Image-Based Floor Segmentation in Visual Inertial Navigation

GUILLEM CASAS BARCELÓ

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden November 2012

XR-EE-SB 2012:019

Abstract

Floor segmentation is a challenging problem in image processing. It has a wide range of applications in the engineering field. In mobile robot navigation systems, detecting which pixels belong to the floor is crucial for guiding the robot within an environment, defining the geometry of the scene, or avoiding obstacles.

This report presents a floor segmentation algorithm for indoor scenarios that works with single grey-scale images. The portion of the floor closest to the camera is segmented by judiciously joining a set of horizontal and vertical lines, previously detected. Unlike similar methods in the literature, it does not rely on computing the vanishing point and, thus, it adapts faster to changes in camera motion and is not restricted to typical corridor scenes.

A second contribution of this thesis project is the moving features detection for points within the segmented floor area. Based on the camera ego-motion, the expected motion of the points on the ground plane is computed and used for rejecting feature points that belong to movable obstacles. A key point of the designed method is its ability to deal with general motion of the camera.

The implemented techniques are to be integrated in a visual-aided inertial navigation system (INS) that combines visual and inertial information. This INS requires a certain number of feature point correspondences on the ground plane to correct data from an inertial measurement unit (IMU) and estimate the ego-motion of the camera. Hence, segmenting the floor region and detecting movable features become relevant tasks in order to ensure that the considered features do belong to the ground.

Sammanfattning

Att segmentera golvet är ett utmanande problem i bildbehandling. Det har ett brett tillämpningsområde inom ingenjörsvetenskapen. I navigeringssystem för mobila robotar är detektering av vilka pixlar som tillhör golvet avgörande för att styra roboten i en inomhusmiljö, för att definiera geometrin för scenen, eller för att undvika hinder.

Denna rapport presenterar en golvsegmenteringsalgoritm för inomhustillämpningar utifrån enstaka gråskalebilder. Golvytan närmast kameran segmenteras genom att på ett väl underbyggt sätt sammanknyta horisontella och vertikala linjer som tidigare detekterats. Till skillnad från liknande metoder i litteraturen är metoden inte beroende av att skatta den så kallade "vanishing point", därigenