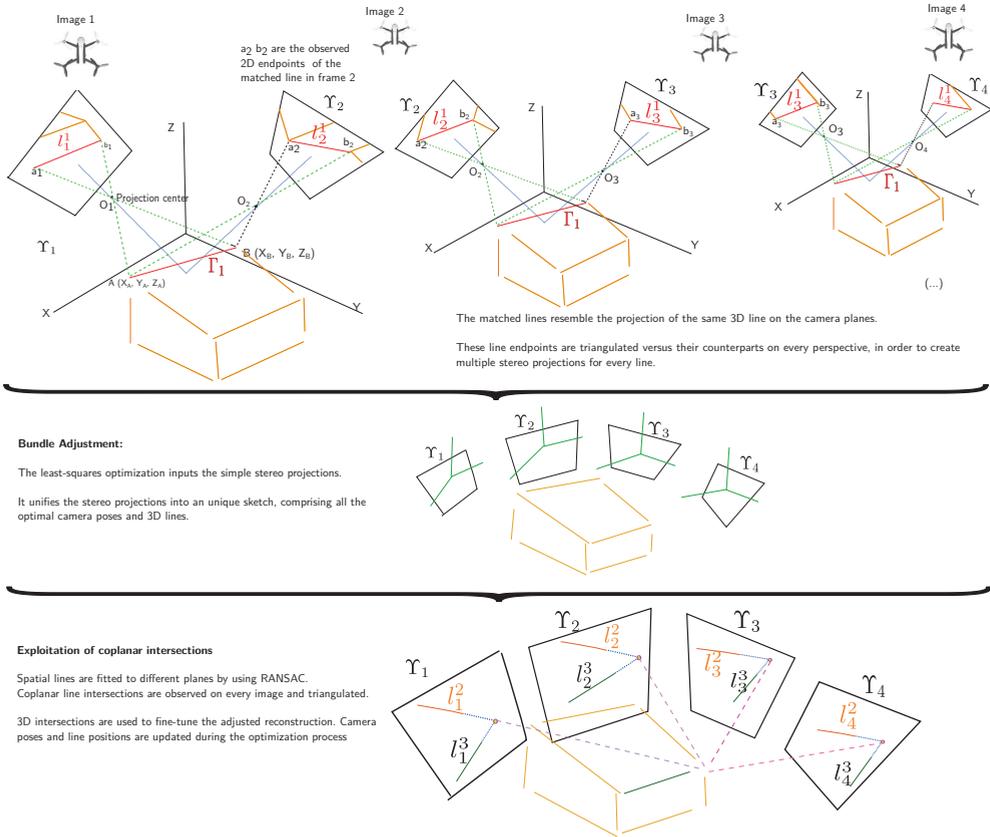


Figure 6.3: Graphic representation of the 3D abstraction layer of the method.

generation. Having  $E$  and  $l$ , the relative camera rotation and translation among the first pair of cameras  $P^j = [R|t]$  are estimated using cheirality check and discarding the triangulated endpoints that are not in front of the cameras. The left camera is chosen to have the pose  $P^1 = [I|0]$ , and the newly added cameras are stacked from this position in the unique reference frame.

The forward projection of lines in 3-space is described in the page 196 of Hartley and Zisserman's book[109]. The 3D forward projection  $\Gamma_i$  of a line, bundled in the same reference

frame, can be obtained using the *DLT* method [48] on the set of stereo 3D camera back-projections. This is performed in homogeneous coordinates because it allows to consider line endpoints in the infinite. Therefore, from now on, when a 2D point is mentioned it will be supposed homogeneous coordinates. There exists a  $3 \times 3$  matrix  $\mathbf{E}$ , known as the essential matrix, such that if  $u$  and  $u'$  are a pair of matched points, then  $u'\mathbf{E}u = 0$ . If a sufficient number of matched points are known, the matrix  $\mathbf{E}$  may be computed as the solution of an overdetermined set of linear equations. For the present problem, the internal calibration of the cameras is known, therefore it is possible to determine from  $\mathbf{E}$  the relative placement of the cameras and hence the relative locations of the 3D points corresponding to the matched points. A linear triangulation method is projective-invariant because only camera and line distances are minimized.

The execution order for DLT over the cameras (view pairs) obeys to the inlier ratio of line matching, i.e. the number of matched lines divided by the total number of detected lines for each camera pair. Therefore, the DLT starts with the segments on the pair of cameras  $\{\Upsilon^a, \Upsilon^b\}$  with the highest inlier ratio. Next, the camera  $\Upsilon^c$  is chosen among the ones with the higher inlier ratio of line matching with  $\Upsilon^a$  and  $\Upsilon^b$ . At successive steps, a new camera  $\Upsilon^n$  is chosen among the ones with the higher inlier ratio of line matching with previously selected cameras.

The inaccuracy in the observation of the 2D lines  $l$  implies that there will not be a 3D point  $\mathbf{X}$  which exactly satisfies that their projections on cameras  $\Upsilon^1$  and  $\Upsilon^2$  are  $\mathbf{x}_1 = \mathbf{P}^1\mathbf{X}$ ,  $\mathbf{x}_2 = \mathbf{P}^2\mathbf{X}$  respectively, and that the image points do not satisfy the epipolar constraint  $\mathbf{x}_2\mathbf{F}\mathbf{x}_1 = 0$ . Therefore, a projective-invariant triangulation method, that only minimizes the image distances to the observations, is required. A linear triangulation[109] method does not depend on the projective frame in which  $\mathbf{X}$  is defined.

The forward projection from a normalized 2D line observed on the camera image plane

$m$ , denoted by  $l_i^m$ , is the plane  $P_m^T l_i^m$ , so the condition for a point  $X_a$  to be in this plane is:

$$(l_i^m)^T P_m X_a = 0. \quad (6.1)$$

Each point  $X_a$  returns a linear equation in the entries of  $P_m$ . Denoting by  $x_{m,E}^i$  and  $x_{m,F}^i$  the forward projection of the endpoints of  $l_m^i$ , named  $X_E^i$  and  $X_F^i$ , under  $P_m$ , then any other 3D point on the line  $X^i(\mu) = X_E^i + \mu X_F^i$  projects to a point:

$$x_i^m(\mu) = P_m(X_{i,E}^i + \mu X_{i,F}^i) = x_{i,E}^m + \mu x_{i,F}^m, \quad (6.2)$$

which is on the line segment  $l_i^m$ .

In this method, an unique reference frame is built. The world reference system is fixed onto the first camera, hence its camera matrix,  $P_E$ , is computed with  $R_E = I$  and  $T_E = 0$ . The extrinsics for the partner camera  $P_m$  on the baseline is obtained from the essential matrix by using RANSAC. Before the subsequent DLT triangulations with a new camera, its extrinsics are estimated also by RANSAC from the 2D-3D results of the already computed DLT. From here, new cameras will be added incrementally, just one per DLT iteration, in order to avoid DLTs between two uninitialized camera projection matrices.

For DLT it is required a set of observed line correspondences,  $l_j^m$  to  $l_j^n$ , matched among images. The projection on the image plane of camera  $m$  of an endpoint  $X_{i,E}$  of the spatial line  $\Gamma_j$  is denoted as  $x_{j,E}^m = P_m X_{j,E}$ . This point on the  $m$ -th camera plane is matched to its counterpart on the  $n$ -th camera  $x_{j,E}^n = P_n X_{j,E}$ . Both equations can be combined into  $A X_{j,E} = 0$ , where  $A$  is the matrix of equation coefficients. It is built from the matrix rows  $A_r$  contributed from each correspondence, whose resemble the movement of each line between both views.  $X_{j,E}$  contains the unknowns for the endpoint position.

By using the cross product on the  $m$ -th camera:  $l_j^m \times (P_m X_{j,E}) = 0$ ,

$$x_m(\mathbf{p}_m^{3T} \mathbf{X}_{j,E}) - (\mathbf{p}_m^{1T} \mathbf{X}_{j,E}) = 0, \quad (6.3)$$

$$y_m(\mathbf{p}_m^{3T} \mathbf{X}_{j,E}) - (\mathbf{p}_m^{2T} \mathbf{X}_{j,E}) = 0, \quad (6.4)$$

$$x_m(\mathbf{p}_m^{2T} \mathbf{X}_{j,E}) - y_m(\mathbf{p}_m^{1T} \mathbf{X}_{j,E}) = 0, \quad (6.5)$$

where  $(x_m, y_m)$  and  $(x_n, y_n)$  are the coordinates of  $\mathbf{x}_{j,E}^m$  and  $\mathbf{x}_{j,E}^n$  respectively.  $\mathbf{p}_m^{rT}$  is the  $r$ -th row of  $\mathbf{P}_m$ . It can be decomposed similarly for  $\mathbf{P}_n$ , and compose the equation of the form  $\mathbf{A}\mathbf{X}_{j,E} = 0$ . Solving:

$$\mathbf{A} = \begin{bmatrix} x_m \mathbf{p}_m^{3T} - \mathbf{p}_m^{1T} \\ y_m \mathbf{p}_m^{3T} - \mathbf{p}_m^{2T} \\ x_n \mathbf{p}_n^{3T} - \mathbf{p}_n^{1T} \\ y_n \mathbf{p}_n^{3T} - \mathbf{p}_n^{2T} \end{bmatrix}. \quad (6.6)$$

The solution for the 4 equations of the over-determined problem (four equations for four homogeneous variables) is only valid up to scale. The set of points in space mapping to a 3D line  $\Gamma_j$  via  $\mathbf{P}_m$ , is the plane  $\mathbf{P}_m \Gamma_j$ .

The result of the linear triangulation process is  $\Gamma_i$  and  $\mathbf{v}^j$ , represented in cartesian coordinates. The next subsection will cover the change of coordinate system that will allow an efficient least-squares optimization for the pose of the lines and cameras.

## SBA optimization

The SBA for the matched lines uses a simple function that takes as the cost a distance measure between the observed line segments in the original images and the reprojected ones on the camera plane. If a line in 3-space is represented by Plücker coordinates, then its image can be expressed as a linear map on these coordinates[109, 157]. For the *3DwSkt* approach, spatial lines are represented in Plücker coordinates in the cost function of the optimization process.

A forward projected 3D line  $\Gamma_j$  is represented in Plücker coordinates, following [109, 7], as  $\Gamma_j = (\mathbf{u}_j, \mathbf{t}_j)$

$$\mathbf{u}_j = \frac{\mathbf{X}_{j,0} - \mathbf{X}_{j,1}}{\|\mathbf{X}_{j,0} - \mathbf{X}_{j,1}\|} \quad (6.7)$$

$$\mathbf{t}_j = \mathbf{X}_{j,d} \times \mathbf{u}_j, \quad (6.8)$$

where  $\mathbf{u}_j$  is the unit vector in the direction of the line, and  $\mathbf{t}_j$  is the moment vector around any 3D point  $\mathbf{u}_j$  laying on the straight segment  $\Gamma_j$ , respectively. As this moment vector  $\mathbf{t}_j$  is independent of the choice of  $\mathbf{X}_{j,d}$  on the segment, we have used the center of gravity for the group of forward projections obtained by DLT with all the camera pairs.

The second constraint imposes that the determinant of the Plücker matrix  $\mathbf{U}$  has to be null[109].

The Cayley representation used for the 3D lines  $\Gamma_j$  is similar to the one employed by[160]. Plücker coordinates have two constraints:

$$\|\mathbf{u}_j\| = 0 \quad , \quad \mathbf{u}_j^T \mathbf{t}_j = 0 \quad , \quad (6.9)$$

which keep the degrees of freedom of a spatial line equal to four. It is not recommended to enforce the constraints during the optimization process[7]. This is the reason why the Cayley representation is defined to ensure that an orthogonal matrix can be built from the Plücker coordinates of a line under the constraints in Eq. 6.9. This orthogonal matrix can be written as

$$\mathbf{Q} = \left[ \mathbf{u}_j, \quad \frac{\mathbf{t}_j}{\|\mathbf{t}_j\|}, \quad \frac{\mathbf{u}_j \times \mathbf{t}_j}{\|\mathbf{u}_j \times \mathbf{t}_j\|} \right] \quad , \quad (6.10)$$

and the skew-symmetric matrix

$$[\mathbf{s}]_{\times} = (\mathbf{Q} - \mathbf{I})(\mathbf{Q} + \mathbf{I})^{-1} \quad , \quad (6.11)$$

The Cayley representation of a spatial line is defined as the four dimensional vector  $\mathbf{v} = (\omega, \mathbf{s})$ , being  $\omega = \|\mathbf{m}\|$  the distance between the origin of the reference frame and the closest

point lying on the line, and the three dimensional vector  $\mathbf{s}$  encodes the rotation of the line. From the chosen definition for  $\mathbf{Q}$  at Eq. 6.10, it can be derived that

$$\mathbf{u}_j = \mathbf{q}_1 \quad , \quad \mathbf{t}_j = \omega \mathbf{q}_2 \quad . \quad (6.12)$$

The equations to convert from Plucker coordinates to the Cayley representation's four parameters are Eq. 6.10 and Eq. 6.11. These four parameters are used in the SBA to update the spatial lines during the optimization process. When the convergence of the least-squares optimization is reached, the lines are converted from Cayley representation back to Plucker coordinates:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3] = (\mathbf{I} - [\mathbf{s}]_{\times})^{-1}(\mathbf{I} - [\mathbf{s}]_{\times}) = \frac{(1 - \|\mathbf{s}\|^2)\mathbf{I} + 2[\mathbf{s}]_{\times} + 2\mathbf{s}\mathbf{s}^T}{1 + \|\mathbf{s}\|^2}, \quad (6.13)$$

The inputs to the line based SBA are the estimated 3D lines converted to Plucker coordinates [52] and written on Cayley representation[7], and the cameras intrinsic parameters. In order to avoid the singularity around the center of the image plane, the error distance for each line is chosen as the squared shortest distance from observed endpoints to the reprojected infinite line, as suggested by[160]. The center of projection for the camera had been previously subtracted from the reprojected lines, in order to compare with the observed line on the image.

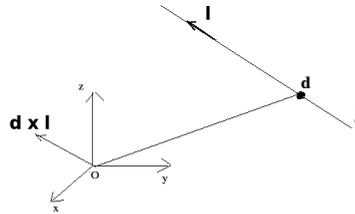
$$\mathbf{Q} = [\mathbf{l}, \mathbf{e}_1, \mathbf{e}_2] \quad (6.14)$$

$$\det(\mathbf{U}) = l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0, \quad (6.15)$$

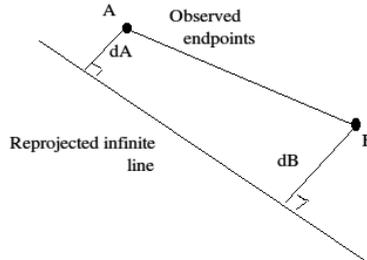
being  $l_{ab}$  the components of the Plucker matrix  $\mathbf{U}$ .

Each input 3D estimate for an spatial line is generated from the main of the triangulated endpoints of all the lines of each cluster,  $\mathbf{p}_2$  and  $\mathbf{p}_1$ , in vectorial form.

The SBA is performed incrementally, starting with the pair of views with the lowest rate of matching inliers, as suggested in [129], and then sequentially adding the remaining cameras with their respective DLT results. After each camera has been added to the set of



**Figure 6.4:** The Plücker coordinates for the line  $\Gamma_j$  are  $(\mathbf{u}_j, \mathbf{t}_j \times \mathbf{u}_j)$ , taking any point  $\mathbf{d}$  on the line, and  $\mathbf{l}$  being the unit vector in the direction of the difference between its endpoints



**Figure 6.5:** In order to avoid the singularity around the center of the image plane, the error function is chosen as the squared shortest distance from the observed segment endpoints to the reprojected infinite line, as suggested by[160]

views to be adjusted, the algorithm executes a line-based SBA run, based on the cameras that have been added so far, and the lines that have counterpart on these views. The cost function minimizes a residual equal to the squared shortest distance from the observed endpoints to the re projection of the adjusted infinite line on the camera plane, as suggested by[160] and represented in Fig. 6.5.

$$\underset{\Sigma \varepsilon_i}{\text{minimize}} \quad \varepsilon_i^2 = d_{iA}^2 + d_{iB}^2, \quad (6.16)$$

The output of the algorithm will be the estimated optimal pose of 3D line segments  $\{\mathbf{L}^i\}$ , and

the optimal poses for the cameras  $\{(\mathbf{R}_i, \mathbf{T}_i)\}$ .

### 6.3 Line intersections

Any line segment detection process is prone to misalignments, wrong segmentation, incorrect placement of endpoints and also cases of unsuited merging of two curved edges resembling a real straight line. Likewise, some line endpoints are neither well defined in an image, when resembling the edges of shadows, or over-exposed photos. Even with these non optimal conditions, edge segments might be correctly matched on several pairs of view. While this kind of inlier matchings would be welcome in a comparison of matching algorithms, they introduce uncertainty in the location of endpoints set used for the spatial abstraction. Some improvements can be achieved by exploiting coplanarity constraints.

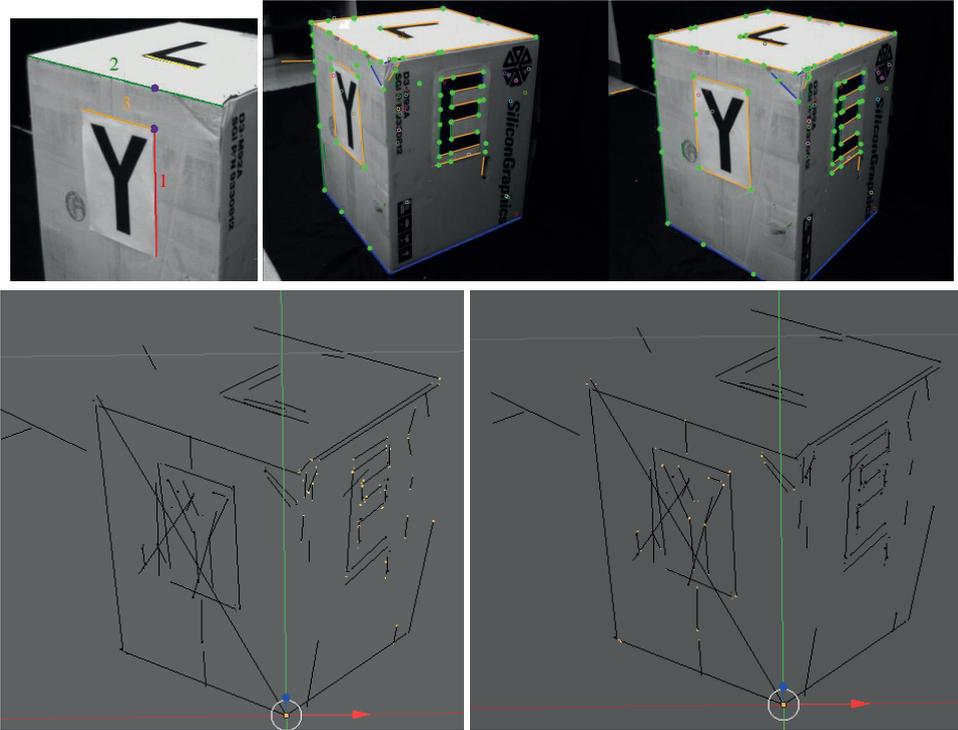
The goal of the *optimization layer* is to extract structural information from groups of coplanar lines and exploit them to improve the accuracy of camera poses and the placement of 3D lines. This approach is rooted on the hypothesis that coplanar line intersections encode accurate geometric information for the reconstruction. The points we are looking for are not just joints of segments touching each other, but also more relevant intersections of the extensions of long structural segments.

The point of intersection of coplanar lines is computed in the original 2D input frames, once we have computed the first 3D line based abstraction. The 3D estimations for the lines  $\Gamma$  are fitted to a set planes  $\mathcal{P} = \{\mathcal{P}^1, \mathcal{P}^2, \dots, \mathcal{P}^p\}$ , being  $p$  the total number of planes. Two spatial lines  $\Gamma_1$  and  $\Gamma_2$  in Plucker coordinates are coplanar if and only if the reciprocal product of their Plucker coordinates is zero:

$$(\hat{\mathbf{u}}_1, \mathbf{t}_1) * (\hat{\mathbf{u}}_2, \mathbf{t}_2) = \hat{\mathbf{u}}_1 \cdot \mathbf{t}_2 + \hat{\mathbf{u}}_2 \cdot \mathbf{t}_1 = 0. \quad (6.17)$$

The group of coplanar Line Observations  $F_t$ , fitted to the plane  $t$ , contains all their counterparts on all the camera planes extended and intersected. The intersection of the observations

$l_i^j$  and  $l_{ii}^j$  is denoted as  $\mathcal{F}_{i,ii}^j = \{l_i^j \cap l_{ii}^j\}$ . The complete group of observed intersections of  $F_t$  is  $\mathcal{F}_t$ . This is expressed right after the first loop in Algorithm 2.



**Figure 6.6:** The image on the top left left is a graphic explanation for the usage of coplanarity conditions. Just the intersections of matched lines that are coplanar are counted (line 1 intersects 2 and 3, but not the yellow line under the "L" label). The pair of images placed in the top right shows the result of the method *SALM* for line matching between two images of the dataset "CUBE"[134]. The intersections of coplanar segments extracted by the algorithm are highlighted in green. Note that just the intersections of coplanar segments were highlighted. The pair of images on the bottom of the figure are from the 3D sketch generated from the pictures {3,4,5,6} of the dataset. These shows the segmentation of endpoints after RANSAC. The planes with more inliers feature 68 and 39 inliers respectively.

Each element in the set of observed coplanar intersections  $\mathcal{F}$  is dealt as a point feature

described by the pair of Line Observations that originate it, and put in correspondence on all images with counterparts for both original Line Observations. The first 3D estimations for the intersection points on all images  $\mathcal{T}$  are obtained by linear triangulation, and denoted with  $\mathcal{L}$ . This is done analogously as would be computed with described feature points, as followed in Algorithm 2 inside the second cluster of loops. For this method, every 2D line intersection is eligible to enter the spatial abstraction if and only if it matches a proximity rule and an structural threshold. The first rule limits the sum of the distances from the intersection to the segments to be less than 1.5 times the sum of lengths of both lines, in order to avoid short segments with inaccurate slope to intersect far from them and with high uncertainty. The second threshold is for the angle drawn by both lines, required to be more than  $15^\circ$  for the sake of accuracy. A visual example of the algorithm for exploitation of coplanar intersections described in this section is embedded in Fig. 6.6. The following subsection explains how the obtained intersections are used like point correspondences in a second run of SBA.

### SBA with lines and planes

The second run of the SBA employs a cost function that inputs the 3D estimations for the line intersections  $\mathcal{L}$ , in addition to the previously optimized camera extrinsic parameters and lines. The squared cost that will be used to compute the residuals is expressed in the last cluster of loops in Algorithm 2. For this optimization process, the residuals are computed for each iteration by using the squared cost like a simple problem of feature point reprojection error:

$$\{\epsilon_k^j\}^2 = d(\text{proj}_{\Upsilon^j}\{\mathcal{L}_k\}, \mathcal{T}_k^j)^2, \quad (6.18)$$

where  $\text{proj}_{\Upsilon^j}\{\mathcal{L}_k\}$  is the reprojection of the 3D intersection  $\mathcal{L}_k$  over the camera plane  $\Upsilon^j$ , and  $\mathcal{T}_k^j$  is the location of the detected intersection in frame  $j$ .  $d(\cdot, \cdot)$  denotes the 2D Euclidean distance. An estimation of the location of a reconstructed intersections  $\mathcal{L}_k$  has been experimentally proven to be accurate enough, when verifying the following conditions:

1. Both lines are not parallel, which can be guaranteed using a threshold of  $\theta > \pi/8$  for the intersection angle. In Plucker coordinates,  $\Gamma_a$  and  $\Gamma_b$  being parallel implies:

$$\hat{\Gamma}_a = \pm \hat{\Gamma}_b. \quad (6.19)$$

2. The perpendicular distance from a plane to each intersection is not greater than the average perpendicular distance from the planar surface to the lines associated to it:

$$d_{\perp}(\mathcal{P}_p, \mathcal{L}_k) < 1/N_p \sum_k d_{\perp}(\mathcal{P}_p, \Gamma_k), \quad (6.20)$$

being  $\Gamma_k$  a 3D line fitted to the plane  $\mathcal{P}_p$ , and  $N_p$  the number of lines associated to plane  $\mathcal{P}_p$ .

After the convergence of the optimization process, the lines  $\Gamma$  are forward projected from the newly optimized camera planes  $\Upsilon$ . The 3D intersections  $\mathcal{L}$  can be represented alongside the lines  $\Gamma$ .

**Algorithm 2:** Line based sketching

---

**Data:**  $L$ , Camera intrinsics  
**Result:**  $\Upsilon$ ,  $\Gamma$ , and  $\mathcal{T}$   
initialization;  
Pairwise Linear Triangulation for  $L \rightarrow \Gamma$   
**for**  $\Upsilon^j \in \Upsilon$  **do**  
    **for**  $\Gamma_i \in \Gamma$  **do**  
        Add cost function with:  
         $\{\varepsilon_i^j\}^2 = d(\text{proj}_{\Upsilon^j} \Gamma_i, A_i^j)^2 + d(\text{proj}_{\Upsilon^j} \Gamma_i, B_i^j)^2$ , with  $\{A_i^j, B_i^j\} \in L_i$  observed  
        endpoints of  $L_i$  on  $\Upsilon^j$   
    **end**  
    **if**  $j \geq 2$  **then**  
        Solve SBA for  $\Upsilon$ ,  $\Gamma$   
    **end**  
**end**  
Fit  $\Gamma$  to several planes  $\mathcal{P} \rightarrow \mathcal{F}$   
**for**  $\Upsilon^j \in \Upsilon$  **do**  
    **for**  $\mathcal{P}_h \in \mathcal{P}$  **do**  
        **for**  $\{l_i^j, l_{ii}^j\}_h \in \mathcal{F}_h$  **do**  
            **if**  $\{\mathcal{T}_h = \{l_i^j \cap l_{ii}^j\} \in \mathcal{T},$   
             $l_i^j \parallel l_{ii}^j, \theta(l_i^j, l_{ii}^j) > \pi/8\}$  **then**  
                Add  $\mathcal{T}_h$  for Linear Triangulation  
            **end**  
        **end**  
    **end**  
**end**  
Pairwise Linear Triangulation for  $\mathcal{T} \rightarrow \mathcal{L}$   
**for**  $\Upsilon^j \in \Upsilon$  **do**  
    **for**  $\mathcal{L}_k \in \mathcal{L}$  **do**  
        Add cost function with:  
         $\{\varepsilon_k^j\}^2 = d(\text{proj}_{\Upsilon^j} \{\mathcal{L}_k\}, \mathcal{T}_k)^2$   
    **end**  
**end**  
Solve SBA for  $\Upsilon$ ,  $\mathcal{L}$   
Forward-project  $L$  from  $\Upsilon \rightarrow$  optimized  $\Gamma$

---

## 6.4 Experimental results of 3D sketch generation

Ground truth datasets for SfM are built by taking synthetic images from 3D models[60], or with real pictures[132] teamed with 3D model data including the pose of the cameras and the measurements from 3D scanning or Lidar. Both synthetic and real Ground-Truth datasets come with a 3D model. The resulting point cloud is aligned with the Ground Truth mesh. The normal distance between the surface of the mesh and the points is computed.

In the case of 3D line sketch: Compare sketch to mesh. Equivalent to cloud to mesh distance. 3D straight segments must be discretized into points. In order to measure the difference in proportions between the generated 3D sketch and the Ground Truth mesh, the normal distance between the surface of the mesh and the points on the lines is computed. Using the obtained error distances, discretized points on the lines are coloured to account how far they are from the surface of the mesh. There are several variables that condition the resulting 3D sketch number of images. The number of images showing common elements of the scene is one of them. Secondly, the number of segments that can be matched between images. Thirdly, the transformation between both images might condition the matching inlier ratio, and hence, the number of segments correctly projected into space.

For the *3DwSkt* method, the length of the final 3D lines will depend on the fragmentation of the detected lines, and its number is closely related to the number of line correspondences between the images. Therefore, results of 3D reconstructions will unavoidably depend on the performance of the method for stages before the spatial projection. Quantitative measurements for 3D abstraction are performed on Ground Truth datasets. The proportions of the generated sketch is measured based on the distance between the segments and the Ground Truth mesh.

The experimental section of this work is willing to test the *3DwSkt* line based reconstruction method and compare it against other methods under the same conditions. Our fully automatic approach performs all the processes from image datasets or video frames capture,

through line detection and matching, to camera pose estimation and 3D abstraction. This section is willing to prove the following facts:

1. The *3DwSkt* method, if based exclusively on lines, returns the pose of the cameras and a line-based reconstruction of the scene up to scale. An analysis compares the number of 3D segments against the number of 3D points obtained with a SIFT-based SfM[148]. Based on this point cloud it is computed the result for the method *Line3D++*[55], and this line based abstraction is also compared with the result of the *3DwSkt* method. In these test cases the *3DwSkt* method is not using any point-based SfM pipeline, but instead it just employs self detection and matching of lines. These are described in Sec. 6.4.
2. The *3DwSkt* method improves the results of feature point based SfM pipelines in various scenarios. For proving this fact, the presented method is teamed with SfM pipelines based on KAZE. Our results are compared against Ground Truth, and the results obtained by[55]. These tests are followed in Sec. 6.4.
3. The current implementation is able to extract a reconstruction by its own, just requiring the images and the camera extrinsics as input.

In this study we have used a set of well known public image datasets [60], [146], [134]. These datasets feature a balanced compromise of line structure combined with large textured elements, it is intended to allow a fair comparison with feature point based reconstruction methods.

Camera intrinsics are provided as inputs for each set of views, and every method has to estimate the extrinsic parameters of all the cameras, the location of feature points or the poses of the principal lines defining the structure of the region of interest, and the planes adjusted to these lines, when applicable.

Two kinds of experiments can be distinguished depending on whether the objective is to validate both the camera poses and the 3D line sketch against the ground truth, or just to validate the 3D line sketch. In the first case the camera poses are estimated altogether with the set of 3D lines, as explained in this chapter. Nevertheless, in the second case the employed camera extrinsics are primarily estimated from point-based SfM. The 3D lines obtained with our model are densely discretized into a set of 3D points laying on the line. Then the Hausdorff distance is computed from these points to the surface of the ground truth polygonal model. The *RMSE* of this distance is computed and provided as a measure of the error of the prediction compared against the ground truth. The figures referred from this section show the complete result returned by the algorithm, no alteration has been performed by human, and all the lines are shown, including outliers and wrongly posed lines.

In Table 6.1 the main high level differences between the *3DwSkt* method and the method *Line3D++*[55] used for the evaluations are overviewed.

Method	Line detection and matching	Depends on cloud	First 3D estimation	Final result
<i>3DwSkt</i>	Appearance and structure[80]	NO	Linear triangulation	SBA
<i>Line3D++</i> [55]	LSD and SfM pipeline	YES	Depends on point cloud	SBA

**Table 6.1:** Overview of evaluated methods

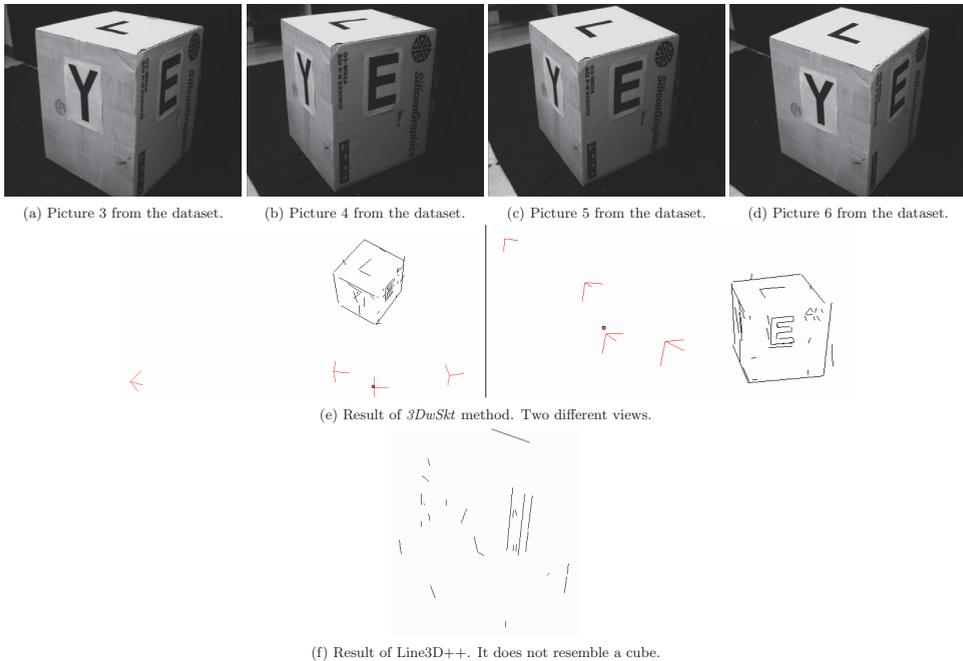
## Experiments estimating camera poses and 3D lines

In this subsection we are evaluating the implementation of the model strictly as explained in this chapter, meaning that the camera extrinsics are estimated right from the matched lines on the images and a solution obtained without using feature points at all. Therefore the images showing a common region of interest are selected from the datasets. The pictures showing just a fraction of this region were also avoided. The selected subsets comprise picture numbers {3-6} from the dataset "*CUBE*"[134]. The result is presented in Fig. 6.7. This set is characterized by low resolution and poor illumination. In order to compare this result, the

same subset has been used with the VisualSfM[148] SIFT-based SfM pipeline teamed with *Line3D++*[55], nevertheless no usable line based result has been obtained with this subset comprised of just 4 low resolution pictures. This is an example of test case for which the *3DwSkt* method performs better than *Line3D++*[55]. Being the line matching based on 2D appearance features, it can still reconstruct the scene on the absence of a thick point cloud. The figure shows 4 estimated camera poses and the line sketch formed by 50 spatial segments. The good fitting to the mesh of a cube proves that the proportions are fairly well represented with the *3DwSkt* method.

### **Experiments teaming *3DwSkt* with feature point descriptors**

The most difficult scenario for line matching are sets of images featuring a large number of lines embedded in an structure of repetitive pattern. Also, most of the ground-truth public datasets we found in the literature feature wide baselines, large relative distances between pictures, and not all the structural elements are framed on each picture. The ideal dataset for line based reconstruction should include most of the structural elements on every picture, and minimize the number of occlusions. A solution we took for creating the test cases was to pick subsets from these sequences of images that can fairly serve for evaluation and comparison. On the other hand, matching outliers make difficult the convergence of the SBA of a large datasets. This happens also in the case of employing line intersections, because a single line mismatch can trigger the misplacement of multiple intersections. For these cases it is unavoidable to team the pose estimator based on line correspondences with a point-based descriptor. In this study KAZE[3] point feature detector and descriptor is teamed with the *3DwSkt* method and executed after the line detection and matching stages, and right before the triangulation among views. The addition of this SfM pipeline gives an initial estimation for the cameras, avoiding the increased uncertainty of obtaining it from triangulation of the line endpoints.



**Figure 6.7:** Comparative results using the *3DwSkt* method of line-based reconstruction exclusively based on lines. The selected subset from the dataset "BOX"[134] comprises the picture numbers {3-6}. a) Sample from the dataset. b) Result obtained with the *3DwSkt* method. Two different views of the sketch are plotted, in order to show that the proportions are correct. c) A SIFT-based reconstruction performed with VisualSfM[148] returned a sparse cloud, and therefore this result of *Line3D++*[55] presents just 26 unconnected 3D lines, and it is not understandable.

Having an initial estimation of the cameras allows to set a maximum distance between the cameras for the pairwise line matching to be performed among the respective images. In the implementation, this solution translates into setting a distance threshold between the camera centers for the line matching algorithm to be executed over the respective images. This prevents that the algorithm spend time on the search for line correspondences between two images showing opposite faces of an object, and avoids potential matching outliers due

to similar sides of the same man-made object. Additionally, limiting the line matching to close neighboring views drops the total processing time dramatically.

The quantitative evaluation of the precision of the method is employing the synthetic dataset *Timber-Frame-House*[60], for performing two different comparative evaluations with the methods *Line3D++*[55] and Jain[60]. Results are shown on Table 6.2, Fig. 6.8 and Fig. 6.9. The *3DwSkt* method returns a 3D line based sketch of the model from as low as 6 images, whereas *Line3D++*[55] requires 12 or more images to draw an abstraction that includes the main long structures of the house. Moreover, the number of images required for the model to output all or mostly all the edges visible in the original 3D representation with fair accuracy is represented on Fig. 6.10. In order to consider a valid reconstruction for this comparison it was also required that most of the 3D segments extend to the whole length of their Ground Truth counterparts, oppositely to feature sequences of atomic short 3D segments. For a resulting sketch, the number of 3D segments correctly represented as a unique entity resembling the whole edge is denoted as  $\mathbf{P}$ . Quantitative measurements have been performed on the results. Every 3D line is discretized into points, and the measure used to evaluate the precision of a resulting abstraction is the distance between these points and the Ground Truth mesh. The level of subdivisions of the octrees at which this Cloud-To-Mesh distance computation was performed is 3[44]. The test cases are created looking for the minimum number of images that resulted in a reconstruction comprising most of the lines shown in the Ground Truth dataset. These include both the edges of the polygonal model and the lines present in the texture. In order to capture all the segments of the representation, each test case comprise two groups of pictures, one for the front and the second one capturing the opposite side of the house. The test cases are labeled as  $S_6$ , comprising image numbers {6,9,86,46,49,126},  $S_8$  further add two more images {89,129} to the list,  $S_{10}$  includes {8,10,12,88,90,48,50,52,128,130}, and  $S_{12}$  further adds images {92,132} to the latter. The resulting 3D line sketches from both sides of the house are aligned by using common lines. This completed sketch is finally aligned to the Ground Truth in order to measure the preci-

sion. The time employed by the *3DwSkt* method is clearly outperformed by *Line3D++*[55]. One of the reasons is the lower number of lines included in the results of the latter one. In the case of *Line3D++*[55], the time taken by the required SfM pipeline[148] is added to the amount.

The quantitative comparison and Ground Truth evaluation is performed firstly using a low number of images, and then more images are incrementally added. The comparisons performed against  $S_6$  and  $S_8$ , are shown in Fig. 6.8. The result proves that the *3DwSkt* method is able to obtain a number of structures of the house from as low as 6 images, and still holding a decent accuracy. On the other hand, the method *Line3D++*[55] returns sparse short segments for both cases. This sparsity complicates the understanding of what the spatial line cloud is resembling, and difficult the alignment to the Ground Truth mesh. Note that this method also fails to retrieve any long segment of the house for these test cases.

The results for a larger number of images of 10 and 12, using  $S_{10}$  and  $S_{12}$ , are shown in Fig. 6.9.  $S_{10}$  turns out to be the minimal test case to return a complete line-based reconstruction from the original synthetic images by using the *3DwSkt* model. It is the subset with the minimum number of images required for the method to generate a complete abstraction, meaning that it includes most of the segments resembling the main edges and lines on the original 3D model and texture. It is proved that the *3DwSkt* method is able to return a greater number of complete segments that resemble the original representation, while holding a level of precision equivalent to the one achieved by the method *Line3D++*[55] in this case. For  $S_{12}$  the point cloud obtained by VisualSfM pipeline[148] is dense enough for *Line3D++*[55] to generate a fair abstraction, showing completeness and continuity between the segments.

The obtained mean values, and the Ground Truth segment count for the mentioned test cases against the two methods are displayed on Table 6.2. The mean distance to GT mesh is lower for *Line3D++*[55] than for the *3DwSkt* method, mainly because their line poses are estimated based on SIFT-originated point clouds. Nevertheless, the number of 3D segments represented as a whole for the first three subsets is much lower than in the *3DwSkt* method,

as clearly seen on the images of Fig. 6.8 and Fig. 6.9.

The result of the method of Jain et al.[60] is included in this subsection for reference. It is based on Ground Truth camera poses, instead of estimating these from the lines or SfM pipelines. The actual code neither the executables are provided by the authors, so it could not be tested against the *3DwSkt* method. Nevertheless the shown result gives an idea of the precision achieved by the method, as reference. This result is provided by the authors, altogether with the complete dataset and the used subset, with 72 images.

Fig. 6.10 shows a qualitative comparison of the *3DwSkt* method and [55] based on the same dataset[60], but including a larger number of cameras facing the same region of interest. The employed number of 12 images was the lowest necessary for SIFT to generate a cloud thick enough to allow *Line3D++*[55] to create the 3D abstraction with most of the line segments present in the model. This minimal cloud must be comprised by at least 10,000 SIFT-generated points. The greater this number of points gets decreased, the *3DwSkt* method gains more edge compared to *Line3D++*[55]. For this test case we want to put in value that the *3DwSkt* method provides an additional grouping of the structural information. The coplanar lines and their intersections are grouped and the 3D abstraction of these intersections are plotted. This is shown as an analogy of a SIFT-like generated point cloud. With the higher amount of views of this test case, the result of *Line3D++*[55] shows a large number of segments precisely located, nevertheless most of the shown edges are comprised by a discontinuous array of short segments. This result contrasts with the completeness and uniqueness of the segments returned by the *3DwSkt* method.

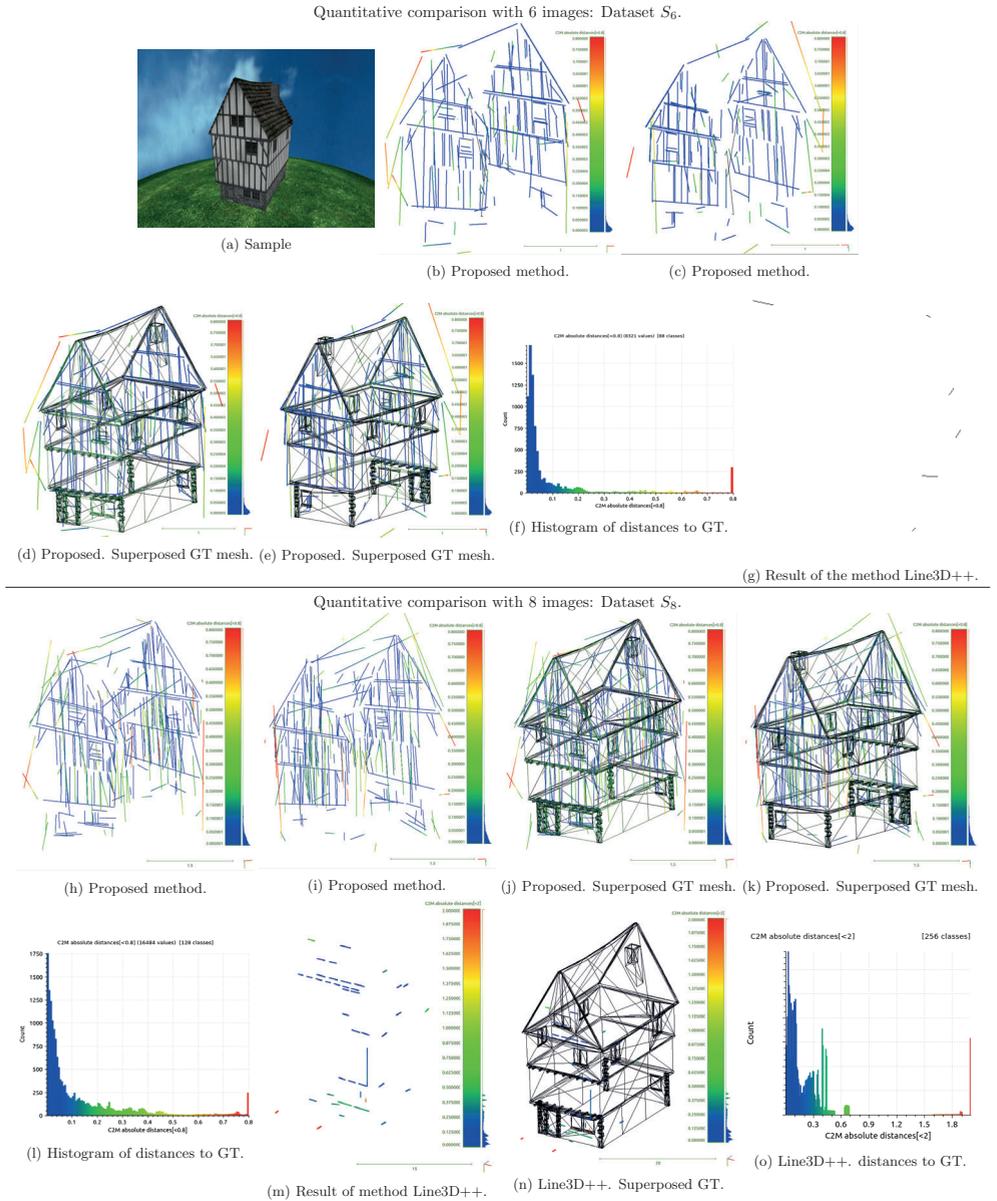
It was found out that our method is able to obtain a reconstruction in adverse situations where *Line3D++*[55] is not able to return a valid abstraction, for instance when the number of pictures is low and they do not present clear textures. The famous datasets[146] comprise few low resolution images, and are always difficult to represent by feature point based SfM pipelines. These have been selected because the addition of 3D segment abstractions improve the overall representation. In addition to plot the reconstructed lines by the *3DwSkt* method,

Set	Method	Lines	Completed lines	Processing time	Mean distance to GT	std deviation
$S_6$	<i>3DwSkt</i>	175	52	122s	0.1	0.36
	<i>Line3D++</i> [55]	6	0	18s	No alignment	No alignment
$S_8$	<i>3DwSkt</i>	294	74	160s	0.21	0.3
	<i>Line3D++</i> [55]	91	1	20s	0.07	0.11
$S_{10}$	<i>3DwSkt</i>	475	244	235s	0.08	0.24
	<i>Line3D++</i> [55]	290	112	30s	0.08	0.11
$S_{12}$	<i>3DwSkt</i>	556	305	374s	0.8	0.25
	<i>Line3D++</i> [55]	622	176	42s	0.07	0.24

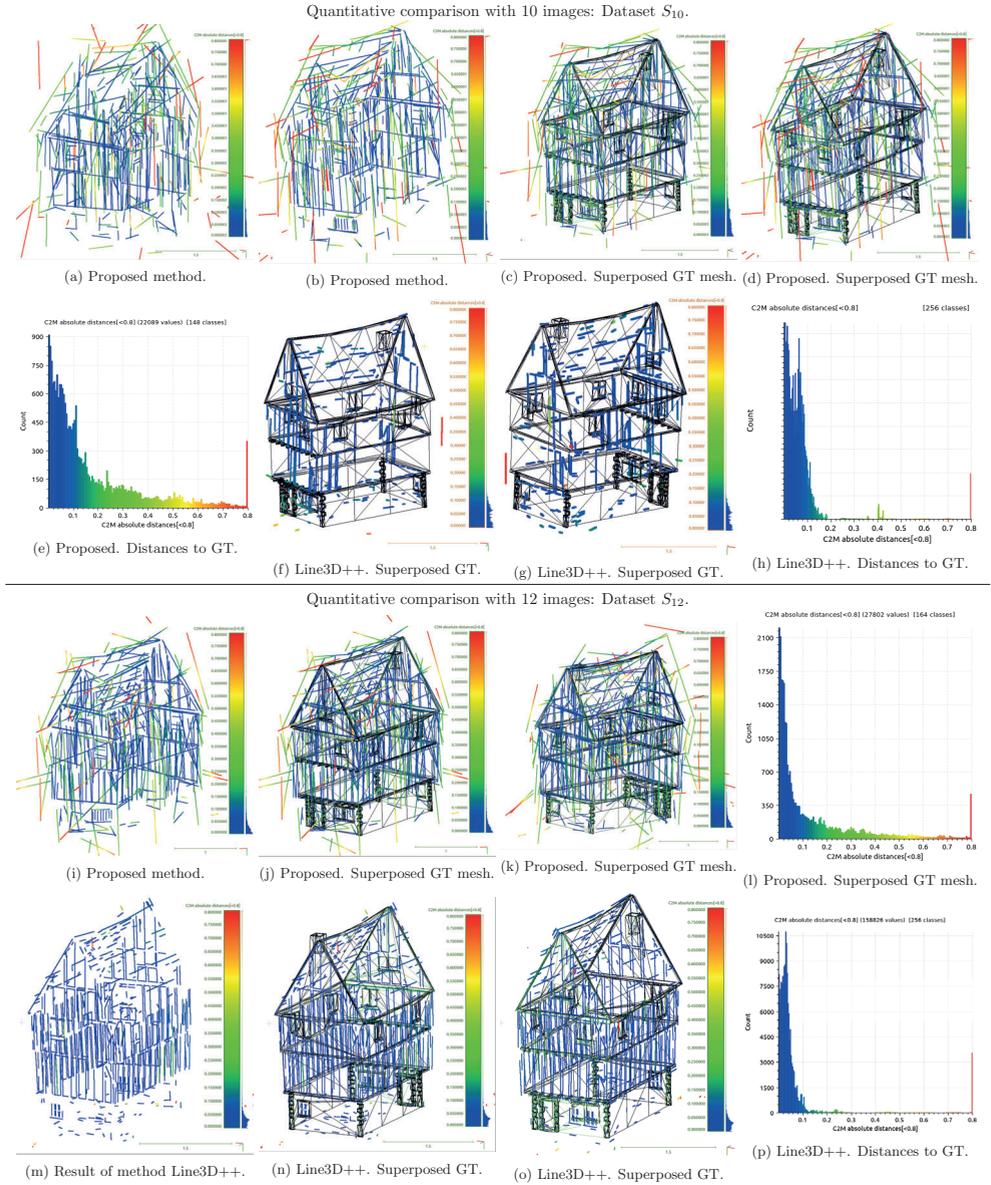
**Table 6.2:** Quantitative comparison of both methods against the subsets  $S_6$ ,  $S_8$ ,  $S_{10}$  and  $S_{12}$  created from the public dataset Timber-Frame-House[60]

Fig. 6.13 also show the estimated spatial intersections of the coplanar lines. For the first one, built of just 3 pictures, the point cloud provided by VisualSfM[148] is too sparse for *Line3D++*[55] to be able to construct a 3D abstraction. For the second dataset, comprising 5 views, it does place 3D segments because the obtained point cloud is denser, as the set comprises more views. Nevertheless, the cloud is not thick enough for *Line3D++*[55] to represent all the main lines in the scene, as can be observed in the picture included in Fig. 6.13. On the other hand, the *3DwSkt* method manages to obtain an usable result for both datasets. In addition, on both cases the lines were fitted to several planes, being the two planes with more inliers the corresponding to the vertical walls of each building. This allowed to use the intersections of coplanar lines to estimate the camera poses for the final bundled results shown on the figures.

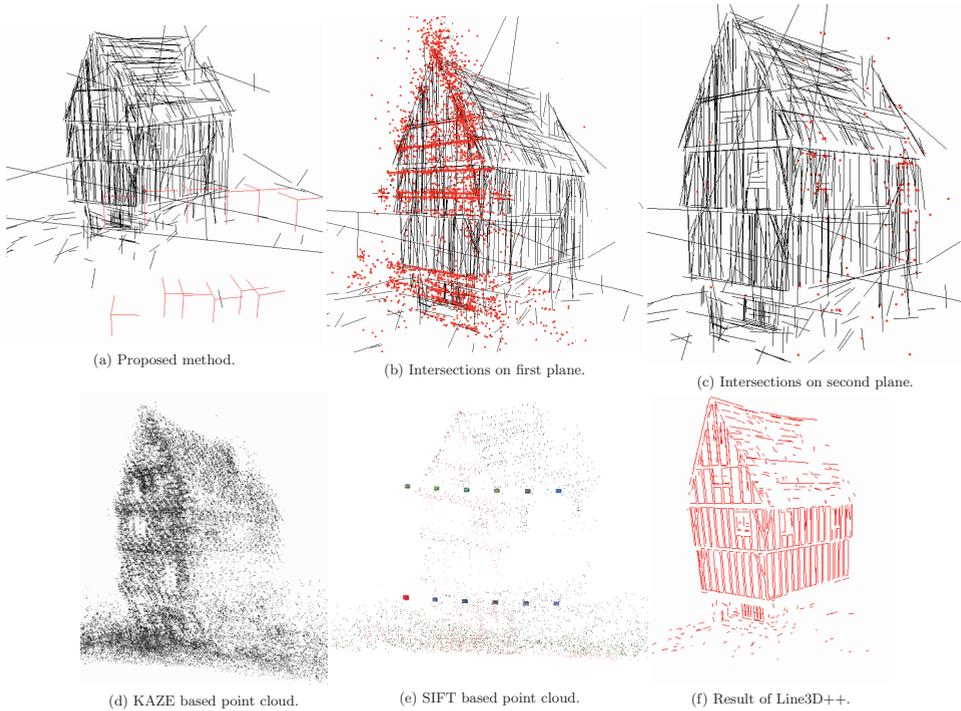
The dataset Building-Blocks[60] is used for another quantitative analysis. Fig. 6.12 allows the comparison of the results of the *3DwSkt* method with the ones provided by VisualSfM[148] and *Line3D++*[55]. A subset of 6 views from close camera positions was selected in order to enhance the differences between the methods. In this case the *3DwSkt* method is able to obtain a bundled result with more lines and with visually better grasped connectivity than the abstraction provided by *Line3D++*[55].



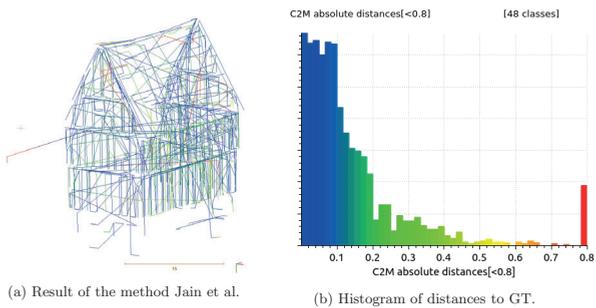
**Figure 6.8:** Quantitative comparison using the sets  $S_6$  and  $S_8$ [60]. This figure is better viewed on a screen with a 4x zoom. a) Sample of the set. b) and c)  $3DwSkt$  method against  $S_6$ , resulting 175 lines. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the  $3DwSkt$  method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the  $Line3D++$ [55] method. It has been aligned with the Ground Truth mesh. h) to l) The  $3DwSkt$  method against the set  $S_8$ , with 294 segments. m) and n) same measurements for the result by  $Line3D++$ [55]. o) shows the histogram for this latter result.



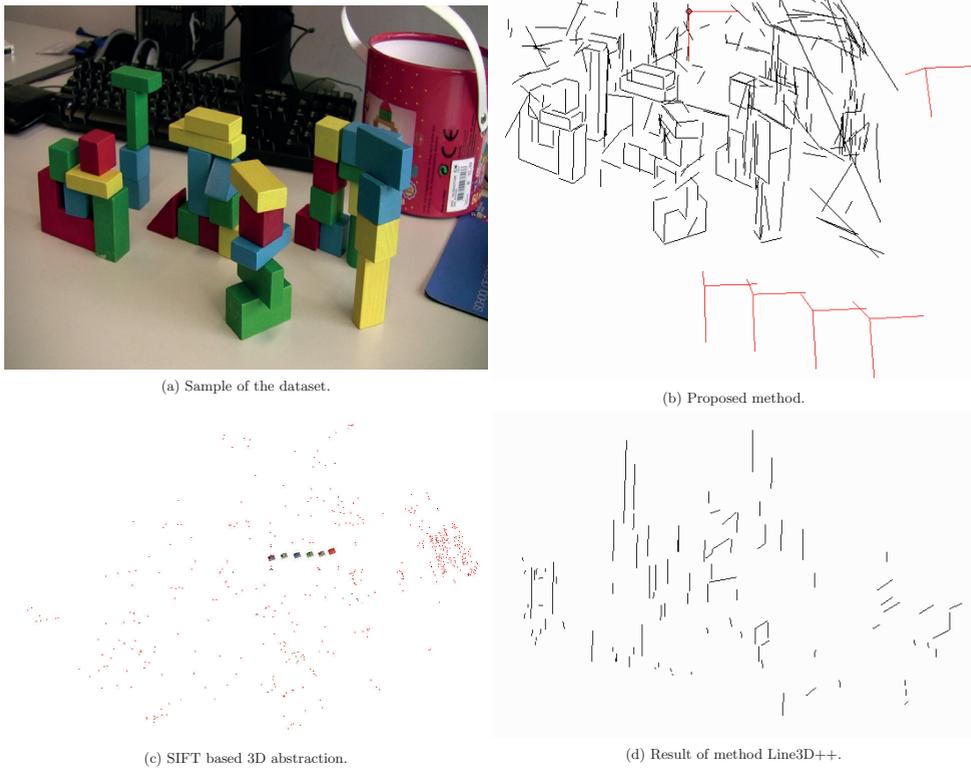
**Figure 6.9:** Quantitative comparison using the sets  $S_{10}$  and  $S_{12}$ [60]. This figure is better viewed on a screen with a 4x zoom. a), b) and c)  $3DwSkT$  method against  $S_{10}$ . The obtained 475 lines have been discretized in points. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the  $3DwSkT$  method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the  $Line3D++$ [55] method. It has been aligned with the Ground Truth mesh. h) to l) Same for the  $3DwSkT$  method against the set  $S_{12}$ , with 556 segments. m) and n) same measurements for the result by  $Line3D++$ [55]. o) shows the histogram for this latter result.



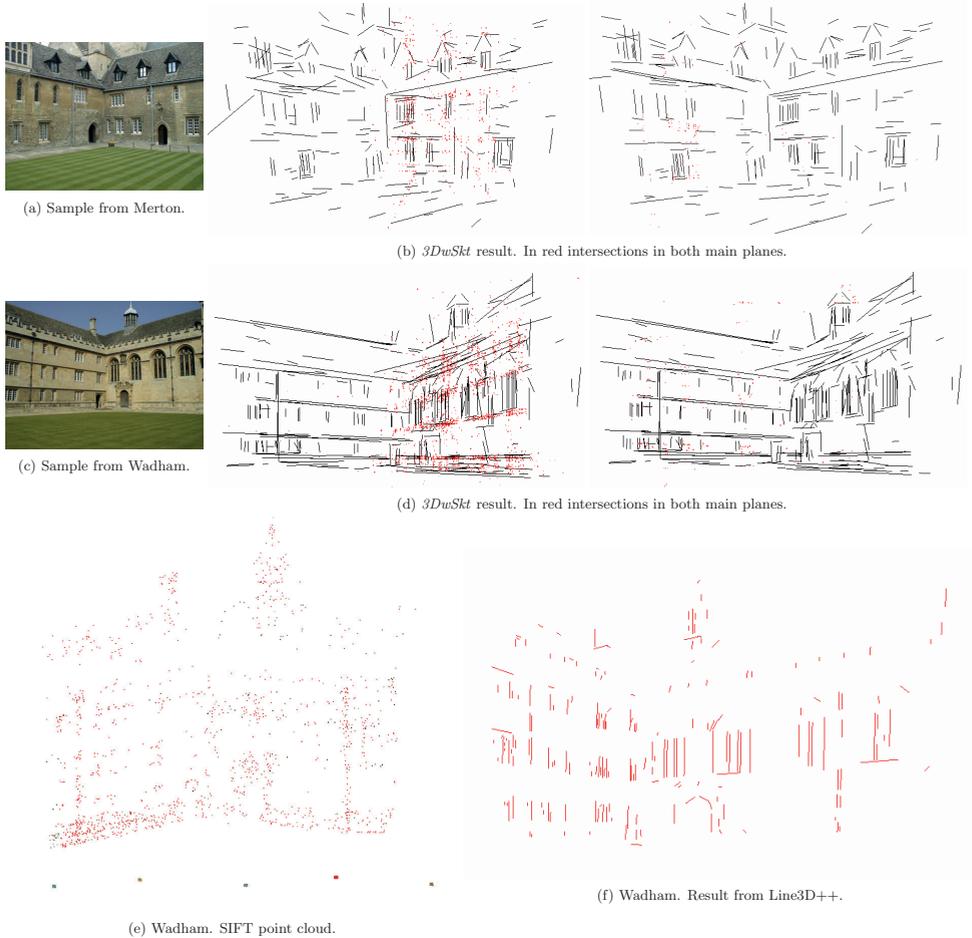
**Figure 6.10:** Qualitative comparison using the public dataset Timber-Frame-House[60]. This figure is better viewed on a screen with a 4x zoom. The used subset comprises image numbers {5-10}, {85-90}, capturing the same corner of the house. a) Result of the *3DwSkt* method, featuring the 12 camera poses and 583 lines. b) *3DwSkt* method. Intersections of the matched lines fitted to the plane with more inliers. c) Same for the next plane with more inliers. d) Point cloud comprising 54041 points by using KAZE[3] point features, used just for retrieving initial estimations of the camera poses for the *3DwSkt* method, looking for a suitable comparison with *Line3D++*[55]. e) SIFT-based reconstruction obtained with VisualSFM[148], featuring 10608 points. f) Line based reconstruction obtained with *Line3D++*[55], built over the result shown in (e). The abstraction features 617 segments, and more accurately located than the *3DwSkt* method.



**Figure 6.11:** a) The result of the method [60] versus a subset of 72 images is provided by the author and shown for reference. b) Computed distances to GT mesh.



**Figure 6.12:** Qualitative comparison using the dataset Building-Blocks [60], of resolution  $1440 \times 1080$ . The used subset comprises the pictures {0,2,4,6,8,10}. a) Sample. b) Result with the *3DwSkt* method, featuring 329 lines and the estimations for the six camera pose. c) On the left, the point based obtained by VisualSfM[148], featuring 3052 points and the 6 camera poses. Based on this point cloud it was generated the result obtained by *Line3D++*[55] that shown on the picture d), with just 102 segments and difficult to understand.



**Figure 6.13:** a) Sample from the dataset Merton College[146], comprised by just 3 pictures. b) Result of *3DwSkt* method, featuring 234 lines. Marked in red the 3D estimations for the intersections of lines that are fitted to the two planes with more inliers. The picture of the left has marked the plane with more RANSAC inliers, and the one on the right the second one. These represent the two planes of the front of the building. For this small dataset the obtained point cloud comprises just 2250 3D points, and is quite sparse and no valid result was obtained by *Line3D++*[55]. c) and d) Same for the Wadham College[146] dataset comprising 5 pictures. e) and f) In this dataset with two more pictures than Merton, the point cloud result of the SIFT-based VisualSfM[148] comprises 3739 3D points. Based on this result, it could generate the result by *Line3D++*[55] on the right, with fewer and shorter lines but located with more accuracy than the *3DwSkt* method.

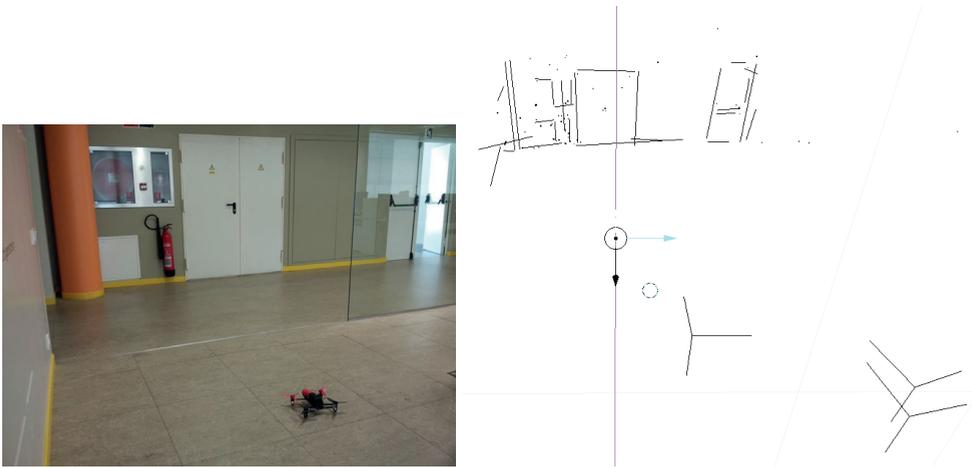
## 6.5 Applications for the 3D sketching method *3DwSkt*

The method *3DwSkt* altogether with the embedded *SALM* are implemented in C++ into a ROS node that includes capturing the streamed video from the commercial drone Parrot Be-bop. This node allowed experimental testing for the whole line detection, matching and 3D sketching method without a human in the loop, nor the requirement of any external image processing pipelines. To our knowledge, there are no other line based reconstruction pipelines completely integrated in ROS. The node was capable of real-time 3D abstraction in a Quad Core 2nd Generation Intel i7 while processing a video streaming with a resolution of  $640 \times 368$  and processing four frames each time. The illumination conditions and number of edges on the image are also crucial for the processing times. The CERES<sup>1</sup> library is used for bundle adjustment.

Fig. 6.14 shows the result of an experiment conducted with the Parrot Be-bop drone for a real-time generation of a 3D sketch of an indoor office. In this case the system fetched four frames of the live video stream while the quadcopter was moving with its camera pointing to the wall. The resulting sketch features lines plotting the main structures on the wall, including doors and pillars.

---

<sup>1</sup><http://ceres-solver.org>



**Figure 6.14:** Experiment with UAV real-time reconstruction indoors, based on lines. The left picture shows the used UAV and the region of interest its camera is pointing to. Four frames of the video stream were processed by *SALM*. The generated sketch on the right hand picture shows the four camera poses and the reconstructed lines.

# Conclusions

The work presented in this thesis goes through the steps followed by the author towards building a straight line based 3D abstraction of a scene or object from pictures. It covers the detection of straight segments in images, the search for their counterparts on other different images, and the 3D estimation for the camera poses and spatial lines.

1. Line segments are extracted in images by means of differential phase congruency in the Gaussian scale-space, which has been proven robust to varying illumination conditions and noise level. The experimental result shows a good performance compared against other methods of the state of the art, achieving a total number of extracted lines very close to the real number of perceived lines in the images.
2. The *SALM* method for two-view matching of straight segments exploits a blend of descriptions of both the individual line appearance and the structure of groups of neighboring segments for finding the counterparts. Its inputs are both the pair of images and the intrinsic parameters of the camera, being the outputs the straight lines matched among the images and possible matching outliers as a measure of the confidence level. The *SALM* method has been evaluated against Ground Truth with public datasets. It has also been quantitatively compared altogether with four other state-of-the-art methods against different man-made scenarios. The presented experimental results show that *SALM* outperforms the competition in the mixed quantitative comparative against the

overall segment matching inlier ratio, by returning relations between longer segments, and by featuring noticeably higher similarity between the length of each segment and its counterpart's. In addition, it was observed lower redundancy and fragmentation of lines compared to the other methods included in the comparative.

3. A novel outliers detection algorithm has been teamed with *SALM*. It is rooted on the hypothesis that geometric relations between coplanar line intersections unveil inconsistencies in the resulting sets of correspondences. The method compares the order of the crossings of the extension of a line segment with the one of their coplanar neighbors. The method has been experimentally proved advantageous by highlighting a large portion of the matching outliers in two datasets, and therefore to reduce the ratio of matching outliers.
4. One of the applications for the line matching method *SALM* has been explored. The 3D abstraction method *3DwSkt* receives as input the camera intrinsic parameters and at least 3 pictures of the scene. It does not require the camera extrinsics estimated from an external SfM pipeline, nor the Ground Truth camera poses. It sources the line correspondences from the method *SALM*, and is able to generate 3D sketches from sets of pictures. It gets an edge against datasets with low number of images, or when these present corrupted texture, blurring, and low definition images where the feature point descriptor fails to detect a fair number of keypoints. The reduced number of correspondences limit the thickness of the point cloud generated by the SfM pipelines, and therefore the accuracy of the estimated camera extrinsics. With inaccurate estimations for the cameras, exploiting homography constraints is not adequate to source line correspondences. Oppositely, *3DwSkt* was able to reconstruct simple line-based sketches with fair precision and number of lines. It required lower number of images to obtain more complete abstractions than the competition. The range of scenarios where it is advantageous to use the *3DwSkt* method for 3D abstraction includes sets of pictures of

simple objects, with low texture, poor illumination, low resolution, blurring or under other conditions that make difficult the success of a point based algorithm. In these scenarios it outperforms the competition in terms of quantity of lines, precision and completeness of the abstraction. Another conclusion is that camera extrinsics are unavoidably required for 3D abstractions featuring many lines, because the estimation for the camera poses will not be accurate if *SALM* returns matching outliers or line fragmentation.

Future work might include a more advanced use of the planes in the line based reconstruction of man-made environments. For the implementation in a desktop or laptop computers, all the code is parallelized for CPU, but in the future, some algorithms will benefit of the use of GPU implementation. On the other hand, an implementation on a Raspberry Pi is a feasible solution for onboard line based 3D reconstruction during UAV flight.



# Bibliography

- [1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building rome in a day. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 72–79. IEEE, 2009.
- [2] Cuneyt Akinlar and Cihan Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.
- [3] Pablo Alcantarilla, Adrien Bartoli, and Andrew Davison. Kaze features. *European Conference on Computer Vision–ECCV 2012*, pages 214–227, 2012.
- [4] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.
- [5] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 328–335, 2014.
- [6] Caroline Baillard, Cordelia Schmid, Andrew Zisserman, and Andrew Fitzgibbon. Automatic line matching and 3d reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, volume 32, pages 69–80, 1999.

- [7] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.
- [8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
- [9] Herbert Bay, Vittorio Ferraris, and Luc Van Gool. Wide-baseline stereo matching with line segments. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 329–336. IEEE, 2005.
- [10] Paul R Beaudet. Rotationally invariant image operators. In *Proc. 4th Int. Joint Conf. Pattern Recog, Tokyo, Japan, 1978*, 1978.
- [11] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4380–4389. IEEE, 2015.
- [12] Djamel Boukerroui, J. Alison Noble, and Michael Brady. On the choice of band-pass quadrature filters. *J. Math. Imaging Vis.*, 21(1):53–80, July 2004.
- [13] Christopher P Bridge. Introduction to the monogenic signal. *arXiv preprint arXiv:1703.09199*, 2017.
- [14] Ümit Budak, Uğur Halıcı, Abdulkadir Şengür, Murat Karabatak, and Yang Xiao. Efficient airport detection using line segment detector and fisher vector representation. *IEEE Geoscience and Remote Sensing Letters*, 13(8):1079–1083, 2016.
- [15] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

- [16] James Cooper, Svetha Venkatesh, and Leslie Kitchen. Early jump-out corner detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):823–828, 1993.
- [17] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 941–947. IEEE, 1999.
- [18] Antonio Criminisi, Ian Reid, and Andrew Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.
- [19] James L Crowley, Patrick Stelmaszyk, Thomas Skordas, and Pierre Puget. Measurement and integration of 3-d structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, 1992.
- [20] Rachid Deriche and Olivier Faugeras. Tracking line segments. *Image and Vision Computing*, 8(4):261–270, 1990.
- [21] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.
- [22] Piotr Dollar, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1964–1971. IEEE, 2006.
- [23] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570, 2015.

- [24] Raquel Dosal, Xosé M. Pardo, Xosé R. Fdez-Vidal, Antón García-Díaz, and Víctor Leborán. A new radial symmetry measure applied to photogrammetry. *Pattern Analysis and Applications*, pages 1–10, 2012.
- [25] Ralph O Dubayah and Jason B Drake. Lidar remote sensing for forestry. *Journal of Forestry*, 98(6):44–46, 2000.
- [26] Ali Elqursh and Ahmed Elgammal. Line-based relative pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3049–3056. IEEE, 2011.
- [27] W.T. Estler, K.L. Edmundson, G.N. Peggs, and D.H. Parker. Large-scale metrology – an update. *CIRP Annals - Manufacturing Technology*, 51(2):587 – 609, 2002.
- [28] A Etemadi. Robust segmentation of edge data. In *Image Processing and its Applications, 1992., International Conference on*, pages 311–314. IET, 1992.
- [29] Bin Fan, Fuchao Wu, and Zhanyi Hu. Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 390–397. IEEE, 2010.
- [30] Bin Fan, Fuchao Wu, and Zhanyi Hu. Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 390–397. IEEE, 2010.
- [31] Bin Fan, Fuchao Wu, and Zhanyi Hu. Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805, 2012.
- [32] Olivier D Faugeras and Francis Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(03):485–508, 1988.

- [33] M. Felsberg. *Low Level Image Processing with the Structure Multivector*. Bericht. Inst. f. Informatik u. Prakt. Mathematik der Christian-Albrechts-Univ., 2002.
- [34] M. Felsberg and G. Sommer. The monogenic scale-space: A unifying approach to phase-based image processing in scale-space. *Journal of Mathematical Imaging and Vision*, 21:5–26, 2003.
- [35] Michael Felsberg and Gerald Sommer. A new extension of linear signal processing for estimating local properties and detecting features. In *Mustererkennung 2000*, pages 195–202. Springer, 2000.
- [36] Michael Felsberg and Gerald Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001.
- [37] Michael Felsberg and Gerald Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, 2001.
- [38] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [39] Pasi Fränti, Eugene I Ageenko, Heikki Kälviäinen, and Saku Kukkonen. Compression of line drawing images using hough transform for exploiting global dependencies. In *Proc. 4th Joint Conf. on Information Sciences (JCIS'98)*, pages 433–436. Citeseer, 1998.
- [40] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

- [41] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.
- [42] Yaroslav Ganin and Victor Lempitsky. N4-fields: Neural network nearest neighbor fields for image transforms. In *Asian Conference on Computer Vision*, pages 536–551. Springer, 2014.
- [43] Andrew Gee and Walterio Mayol-Cuevas. Real-time model-based slam using line segments. *Advances in Visual Computing*, pages 354–363, 2006.
- [44] David Girardeau-Montaut. Cloudcompare (version 2.8) [gpl software]. (2017). retrieved from <http://www.cloudcompare.org/>.
- [45] Claudia I Gonzalez, Patricia Melin, Juan R Castro, Olivia Mendoza, and Oscar Castillo. An improved sobel edge detection method based on generalized type-2 fuzzy logic. *Soft Computing*, 20(2):773–784, 2016.
- [46] R Grompone and J Jakubowicz. Geometry-based unsupervised urban-area detection. *IEEE Geoscience and Remote Sensing Letters*, 2007.
- [47] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.
- [48] Richard Hartley, Rajiv Gupta, and Tom Chang. Stereo from uncalibrated cameras. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 761–764. IEEE, 1992.
- [49] Richard I Hartley. A linear method for reconstruction from lines and points. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 882–887. IEEE, 1995.

- [50] Richard I Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.
- [51] Robert J Hocken and Paulo H Pereira. *Coordinate measuring machines and systems*. CRC Press, 2016.
- [52] W. V. D. Hodge and D. Pedoe. *Methods of Algebraic Geometry*, volume 1 (reprint of the 1947 original). 1994.
- [53] Manuel Hofer, Michael Maurer, and Horst Bischof. Improving sparse 3d models for man-made environments using line-based 3d reconstruction. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 535–542. IEEE, 2014.
- [54] Manuel Hofer, Michael Maurer, and Horst Bischof. Line3d: Efficient 3d scene abstraction for the built environment. In *German Conference on Pattern Recognition*, pages 237–248. Springer, 2015.
- [55] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017.
- [56] Manuel Hofer, Andreas Wendel, and Horst Bischof. Line-based 3d reconstruction of wiry objects. In *18th Computer Vision Winter Workshop*, pages 78–85, 2013.
- [57] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.
- [58] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 473–479. IEEE, 2008.

- [59] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Crisp boundary detection using pointwise mutual information. In *European Conference on Computer Vision (ECCV)*, 2014.
- [60] Arjun Jain, Christian Kurz, Thorsten Thormählen, and Hans-Peter Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segments from images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1586–1593. IEEE, 2010.
- [61] Arjun Jain, Christian Kurz, Thorsten Thormhlen, and Hans-Peter Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segment from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, San Francisco, CA, June 2010.
- [62] Qi Jia, Xinkai Gao, Xin Fan, Zhongxuan Luo, Haojie Li, and Ziyao Chen. Novel coplanar line-points invariants for robust line matching across views. In *European Conference on Computer Vision*, pages 599–611. Springer, 2016.
- [63] Chelhwon Kim and Roberto Manduchi. Planar structures from line correspondences in a manhattan world. In *Asian Conference on Computer Vision*, pages 509–524. Springer, 2014.
- [64] Hyunwoo Kim and Sukhan Lee. Simultaneous line matching and epipolar geometry estimation based on the intersection context of coplanar line pairs. *Pattern Recognition Letters*, 33(10):1349–1363, 2012.
- [65] Les Kitchen and Azriel Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, 1(2):95–102, 1982.
- [66] Josef Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37–42, 1983.

- [67] Thomas Koletschka, Luis Puig, and Kostas Daniilidis. Mevo: Multi-environment stereo visual odometry. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4981–4988. IEEE, 2014.
- [68] Scott Konishi, Alan L. Yuille, James M. Coughlan, and Song Chun Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):57–74, 2003.
- [69] P. Kovesei. Phase congruency: a low-level image invariant. *Psychol Res*, 64(2):136–48, 2000.
- [70] P. D. Kovesei. MATLAB and Octave functions for computer vision and image processing. Available from: <<http://www.peterkovesei.com/matlabfns/>>.
- [71] Peter Kovesei. Phase congruency detects corners and edges. In *The Australian Pattern Recognition Society Conference: DICTA 2003*, 2003.
- [72] Peter Kovesei. Phase congruency detects corners and edges. In *The Australian Pattern Recognition Society Conference: DICTA 2003*, pages 309–318, 2003.
- [73] Ullrich Köthe and Michael Felsberg. Riesz-transforms versus derivatives: On the relationship between the boundary tensor and the energy tensor. In Ron Kimmel, Nir A. Sochen, and Joachim Weickert, editors, *Proceedings Scale-Space '05*, volume 3459 of *Springer LNCS*, pages 179–191. Springer, 2005.
- [74] Kai Li, Jian Yao, Mengsheng Lu, Yuan Heng, Teng Wu, and Yinxuan Li. Line segment matching: A benchmark. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016.
- [75] Kai Li, Jian Yao, Xiaohu Lu, Li Li, and Zhichao Zhang. Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220, 2016.

- [76] Kai Li, Jian Yao, Menghan Xia, and Li Li. Joint point and line segment matching on wide-baseline stereo images. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016.
- [77] Joseph J Lim, C Lawrence Zitnick, and Piotr Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3158–3165, 2013.
- [78] Yuncai Liu and Thomas S Huang. Estimation of rigid body motion using straight line correspondences. *Computer Vision, Graphics, and Image Processing*, 43(1):37–52, 1988.
- [79] Yuncai Liu and Thomas S Huang. A linear algorithm for motion estimation using straight line correspondences. In *Pattern Recognition, 1988, 9th International Conference on*, pages 213–219. IEEE, 1988.
- [80] Juan López, Roi Santos, Xosé R Fdez-Vidal, and Xosé M Pardo. Two-view line matching algorithm based on context and appearance in low-textured images. *Pattern Recognition*, 48(7):2164–2184, 2015.
- [81] Manolis IA Lourakis, Spyros T Halkidis, and Stelios C Orphanoudakis. Matching disparate views of planar surfaces using projective invariants. *Image and Vision Computing*, 18(9):673–683, 2000.
- [82] David G Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The proceedings of the seventh IEEE International Conference on*, volume 2, pages 1150–1157. IEEE, 1999.
- [83] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004.

- [84] Juan López, Roi Santos, Xosé R. Fdez-Vidal, and Xosé M. Pardo. Two-view line matching algorithm based on context and appearance in low-textured images. *Pattern Recognition*, 48(7), 2015.
- [85] Sowmya Mahadevan and David P Casasent. Detection of triple junction parameters in microscope images. In *Proceedings of SPIE*, volume 4387, pages 204–214, 2001.
- [86] Francesco Mancini, Marco Dubbini, Mario Gattelli, Francesco Stecchi, Stefano Fabbri, and Giovanni Gabbianelli. Using unmanned aerial vehicles (uav) for high-resolution reconstruction of topography: The structure from motion approach on coastal environments. *Remote Sensing*, 5(12):6880–6898, 2013.
- [87] David Marr and Ellen Hildreth. Theory of edge detection. *Proc. R. Soc. Lond. B*, 207(1167):187–217, 1980.
- [88] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.
- [89] James H McIntosh and Kathleen M Mutch. Matching straight lines. *Computer Vision, Graphics, and Image Processing*, 43(3):386–408, 1988.
- [90] Gérard Medioni and Ramakant Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31(1):2–18, 1985.
- [91] Branislav Micusik and Jana Kosecka. Piecewise planar city 3d modeling from street view panoramic sequences. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2906–2912. IEEE, 2009.
- [92] Branislav Micusik and Horst Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. *International Journal of Computer Vision*, 124(1):65–79, 2017.

- [93] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.
- [94] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.
- [95] Florica Mindru, Tinne Tuytelaars, Luc Van Gool, and Theo Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94:3–27, 2004.
- [96] P. Montesinos, V. Gouet, and R. Deriche. Differential invariants for color images. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 838–840, 1998.
- [97] Daniel D Morris and Takeo Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Computer Vision, 1998. Sixth International Conference on*, pages 696–702. IEEE, 1998.
- [98] M. C. Morrone and D. C. Burr. Feature detection in human vision: a phase-dependent energy model. In *Proceedings of the Royal Society of London. Series B. Biological Sciences*, volume 235, pages 221–245, 1988.
- [99] M. C. Morrone and R. A. Owens. Feature detection from local energy. *Pattern Recognition Letters*, 6(5):303–313, December 1987.
- [100] Pierre Moulon, Pascal Monasse, Renaud Marlet, and Others. Openmvg. an open multiple view geometry library. <https://github.com/openMVG/openMVG>.

- [101] Marcos Nieto and Luis Salgado. Real-time robust estimation of vanishing points through nonlinear optimization. In *SPIE Photonics Europe*, pages 772402–772402. International Society for Optics and Photonics, 2010.
- [102] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.
- [103] Nati Ofir, Meirav Galun, Sharon Alpert, Achi Brandt, Boaz Nadler, and Ronen Basri. On detection of faint edges in noisy images. *arXiv preprint arXiv:1706.07717*, 2017.
- [104] Kanuengnit Patoommakesorn, Frédéric Vignat, and François Villeneuve. A new straight line matching technique by integration of vision-based image processing. *Procedia CIRP*, 41:777–782, 2016.
- [105] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- [106] G N Peggs, Paul G Maropoulos, E B Hughes, A B Forbes, S Robson, M Ziebart, and B Muralikrishnan. Recent developments in large-scale dimensional metrology. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 223((B6)):571–595, January 2009.
- [107] Pietro Perona and Jitendra Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *In Proceedings 3rd International Conference on Computer Vision*, pages 52–57, 1991.
- [108] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

- [109] A. Zisserman R. Hartley. *Multiple View Geometry in Computer Vision, second ed.* 2004.
- [110] Carolina Raposo, Michel Antunes, and Joao P Barreto. Piecewise-planar stereoscan: structure and motion from plane primitives. In *European Conference on Computer Vision*, pages 48–63. Springer, 2014.
- [111] Guner S Robinson. Color edge detection. *Optical Engineering*, 16(5):165479, 1977.
- [112] Lukas Rosenthaler, Friedrich Heitger, Olaf Kübler, and Rüdiger von der Heydt. Detection of general edges and keypoints. In *European Conference on Computer Vision*, pages 78–86. Springer, 1992.
- [113] Mathias Rothmel, Konrad Wenzel, Dieter Fritsch, and Norbert Haala. Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings LC3D Workshop, Berlin*, volume 8, 2012.
- [114] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.
- [115] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Multiscale line segment detector for robust and accurate sfm. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2000–2005. IEEE, 2016.
- [116] Yohann Salaün, Renaud Marlet, and Pascal Monasse. Line-based robust sfm with little image overlap. In *3D Vision (3DV), 2017 International Conference on*, pages 195–204. IEEE, 2017.
- [117] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

- [118] T Santos, M Moreira, J Almeida, A Dias, A Martins, J Dinis, J Formiga, and E Silva. Plined: Vision-based power lines detection for unmanned aerial vehicles. In *Autonomous Robot Systems and Competitions (ICARSC), 2017 IEEE International Conference on*, pages 253–259. IEEE, 2017.
- [119] C. Schmid and A. Zisserman. Automatic line matching across views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1997.
- [120] Cordelia Schmid and Andrew Zisserman. Automatic line matching across views. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 666–671. IEEE, 1997.
- [121] Cordelia Schmid and Andrew Zisserman. The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–233, 2000.
- [122] H. Shao, T. Svoboda, and L. Van Gool. ZuBuD — Zürich buildings database for image based recognition. Technical Report 260, Computer Vision Laboratory, Swiss Federal Institute of Technology, March 2003.
- [123] Hao Shao, Tomáš Svoboda, and Luc Van Gool. Zubud-zurich buildings database for image based recognition. *Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland, Tech. Rep.*, 260(20):6, 2003.
- [124] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991, 2015.
- [125] Yan Shen and Yuxing Dai. Structure from motion with efficient homography-based line matching. *JOSA A*, 35(2):200–209, 2018.

- [126] Ajit Singh and Michael Shneier. Grey level corner detection: A generalization and a robust real time implementation. *Computer Vision, Graphics, and Image Processing*, 51(1):54–69, 1990.
- [127] Paul Smith, Ian D Reid, and Andrew J Davison. Real-time monocular slam with straight lines. 2006.
- [128] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.
- [129] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Transactions On Graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.
- [130] M. Felsberg G. Sommer. The monogenic scale-space: A unifying approach to phase-based image processing in scale-space. *Journal of Mathematical Imaging and Vision*, 21(1):5–26, 2004.
- [131] Minas E Spetsakis and John Yiannis Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4(3):171–183, 1990.
- [132] Christoph Strecha, Wolfgang Von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. Ieee, 2008.
- [133] Chris Sweeney. Theia multiview geometry library: Tutorial & reference.  
<http://theia-sfm.org>.
- [134] Camillo J Taylor and David J Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, 1995.

- [135] Cihan Topal and Cuneyt Akinlar. Edge drawing: a combined real-time edge and segment detector. *Journal of Visual Communication and Image Representation*, 23(6):862–872, 2012.
- [136] Bill Triggs. Factorization methods for projective structure and motion. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 845–851. IEEE, 1996.
- [137] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer, 1999.
- [138] Bart Verhagen, Radu Timofte, and Luc Van Gool. Scale-invariant line descriptors for wide baseline matching. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 493–500. IEEE, 2014.
- [139] Thierry Vieville and Olivier Faugeras. Feed-forward recovery of motion and structure from a sequence of 2d-lines matches. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 517–520. IEEE, 1990.
- [140] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.
- [141] Luke Wallace, Arko Lucieer, Zbyněk Malenovský, Darren Turner, and Petr Vopěnka. Assessment of forest structure using two uav techniques: A comparison of airborne laser scanning and structure from motion (sfm) point clouds. *Forests*, 7(3):62, 2016.
- [142] Lu Wang, Ulrich Neumann, and Suya You. Wide-baseline image matching using line signatures. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1311–1318. IEEE, 2009.

- [143] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MslD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009.
- [144] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, May 2009.
- [145] Juyang Weng, Thomas S Huang, and Narendra Ahuja. Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336, 1992.
- [146] Tom Werner and Andrew Zisserman. Model selection for automated architectural reconstruction from multiple views. British Machine Vision Conference (BMVC), 2002.
- [147] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064. IEEE, 2011.
- [148] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011.
- [149] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [150] Zezhong Xu, Bok-Suk Shin, and Reinhard Klette. Accurate and robust line segment extraction using minimum entropy with hough transform. *IEEE Transactions on Image Processing*, 24(3):813–822, 2015.
- [151] Shichao Yang and Sebastian Scherer. Direct monocular odometry using points and lines. *arXiv preprint arXiv:1703.06380*, 2017.
- [152] Zhengwei Yang and Fernand S Cohen. Image registration and object recognition using affine invariants and convex hulls. *IEEE Transactions on Image Processing*, 8(7):934–946, 1999.

- [153] Zhengwei Yang, S. Cohen, and Senior Member. Image registration and object recognition using affine invariants and convex hulls. *IEEE Transactions on Image Processing*, 8:934–946, 1999.
- [154] Jiexian Zeng, Liqin Zhan, Xiang Fu, and Binbin Wang. Straight line matching method based on line pairs and feature points. *IET Computer Vision*, 10(5):459–468, 2016.
- [155] Dapeng Zhang and Wei Shu. Two novel characteristics in palmprint verification: datum point invariance and line feature matching. *Pattern Recognition*, 32(4):691–702, 1999.
- [156] Lijun Zhang and Xuexiang Huang. A straight line detection method based on edge following and line segments integration. In *Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on*, pages 297–300. IEEE, 2015.
- [157] Lilian Zhang and Reinhard Koch. Hand-held monocular slam based on line segments. In *2011 Irish Machine Vision and Image Processing Conference*, pages 7–14. IEEE, 2011.
- [158] Lilian Zhang and Reinhard Koch. Line matching using appearance similarities and geometric constraints. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 236–245. Springer, 2012.
- [159] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013.
- [160] Lilian Zhang and Reinhard Koch. Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation*, 25(5):904–915, 2014.

- [161] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.
- [162] Chuhan Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018.

# Summary of the thesis

Implementing environment comprehension into machines is a challenge for mankind. The ability to fetch, classify and interrelate the perceptible elements captured by cameras is the link between our global net of cameras and the Artificial Intelligence. Therefore, the next giant leap will occur when the digital captures of the outside world are observed by neural networks, without requiring a human interpreter and translator. In order to make pictures understandable by machines, these have to be reduced to atomic describable concepts like points, lines or ellipses. The most simple forms, such as points or straight lines, are referred to as primitives. The most traditional techniques detect primitives and classify them attending to their apparent attributes. During these early stages many problems had to be solved, originated by limitations of the digital technology, the similarity between primitives in the same image, the impossibility of characterizing them unequivocally, and the nature of the capture of light with changes on illumination or contrast. These approaches for description and matching of primitives have been developed in parallel with spatial abstraction methods, in such a way that nowadays it is common to derive from a series of pictures an unique 3D representation including estimations for some of the captured primitives with the relative position and orientation of the cameras. This memory is focused on a single kind of primitives: the **straight line segments**. It goes through straight segment matching between images and the other operations that lead to the creation of 3D

representations from these detected primitives.

Straight line segments are frequently found in captures of man-made environments. The inclusion of straight lines in 3D representations provide structural information about the captured shapes and their limits, such as the intersection of planar structures.

The presented memory starts with an introduction. This first chapter goes through the motivation of the author, and a description for every challenge faced during the making of the work that derived to this thesis. The motivation resembles a link between the state of the art and the goals of the work. The introduction is followed by an overview of the background and the state of the art of the covered topics. The different concepts referred to in this thesis are defined in this second chapter. These definitions are the base to understand the rest of this memory. Concepts like primitives and their limits are introduced in the background.

After the background, a review of the related works published in the literature is followed

The first part of the thesis covers **straight line detection methods in images**. Straight lines have to be obtained from the edges in the image. Edge detection methods are the root to obtain primitives or forms in the image, such as points, lines or ellipses. A straight line can be thought as an special case of an edge. The approach employed for edge detection is reviewed in Chapter 4. A straight line segment in 2D is a single primitive that can be defined by the coordinates of its two endpoints, or alternatively by just one point plus the angle and the length of the segment. Straight line detection is commonly rooted on edges detected in the image. A straight line detection algorithm generally works altogether with an edge detector that provides it with an edge skeleton comprised by curvy and straight sections. Every branch in the edge skeleton needs to be chopped into sections and fitted to straight segments. Therefore, in an image, several straight segments can be fitted to a curvy edge. The early and basic detectors of edges in images are gradient based, meaning that the edge detection algorithm marks the points in the image with the highest contrast of pixel values, then draws the edge as a patch through them, adding joints where necessary. The problem with these are that changes in the global contrast will directly affect the edge detection, and

moreover, filters can also affect the shape of edges and joints.

Fragmentation of lines occurs when an uniquely perceived segment entity is detected as an array of shorter segments. The generated problem is that these sections have to be merged into an unique entity before searching for segment matching hypotheses. One approach to minimize fragmentation is to fetch segments in a way resilient to changes in contrast and illumination. It was proved that this can be accomplished by rooting straight segment detection on edges detected in the scale-space and employing the monogenic signal, because the maxima of differential phase congruency resemble the main edges on every picture in the scale-space.

The straight line segment detection algorithm described in this thesis is rooted on an edge detection method that exploits local energy and local phase. This approach has been proved profitable in the literature. For this method, edge detection is performed in several scaled versions of the original image. These downsampled versions of the image are referred to as octaves.

The first hypothesis of the presented line detection method is that a straight segment detection method will profit of detecting edges from the maxima of phase congruency in the scale-space, specially in sets of images with variable illumination conditions. The second hypothesis is that these edges can be reliably fitted to straight line segments by using linear regression, like performed when fitting data to lines in a plot. The third hypothesis is that additional appearance information can be extracted from the scale-space after merging the straight lines detected in different octaves. Therefore lines detected in the original image can be classified according to the depth it was detected in the scale-space, meaning in how many octaves the line segment has been observed. The goal of the method is to input an image, and without additional information, detect solely straight line segments in the whole area of the image, avoiding fragmentation, and resiliently to changes in contrast and illumination. Chapter 5 of the present memory covers **line matching between pairs of images**. The term **matching** in Computer Vision usually refers to the building of relations between elements of

an image or several images. It is a basic tool and the root for many applications in manufacturing industry, robotics or autonomous vehicles. A basic problem in matching of primitives takes two different images showing the same scene, common elements or the same object. Both images may be different in image filtering, illumination, contrast, pose of 2D shapes or objects, different viewpoint or coloring. Additionally, the algorithm has to be provided with a set of detected primitives of the same kind in the images, which may be obtained from a detection algorithm on the images, other physical perception methods, or provided by humans. The challenge of a line matching method is to relate every primitive in one image to its counterpart in the other view.

Matching between edges on a pixel-by-pixel basis works in stereo systems with known epipolar camera geometry. However for the problem of freely moving cameras it is not possible to obtain the exact correspondence for each pixel between images without knowing the extrinsic and intrinsic parameters of the cameras. Representing curvy edges as a set of straight line fragments might not be a good approach for trying to match them. These ample arcs can be detected or decomposed as discrete straight lines that the matching algorithm can handle. Nevertheless, the segment matching might not work properly in this case, because curvy edges and textured edges are expected to be decomposed differently for each different capture of the same object or scene.

The matching method the author is presenting in this thesis is referred to as *SALM*, which is the acronym of *Structure and Appearance Line Matching*. It integrates an outliers removal extension that employs 3D projections to group lines according to their coplanarity. A line matching method comprises a set of algorithms which put in correspondence segments across different images showing common elements, environment or regions of interest. A 3D reconstruction is the result of an estimation of the position for singular primitives captured in several images. The approach followed by *SALM* is framed in the group of line matching methods aimed for 3D reconstruction from pictures of objects built by humans, buildings, urban structures, industrial elements or computer generated models. The *SALM*

line matching algorithm is aimed for application altogether with 3D line based abstraction. It employs an iterative voting algorithm running in groups of lines with the same structural distribution, and the outliers rejection algorithm exploits 3D structures to discriminate potential outliers. *SALM*'s approach is based on three core elements: The first one is that a segment matching method will profit of detecting the segments based on edges obtained from the maxima of phase congruency in the scale-space, specially in sets of images with variable illumination conditions. Secondly, a blend of descriptions of individual line appearance and the structure of groups of neighboring segments is the best approach for finding counterparts of the straight segments detected on different images. The last core element introduces an outliers detection algorithm based on coplanar line intersections of the lines put in correspondence. The inputs for the segment matching method are both the images and the intrinsic parameters of the camera, being the output the relation of matched lines among the images and the potential matching outliers.

Chapter 6 goes through the engineering of a 3D abstraction method based on straight line segments. It employs the *SALM* line matching method for building correspondences between pairs of images. Then it groups the geometric relations and exploits them to generate the 3D sketch. The 3D sketch is a spatial representation based on lines that features estimations for the camera poses and for the 3D straight segments.

The logical evolution of the environment abstraction from multiple views is to incorporate line-based pipelines that do not require a detailed point-based description of the areas of interest. Beside, coplanar line primitives can be intersected to further reveal geometrical information. Likewise, groups of segments will also indicate the location of the most probable vanishing points from a camera plane. These advantages make lines a good candidate to team with point feature detectors and descriptors. They offer the possibility of combining individual similarities of pairs of segments, related to strong constraints of parallelism and orthogonality, and specially coplanarity constraints. This chapter exploits the latter ones, and the potential outliers are determined based on the homography between

different views. The exploited constraint is that the intersection of a pair of coplanar lines, even if it might not resemble a physical point, is still geometrically invariant under perspective projection.

The *3DwSkt* 3D abstraction or sketching method features a set of improvements over the state-of-the-art algorithms based on lines. The contributions of the *3DwSkt* method are aimed for a more suitable 3D abstraction from images featuring low texture, and in scenarios in which it is impossible to generate a dense point cloud:

1. The *3DwSkt* method does not require to be provided with camera poses, nor a dense point cloud obtained from the input pictures. It makes feasible an abstraction based exclusively on line correspondences, and performed independently over pairs of images. Our approach estimates camera extrinsics, but does not root the line matching on these spatial projections, preventing the uncertainty from SfM to propagate and merge with the uncertainty related to 2D line detection and matching. Previous reconstruction methods rely on a third party SfM pipelines, in order to source the camera poses and dense point clouds, to base line matching and 3D reconstruction on.
2. The *Line Observation* is built by merging independent line matchings over pairs of images. The correspondences between segments for each pair of images are obtained with the *SALM* method. The groups of matched 2D lines define unique global entities, and every entity is defined before the 3D stages of reconstruction, in order to avoid the problem of redundancy of 3D lines and to reduce the number of matching outliers when dealing with multiple views. On the other hand, some recent methods qualify the matching candidates based on their support on neighboring views, for later clustering them based on their spatial proximity, instead of performing a global matching of the observed lines individually. This may represent a source of uncertainty if the camera poses provided by the point-based SfM are not accurate

enough, as the matching criteria is tied to the accuracy of every camera pose that was adjusted with point-based SfM, and used as input.

3. The method for exploiting 3D structures described in this memory groups the set of the spatial lines generated by the *3DwSkt* method, attending to coplanarity, by fitting them to different planes through RANSAC. The observed intersections of 2D coplanar lines are therefore described according to the observed matched lines. These segmented group of intersections are projected from every camera plane. They are finally included into the cost function for a second SBA run, taking advantage of this accurate source of observed points in correspondence. Most of the published methods are intended for urban environments, where many lines are coplanar. Nevertheless, they do not retrieve additional information from the images according to the spatial structure. Hence, the projected lines use to be the sole primitive input to the cost function for a least-squares minimization.

The enumerated contributions are intended for spatial abstractions of man made objects and environments. Our approach is fully automatic and only requires a set of pictures or video frames, and camera calibration parameters as the inputs. This complete automatic process will be followed in the different sections of this work. At the end of the Chapter 6, the *3DwSkt* method is experimentally proved, and compared through quantitative and qualitative experiments.

The conclusion of the summary of this thesis follows: The present memory goes through the steps followed by the author towards building a straight line based 3D abstraction of a scene or object from pictures. It covers the detection of straight segments in images, the search for their counterparts on other different images, and the 3D estimation for the camera poses and spatial lines.

1. Line segments are extracted in images by means of differential phase congruency in the Gaussian scale-space, which has been proven robust to varying illumination

conditions and noise level. The experimental result shows a good performance compared against other methods of the state of the art, achieving a total number of extracted lines very close to the real number of perceived lines in the images.

2. The *SALM* method for two-view matching of straight segments exploits a blend of descriptions of both the individual line appearance and the structure of groups of neighboring segments for finding the counterparts. Its inputs are both the pair of images and the intrinsic parameters of the camera, being the outputs the straight lines matched among the images and possible matching outliers as a measure of the confidence level. This line matching algorithm starts by computing individual visual attributes for every sparse segment. The stored values serve to estimate a global rotation, translation or change of scale. The final measure of similarity between segments is computed by letting every correspondence to vote the others within its line neighborhoods.
3. The *SALM* method has been evaluated against Ground Truth with public datasets. It has also been quantitatively compared altogether with four other state-of-the-art methods against different man-made scenarios. The chosen images feature different man-made scenarios, including low texture, high texture with complex structures, changes of illumination, camera viewpoints, global rotations, and scale. The ratio of line matching inliers versus the total number of correspondences returned have been computed for every method.
4. The experimental results presented in this memory show that *SALM* outperforms the competition in the mixed quantitative comparative against the overall segment matching inlier ratio, by returning relations between longer segments, and by featuring noticeably higher similarity between the length of each segment and its counterpart's. In addition, it was observed lower redundancy and fragmentation of lines compared to the other methods included in the comparative.

5. A novel outliers detection algorithm has been teamed with *SALM*. It is rooted on the hypothesis that geometric relations between coplanar line intersections unveil inconsistencies in the resulting sets of correspondences. The method compares the order of the crossings of the extension of a line segment with the one of their coplanar neighbors. The method has been experimentally proved advantageous by highlighting a large portion of the matching outliers in two datasets, and therefore to reduce the ratio of matching outliers.
  
6. One of the applications for the line matching method *SALM* has been explored. The 3D abstraction method *3DwSkt* receives as input the camera intrinsic parameters and at least 3 pictures of the scene. It does not require the camera extrinsics estimated from an external SfM pipeline, nor the Ground Truth camera poses. It sources the line correspondences from the method *SALM*, and is able to generate 3D sketches from sets of pictures. It gets an edge against datasets with low number of images, or when these present corrupted texture, blurring, and low definition images where the feature point descriptor fails to detect a fair number of keypoints. The reduced number of correspondences limit the thickness of the point cloud generated by the SfM pipelines, and therefore the accuracy of the estimated camera extrinsics. With inaccurate estimations for the cameras, exploiting homography constraints is not adequate to source line correspondences. Oppositely, *3DwSkt* was able to reconstruct simple line-based sketches with fair precision and number of lines. It required lower number of images to obtain more complete abstractions than the competition. The range of scenarios where it is advantageous to use the *3DwSkt* method for 3D abstraction includes sets of pictures of simple objects, with low texture, poor illumination, low resolution, blurring or under other conditions that make difficult the success of a point based algorithm. In these scenarios it outperforms the competition in terms of quantity of lines, precision and completeness of the abstraction. Another conclusion is

that camera extrinsics are unavoidably required for 3D abstractions featuring many lines, because the estimation for the camera poses will not be accurate if *SALM* returns matching outliers or line fragmentation.

## Resumo da tese

Implementar a comprensión da contorna nas máquinas é un reto para a humanidade. A habilidade para detectar, clasificar e interrelacionar os elementos perceptibles capturados polas cámaras é o vínculo entre a nosa rede global de cámaras e a Intelixencia Artificial. Polo tanto, o próximo gran salto para a humanidade ocorrerá cando as capturas dixitais do mundo exterior sexan observadas polas redes neuronais, sen necesidade dun intérprete ou tradutor humano. Para que as imaxes poidan ser comprendidas polas máquinas, deben ser primeiro reducidas a conceptos atómicos susceptibles de ser descritos, como puntos, liñas ou elipses. Ás formas máis simples, como os puntos ou liñas rectas, denomínaselles primitivas. As técnicas máis tradicionais detectan as primitivas e clasifícanas atendendo aos seus atributos aparentes. Durante estas primeiras etapas, moitos problemas tivéronse que resolver, orixinados polas limitacións da tecnoloxía dixital, a similitude entre primitivas na mesma imaxe, a imposibilidade de caracterizalas univocamente, e porque os cambios na iluminación e o contraste son inherentes á natureza da captura da luz. Estes modelos para describir e relacionar primitivas desenvolvéronse en paralelo con métodos de abstracción espacial. Desta forma, hoxe é común obter unha representación 3D a partir dunha serie de imaxes, que inclúa estimacións para as posicións espaciais das primitivas, xunto coa posición relativa e orientación das cámaras.

Os segmentos de liña recta atópanse frecuentemente en capturas de contornas construídas

polo home. A inclusión de liñas rectas nas representacións 3D proporciona a esta últimas información estrutural acerca das formas rexistradas nas imaxes, así como dos seus límites. Este é o caso das liñas que forman a intersección de dúas estruturas planas. A memoria presentada comeza cunha **introdución**. Este primeiro capítulo describe as motivacións do autor, e os retos aos que se enfrontou durante a realización do traballo que derivou nesta tese. A motivación representa un vínculo entre a estado da arte e os obxectivos do traballo. A introdución dá paso a unha revisión das bases necesarias para comprender o resto desta memoria. Conceptos como as primitivas e os seus límites son introducidas neste segundo capítulo. O terceiro capítulo é unha **revisión bibliográfica** dos traballos relacionados publicados. O cuarto capítulo é o primeiro dos contidos, e versa sobre a a **detección de liñas** rectas en imaxes. As liñas rectas deben obterse a partir dos bordos detectados na imaxe. Os métodos para a detección destes bordos son a base para obter primitivas ou formas nas imaxes, tales como puntos, liñas ou elipses. Unha liña recta pode, de feito, ser imaxinada como un caso especial dun bordo. O método empregado para detectar bordos descríbese tamén no mesmo capítulo 4. Un segmento de liña recta en 2D é unha primitiva simple que pode ser definida polas coordenadas dos seus dous puntos extremos, ou ben simplemente por un punto, o ángulo e a lonxitude do segmento. A detección de liñas está normalmente baseada en bordos detectados nas imaxes. Un algoritmo de detección de liñas rectas xeralmente funciona xunto a un detector de bordos. Este proporciónalle ao primeiro un esqueleto formado tanto por seccións de liña curvas como rectas, así como por distintos tipos de unións entre bordos. Cada rama do esqueleto de bordos debe ser dividida en seccións. O algoritmo debe tratar de axustar estas seccións a segmentos de liña recta. Polo tanto, nunha imaxe, distintos segmentos de liña recta poden ser axustados a un bordo curvo. Os detectores de bordos en imaxes máis básicos baséanse en gradientes. Isto significa que estes algoritmos de detección de bordos marcan os puntos da imaxe co contraste máis alto entre valores de píxel, e traza o bordo a través destes puntos, engadindo unións onde sexa necesario. O problema con este tipo de detectores é que o seu resultado depende de cambios globais no

contraste da imaxe, e ademais os filtros poden tamén afectar á forma dos bordos e unións. A fragmentación de liñas ocorre cando un segmento detéctase como unha sucesión de segmentos máis curtos aliñados. O problema a resolver é a unión destas seccións nun único segmento, para poder describilo e logo poder buscar aos seus equivalentes noutras imaxes. Unha forma de minimizar a fragmentación é detectar os bordos dunha forma resiliente a cambios en contraste e intensidade na imaxe. Probase que isto pode lograrse baseando a detección de liñas rectas en bordos detectados no espazo escala, e empregando o sinal monogénica, porque o máximo da diferenza de fase diferencial coincide cos bordos principais da imaxe.

O algoritmo de detección de liñas rectas descrito nesta tese está baseado nunha detección de bordos que explota a enerxía e fase locais. Esta vía foi provada beneficiosa na literatura. Para este método, a detección de bordos realízase en diferentes versións escaladas da imaxe orixinal. Estas versións escaladas da imaxe denomínanse oitavas.

A primeira hipótese do método de detección de liñas presentado é que vai obter beneficios de detectar bordos a partir dos máximos da congruencia de fase no espazo escala, especialmente para imaxes que presentan condicións de iluminación variables. A segunda hipótese é que estes bordos poden ser axustados a segmentos de liña recta usando regresión lineal, tal e como se fai ao axustar a unha recta unha dispersión de puntos nunha gráfica. A terceira hipótese é que a información adicional de aparencia pode ser extraída do espazo escala tras unir os segmentos de liña detectadas en diferentes oitavas. Polo tanto, as liñas detectadas na imaxe orixinal poden ser clasificadas atendendo á profundidade á que foron detectadas no devandito espazo escala, ou noutras palabras, ao número de oitavas nas que devandito segmento foi observado. A partires dunha imaxe, e sen información adicional, o obxectivo do método é detectar segmentos de liña recta en todo a área da imaxe, evitando fragmentación e de forma resiliente a cambios en contraste e iluminación. O capítulo 5 da presente memoria abarca a procura de relacións entre pares de imaxes. O termo **matching** en Visión por Ordenador refírese xeralmente a construír relacións entre elementos

homólogos dunha imaxe, ou entre múltiples imaxes. É unha ferramenta básica e a base para moitas aplicacións na industria de fabricación, robótica ou vehículos autónomos. Un problema básico en matching de primitivas toma dúas imaxes distintas mostrando a mesma escena, elementos comúns ou o mesmo obxecto. Ambas imaxes poden ser diferentes en iluminación, contraste, filtrado, cor, pose de formas bidimensionales ou pose da cámara no espazo. Adicionalmente, débesele achegar ao algoritmo as primitivas do mesmo tipo detectadas nas imaxes, que poden ter a súa orixe nun detector de segmentos de liña recta nas imaxes, outros métodos de percepción, ou directamente proporcionadas por un humano. O obxectivo dun método de matching de liñas é relacionar cada primitiva nunha imaxe coa súa homóloga noutra imaxe.

A procura de homólogos entre bordos píxel a píxel funciona ben en sistemas estéreo nos cales se coñece a xeometría epipolar das cámaras. Con todo, para o problema de cámaras que se moven libremente, non é posible obter a correspondencia exacta para cada píxel entre imaxes sen coñecer os parámetros extrínsecos e intrínsecos das cámaras. Polo mesmo motivo, representar bordos curvos como unha serie de fragmentos de liña recta podería non ser o mellor primeiro paso para logo poñelas en correspondencia. Estes arcos amplos deben ser detectados ou descompostos como segmentos de recta discretos que o algoritmo de matching poida describir. Con todo, o matching de segmentos rectos podería non funcionar adecuadamente neste caso, debido a que os bordos curvos e os bordos texturizados son frecuentemente descompostos de forma diferente para cada captura do mesmo obxecto ou escena.

O método de matching que o autor presenta nesta tese denomínase *SALM*, que é o acrónimo de *Structure and Appearance Line Matching*. Integra unha extensión para sinalar posibles erros de matching que se basea en agrupar liñas atendendo á súa coplanaridade. Un método de matching de liñas engloba unha serie de algoritmos que poñen en correspondencia segmentos entre diferentes imaxes que mostren o mesmo contorna, ou elementos e rexións de interese comúns. Unha reconstrución 3D é o resultado da estimación da posición de

primitivas singulares capturadas en distintas imaxes. A aproximación seguida por *SALM* está incluída no grupo de métodos de matching de liñas pensados para a reconstrución 3D a partir de imaxes de zonas urbanas, oficinas, edificios, elementos industriais, modelos xerados por computadora ou obxectos construídos polos humanos. O método para matching de liñas *SALM* está pensado para aplicarse xunto a un método de abstracción 3D baseado en liñas. Emprega un algoritmo iterativo de votación que funciona sobre grupos de liñas coa mesma distribución estrutural, e o algoritmo de selección de potenciais erros de matching explota as estruturas 3D para discriminar potenciais outliers. O método *SALM* baséase en tres premisas básicas: A primeira é que un método de matching de segmentos vai beneficiar de detectar segmentos en base ao máximo da congruencia de fase no espazo escala, especialmente baseado en imaxes con condicións de iluminación variables. En segundo lugar, a unión de descritores para a aparencia de segmentos de liña individuais é a mellor forma de atopar homólogos para segmentos de liña recta detectados en diferentes imaxes. A última premisa básica introduce un algoritmo que busca relacións entre liñas con baixa fiabilidade, baseándose en interseccións das liñas coplanares que foron postas en correspondencia. As entradas para este método de matching son as imaxes e os parámetros intrínsecos da cámara, sendo a saída as liñas relacionadas entre as imaxes e cales delas poderían non ser fiables.

O capítulo 6 abarca a creación dun método de abstracción 3D baseado en segmentos de liña recta. Emprega o método de matching de liñas *SALM* para construír correspondencias entre pares de imaxes. Seguidamente agrupa as relacións xeométricas e explótaas para xerar a abstracción 3D. Esta abstracción 3D é unha representación espacial baseada en liñas que mostra estimacións para as poses das cámaras e para os segmentos de liña 3D.

A evolución lóxica da abstracción da contorna desde múltiples vistas é incorporar solucións baseadas en liñas que non requiran descrições detalladas baseadas en puntos das áreas de interese. Da mesma forma, grupos de segmentos van indicar tamén a localización dos puntos de fuga máis probable para cada plano de cámara. Estas vantaxes convierten ás liñas en bos

candidatos para unirse a detectores de puntos característicos e descriptors. Éstes ofrecen a posibilidade de combinar a similitude individual de pares de segmentos, relacionada coas ligaduras de fortes de paralelismo e ortogonalidade, e especialmente ligaduras de coplanaridade. Este capítulo explota as últimas, e os potenciais outliers determínanse en base á homografía entre diferentes vistas. A ligadura explotada é que a intersección dun par de liñas coplanares, mesmo se non representara un punto físico, segue sendo geoméricamente invariante baixo proxección de perspectiva.

O método de abstracción 3D *3DwSkt* presenta unha serie de achegas sobre os algoritmos da estado da arte baseados en liñas. As contribucións do método *3DwSkt* están pensados para obter unha abstracción 3D máis adecuada desde imaxes que presentan baixa textura, do mesmo xeito que en escenarios nos cales é imposible xerar unha nube densa de puntos a partires dunha solución SfM. O método *3DwSkt* diferénciase da competencia nos seguintes puntos:

1. O método *3DwSkt* non require achegar as poses das cámaras como entrada, nin tampouco unha nube densa de puntos obtida a partires das imaxes de entrada. Ao contrario, é capaz de xerar unha abstracción 3D baseada exclusivamente en correspondencias de liñas obtidas de forma independente sobre pares de imaxes. O método estima os parámetros extrínsecos da cámara a partir das correspondencias entre liñas, no canto de obter estas explotando a homografía entre as cámaras. Isto evita que a incerteza do Structure From Motion propáguese ou se combine coa incerteza carrexada pola detección e posta en correspondencia de liñas 2D. Outros métodos de reconstrución empregan a estimación de cámaras obtida a partir de solucións baseadas en puntos característicos de terceiros. Estas solucións son capaces de estimar as poses da cámara e a nubes densa de puntos.
2. As **correspondencias entre segmentos** obtense para cada par de imaxes co método *SALM*. Cada liña 2D relacionada en múltiples vistas define unha entidade global

única. Cada unha destas entidades defínese antes das etapas de reconstrución 3D, para evitar que aparezan redundancias na forma de liñas 3D repetidas, e para reducir o número de erros ao sinalar aos homólogos. Doutra banda, algúns métodos recentes clasifican aos segmentos candidatos a homólogos en función da súa proxección sobre os planos de cámaras veciñas, para logo agrupalos en función da súa proximidade espacial, en lugar de realizar primeiro unha comparación global das liñas observadas individualmente. Isto pode representar unha fonte de incerteza se as poses das cámaras proporcionadas polo SfM baseado en puntos non son o suficientemente precisas, xa que os criterios para a posta en correspondencia están ligados á exactitude de cada pose de cámara que se axustou co SfM baseado en puntos.

3. O método para explotar estruturas 3D descrito nesta memoria agrupa as liñas espaciais xeradas polo método *3DwSkt*, atendendo á súa coplanaridade. Os grupos créanse tras axustar as liñas 3D a diferentes planos, empregando RANSAC como xerador de hipóteses. Unha vez creados estes grupos, as liñas 2D coplanares que conteñen interseccións en cada unha das imaxes orixinais onde foran postas en correspondencia. Os grupos de interseccións creados proxéctanse dende cada plano de cámara. Finalmente, os puntos 3D inclúense na función de custo para un segundo axuste por mínimos cadrados. A maioría dos métodos de abstracción 3D baseados en liñas publicados están destinados a contornas urbanas, onde abundan as liñas coplanares. Con todo, a meirande parte deles non chegan a obter información adicional das imaxes de acordo coa estrutura espacial.

As contribucións enumeradas están pensadas para abstraccións de obxectos e contornas construídos polo home. A nosa técnica é completamente automática e só require achegarlle un conxunto de imaxes ou fotogramas de vídeo, xunto cos parámetros intrínsecos de calibración das cámaras. Este proceso completo e automático detállase nas diferentes

seccións deste traballo. Ao final do capítulo 6, o método *3DwSkt* próbase experimentalmente, e compárase mediante probas cuantitativas e cualitativas.

A conclusión deste resumo da tese é a seguinte: A presente memoria explica os pasos seguidos polo autor co obxectivo de crear un método de abstracción 3D dunha escena ou obxecto a partir de imaxes. A tese cobre a detección de segmentos de liña recta en imaxes, a procura dos seus homólogos noutras imaxes, e a estimación da pose 3D para as cámaras e liñas espaciais.

1. Os segmentos de liña extráense en imaxes baseándose na congruencia de fase diferencial no espazo-escala Gaussiano. A robustez desta técnica foi probada contra cambios en iluminación e nivel de ruído. Os resultados experimentais mostran un número total de liñas detectadas próximo ao número de liñas percibidas por un humano nas imaxes.
2. O método *SALM* para a posta en correspondencia de segmentos de liña recta explota unha mestura de descrições da aparencia individual das liñas e da estrutura dos grupos de segmentos veciños aos que pertence. As entradas do método son un par de imaxes e os parámetros intrínsecos da cámara, sendo as saídas a relación de liñas en correspondencia entre ambas imaxes, e tamén indica cales destas teñen un nivel de fiabilidade baixo. Este algoritmo de matching comeza calculando os atributos visuais individuais de cada segmento por separado. O conxunto de todos os atributos observados nas liñas dunha imaxe serven para estimar unha rotación, translación ou cambio de escala. A medida final de similitude entre segmentos calcúlase deixando que cada correspondencia entre liñas vote ás posibles correspondencias das liñas veciñas.
3. O método *SALM* foi avaliado contra Ground Truth en bases de imaxes públicas. Tamén se comparou cuantitativamente con outros catro métodos da estado da arte para

matching de liñas. As imaxes elixidas mostran diferentes escenas creadas polo home, que inclúen dende baixa textura ata alta textura, estruturas complexas, cambios de iluminación, punto de vista de cámaras, rotacións globais ou cambios de escala. A fracción de acertos dos distintos métodos para matching de liñas foi calculada e comparada co número total de correspondencias devoltas.

4. Os resultados experimentais presentados nesta memoria mostran que o método *SALM* obtén unha mellor fracción de acertos que a competencia na comparativa mixta, devolvendo relacións entre segmentos máis longos e presentando estes menor variabilidade de lonxitudes en comparación co seu homólogo. En segundo lugar, observouse unha menor redundancia e fragmentación de liñas en comparación con outros métodos incluídos na comparativa.
5. Un novo algoritmo para detección de erros na posta en correspondencia de liñas conxúntase con *SALM*. Este algoritmo está baseado na hipótese de que as relacións xeométricas entre liñas coplanares e as súas interseccións poden revelar inconsistencias no grupo de correspondencias. O método compara a orde na cal unha liña interseca outras veciñas e coplanares, coa orde na cal se intersecan as súas homólogas na outra imaxe. Este método foi probado experimentalmente en dous dataset, resultando vantaxoso ao sinalar unha parte importante das correspondencias erróneas devoltas polo algoritmo de matching. Como conclusión, o algoritmo serve para reducir a fracción de erros na posta en correspondencia de liñas entre diferentes imaxes.
6. Unha das aplicacións do método de posta en correspondencia de liñas *SALM* describiuse no capítulo 6. O método de abstracción 3D recibe como entrada os parámetros intrínsecos das cámaras e, polo menos, 3 imaxes da mesma escena. Non require que os parámetros extrínsecos das cámaras fosen previamente estimados mediante unha solución externa SfM, nin tampouco as poses das cámaras obtidas por

outros medios. O método obtén as correspondencias de liñas de *SALM* e é capaz de xerar abstraccións 3D formadas por liñas a partir de conxuntos de imaxes. O método obtén unha vantaxe fronte á competencia en grupos de poucas imaxes, ou ben cando estas imaxes presentan textura corrupta, blurring e baixa definición. Nestes escenarios, os descritores de puntos característicos non logran relacionar un número de puntos elevado. Este número de puntos escaso limita a densidade da nube de puntos 3D obtida polas solucións SfM, e polo tanto a precisión da estimación para as poses das cámaras. As estimacións imprecisas para as cámaras dificultan a explotación das ligaduras da homografía para obter as correspondencias entre as liñas detectadas nas imaxes. Pola contra, nos mesmos escenarios *3DwSkt* foi capaz de crear abstraccións cunha precisión e número de liñas aceptable. Un menor número de imaxes é requirido para obter abstraccións máis completas que a competencia. O rango de escenarios nos cales é vantaxoso empregar *3DwSkt* para abstracción 3D inclúe conxuntos de capturas de obxectos simples, con baixa textura, iluminación deficiente, baixa resolución, blurring ou outras condicións que fagan difícil para unha solución SfM a creación de numerosas relacións entre puntos característicos. Nos devanditos escenarios *3DwSkt* mellora os resultados da competencia respecto á cantidade de liñas, precisión e completitude da abstracción. Outra conclusión é que os parámetros extrínsecos das cámaras son inevitablemente requiridos para crear abstraccións 3D que amosen múltiples liñas, porque a estimación das poses das cámaras non será precisa se *SALM* devolve correspondencias erróneas entre as liñas, ou ben fragmentación de liñas.

# List of Figures

1.1	Gradient-based edge detectors lead to false edges in regions with changes of illumination or noise. . . . .	8
1.2	Example of line matching using the method described in this thesis, shown to highlight the difficulties due to line occlusions on a viewpoint change. Segment 6, colored in magenta, is mismatched due to occlusion by the limits of the frame. Segment 17, colored in cyan, is mismatched due occlusion by viewpoint change. Images from the dataset [74]. . . . .	9
1.3	Classification of scenes employed for quantitative experiments. Taken from public datasets[29, 123, 74, 61] . . . . .	11
1.4	Example of line matching using the method described in this thesis against pictures of the dataset [74]. The high density of short segments in the images raises the need of specific rules for discerning correct correspondences using the Ground Truth human assessment. .	12
1.5	On the top, examples of kinds of neighborhoods, defining the structure of groups of line segments. On the bottom a simplified visual representation showing how the orange line was not detected in the third image, and therefore the description of this neighborhood can just be used with the first two pictures . . . . .	14

1.6	On the top, examples of individual appearance attributes that serve to distinguish sparse segments independently of their neighboring lines. On the bottom, a dataset featuring different zooms and rotations between images. It is an example for which <b>the slope and length of the segments are transformed differently</b> and hence have to be compared separately <b>for each possible pair of images of the set</b> . . . . .	15
1.7	Representation of the challenge of converting a set of 4 pictures into a 3D sketch featuring the line segments and camera axis. . . . .	16
2.1	Definitions for the main concepts included in this thesis, in order of complexity. . . . .	25
2.2	Real-time generation of 3D abstraction . . . . .	29
2.3	Incremental addition of cameras to the 3D abstraction [133]. . . . .	30
4.1	Representation of the different steps of the line detection method described in this chapter. . . . .	49
4.13	The variation of the total number of lines extracted by the methods for the same real image with different average gray levels. . . . .	76
5.1	Diagram of blocks for the SALM method. The upper region represents the integrated line detection method disclosed in Chapter 2. . . . .	82
5.2	Computation of the similarity measure for each pair of lines. The total similarity between two lines (equation 5.22) depends on the structural similarity of their neighborhoods (equation 5.20), which integrates the line-to-line similarity (equation 5.1) between the neighboring segments. . . . .	83
5.3	Local region $R_i$ defined for a line segment $i$ , consisting of $n_l \times n_r$ samples uniformly distributed along the segment, with $n_r = 61$ and $n_l = 100$ . Examples of these regions for three different lines on a real image. . . . .	85
5.4	Computation of the individual similarity for the line segments, in the <i>SALM</i> method. . . . .	85

5.5 Histogram of the component  $d_l$  of the similarity vector, showing the change of length of the matched lines for an image pair. The dominant value in the histogram corresponds to the dominant scale transformation between the two images. . . . . 87

5.6 Classification of the different kinds of neighborhoods for a line  $i$  attending to the relative pose of its neighboring lines  $i_j$ . . . . . 93

5.7 Example of a real situation which illustrates the need of several line neighborhoods. The images show three simultaneously coexistent neighborhoods. The image on the left shows the original region of the low-textured scene, in which the six central lines are very similar. The middle left and middle right images show line neighborhoods of the type 'Three on its right' for two different lines. Both neighborhoods are too similar to be discerned by the algorithm when matching to a different view. Therefore, more structural information has to be provided in order to distinguish the counterparts of both segments in other image. In the right hand image, another kind of neighborhood ('Head and Tail') is represented for both lines. Note that this latter neighborhood solves the disambiguation, meaning that the votes of the segments in these neighborhoods will produce different scores that will be used to distinguish both segments. . . . . 93

5.8 Process of neighborhood computation for the SALM method. . . . . 95

5.9 Image on the left: example of the convex hull (shaded region) from a line neighborhood for a real image. In the center: representation of the convex hull of the line neighborhood  $L = \{i_1, i_2, i_3\}$  of the line  $i$ . Image on the right: construction example of an affine invariant vector  $\Omega_k^B$ , which is obtained by taking the ratio of the areas of the four triangles  $\{k, k_{+1}, k_{+2}\}$ ,  $\{k_{+1}, k_{+2}, k_{+3}\}$ ,  $\{k_{+2}, k_{+3}, k\}$ ,  $\{k_{+3}, k, k_{+1}\}$  to the area of the quadrangle  $\{k, k_{+1}, k_{+2}, k_{+3}\}$ . . . . . 95

5.10 Graphical representation for the computation of the similarity of neighborhoods, the integration with the individual similarity to obtain the total similarity, and last steps towards obtaining the final segment matchings. . . . . 98

- 5.13 Graphical representation for the algorithm that exploits planes to obtain possible line matching outliers. This algorithm can be run after a line matching process. . . . . 103
- 5.11 Flow chart embedding both the line detection algorithm and *SALM*. . . . . 104
- 5.12 Visual example of the k-Nearest-Neighbors search for close coplanar intersections. . . . . 105
- 5.14 On the top, three pairs of industrial images. The rest of the figure shows the matching results by the method *SALM* for the three pairs of images. The coloring of the line segments corresponds to the human ground truth: Black lines are correct matches, red lines are wrong matches, and blue lines are non-significant lines. . . . . 112
- 5.15 On the top, three pairs of images from public datasets[30][122]. The rest of the figure shows the matching results by the method *SALM* for the three pairs of images. The coloring of the line segments corresponds to the human ground truth: Black lines are correct matches, red lines are wrong matches, and blue lines are non-significant lines. . . . 113
- 5.16 Examples of the search for outliers using coplanar neighbor intersections. . . . . 115
- 6.1 Flow for the *3DwSkt* method. The *SALM* method is integrated into the upper *2D layer*. . . 126
- 6.2 Example of merging of two groups of counterparts. . . . . 128
- 6.3 Graphic representation of the *3D abstraction layer* of the method. . . . . 130
- 6.4 The Plücker coordinates for the line  $\Gamma_j$  are  $(\mathbf{u}_j, \mathbf{t}_j \times \mathbf{u}_j)$ , taking any point  $\mathbf{d}$  on the line, and  $\mathbf{l}$  being the unit vector in the direction of the difference between its endpoints . . . . 136
- 6.5 In order to avoid the singularity around the center of the image plane, the error function is chosen as the squared shortest distance from the observed segment endpoints to the reprojected infinite line, as suggested by[160] . . . . . 136

- 6.6 The image on the top left left is a graphic explanation for the usage of coplanarity conditions. Just the intersections of matched lines that are coplanar are counted (line 1 intersects 2 and 3, but not the yellow line under the "L" label). The pair of images placed in the top right shows the result of the method *SALM* for line matching between two images of the dataset "CUBE"[134]. The intersections of coplanar segments extracted by the algorithm are highlighted in green. Note that just the intersections of coplanar segments were highlighted. The pair of images on the bottom of the figure are from the 3D sketch generated from the pictures {3,4,5,6} of the dataset. These shows the segmentation of endpoints after RANSAC. The planes with more inliers feature 68 and 39 inliers respectively. 138
- 6.7 Comparative results using the *3DwSkt* method of line-based reconstruction exclusively based on lines. The selected subset from the dataset "BOX"[134] comprises the picture numbers {3-6}. a) Sample from the dataset. b) Result obtained with the *3DwSkt* method. Two different views of the sketch are plotted, in order to show that the proportions are correct. c) A SIFT-based reconstruction performed with VisualSfM[148] returned an sparse cloud, and therefore this result of *Line3D++*[55] presents just 26 unconnected 3D lines, and it is not understandable. . . . . 146
- 6.8 Quantitative comparison using the sets  $S_6$  and  $S_8$ [60]. This figure is better viewed on a screen with a 4x zoom. a) Sample of the set. b) and c) *3DwSkt* method against  $S_6$ , resulting 175 lines. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the *3DwSkt* method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the *Line3D++*[55] method. It has been aligned with the Ground Truth mesh. h) to l) The *3DwSkt* method against the set  $S_8$ , with 294 segments. m) and n) same measurements for the result by *Line3D++*[55]. o) shows the histogram for this latter result. 151

- 6.9 Quantitative comparison using the sets  $S_{10}$  and  $S_{12}$ [60]. This figure is better viewed on a screen with a 4x zoom. a), b) and c) *3DwSkt* method against  $S_{10}$ . The obtained 475 lines have been discretized in points. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the *3DwSkt* method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the *Line3D++*[55] method. It has been aligned with the Ground Truth mesh. h) to l) Same for the *3DwSkt* method against the set  $S_{12}$ , with 556 segments. m) and n) same measurements for the result by *Line3D++*[55]. o) shows the histogram for this latter result. . . . . 152
- 6.10 Qualitative comparison using the public dataset Timber-Frame-House[60]. This figure is better viewed on a screen with a 4x zoom. The used subset comprises image numbers {5-10},{85-90}, capturing the same corner of the house. a) Result of the *3DwSkt* method, featuring the 12 camera poses and 583 lines. b) *3DwSkt* method. Intersections of the matched lines fitted to the plane with more inliers. c) Same for the next plane with more inliers. d) Point cloud comprising 54041 points by using KAZE[3] point features, used just for retrieving initial estimations of the camera poses for the *3DwSkt* method, looking for a suitable comparison with *Line3D++*[55]. e) SIFT-based reconstruction obtained with VisualSFM[148], featuring 10608 points. f) Line based reconstruction obtained with *Line3D++*[55], built over the result shown in (e). The abstraction features 617 segments, and more accurately located than the *3DwSkt* method. . . . . 153
- 6.11 a) The result of the method [60] versus a subset of 72 images is provided by the author and shown for reference. b) Computed distances to GT mesh. . . . . 154

6.12 Qualitative comparison using the dataset Building-Blocks [60], of resolution  $1440 \times 1080$ . The used subset comprises the pictures {0,2,4,6,8,10}. a) Sample. b) Result with the *3DwSkt* method, featuring 329 lines and the estimations for the six camera pose. c) On the left, the point based obtained by VisualSfM[148], featuring 3052 points and the 6 camera poses. Based on this point cloud it was generated the result obtained by *Line3D++*[55] that shown on the picture d), with just 102 segments and difficult to understand. . . . . 155

6.13 a) Sample from the dataset Merton College[146], comprised by just 3 pictures. b) Result of *3DwSkt* method, featuring 234 lines. Marked in red the 3D estimations for the intersections of lines that are fitted to the two planes with more inliers. The picture of the left has marked the plane with more RANSAC inliers, and the one on the right the second one. These represent the two planes of the front of the building. For this small dataset the obtained point cloud comprises just 2250 3D points, and is quite sparse and no valid result was obtained by *Line3D++*[55]. c) and d) Same for the Wadham College[146] dataset comprising 5 pictures. e) and f) In this dataset with two more pictures than Merton, the point cloud result of the SIFT-based VisualSfM[148] comprises 3739 3D points. Based on this result, it could generated the result by *Line3D++*[55] on the right, with fewer and shorter lines but located with more accuracy than the *3DwSkt* method. . . . . 156

6.14 Experiment with UAV real-time reconstruction indoors, based on lines. The left picture shows the used UAV and the region of interest its camera is pointing to. Four frames of the video stream were processed by *SALM*. The generated sketch on the right hand picture shows the four camera poses and the reconstructed lines. . . . . 158

# List of Tables

0.1	Table of Notations used in this thesis . . . . .	1
0.2	Table of acronyms used in this thesis . . . . .	1
4.1	This table follows the process of straight line detection in an image, and it is better viewed in a computer monitor with a 400%. a) Original image. b) Marked in black, the points of differential phase congruency. c) Points are filtered using a threshold attending to the Phase Congruency, length and inclination. d) The remainder of points are fitted to straight lines by linear regression. e) After the fusion algorithm for fragments of segments. . . . .	51
5.1	Quantitative evaluation of methods for matching of lines across two images . . . . .	107
5.2	Number of line matchings given by the methods on a set of image pairs of unrelated views. The images of each pair have been randomly selected from industrial and public[30, 122] datasets. . . . .	110
5.3	Average line matching accuracy and processing times from the results shown in table 5.1 . . . . .	111
5.4	<i>SALM</i> method. Average line matching accuracy and processing times from the results shown in table 5.1 . . . . .	114
5.5	Quantitative evaluation of methods for matching of lines against the dataset "Castle". The images are recommended to be displayed in a computer monitor with a 500% zoom. . . . .	116

5.6 Quantitative evaluation of methods for matching of lines against the dataset "Low Texture". The images are recommended to be displayed in a computer monitor with a 500% zoom. . . . . 117

5.7 Quantitative evaluation of methods for matching of lines against the dataset "Textureless corridor". The images are recommended to be displayed in a computer monitor with a 500% zoom. . . . . 118

5.8 Quantitative evaluation of methods for matching of lines against the dataset "Outdoor light". The images are recommended to be displayed in a computer monitor with a 500% zoom. . . . . 119

5.9 Quantitative evaluation of methods for matching of lines against the dataset "Leuven". The images are recommended to be displayed in a computer monitor with a 500% zoom. . 120

5.10 Quantitative evaluation of methods for matching of lines against the dataset "Drawer". The images are recommended to be displayed in a computer monitor with a 500% zoom. . 121

6.1 Overview of evaluated methods . . . . . 144

6.2 Quantitative comparison of both methods against the subsets  $S_6$ ,  $S_8$ ,  $S_{10}$  and  $S_{12}$  created from the public dataset Timber-Frame-House[60] . . . . . 150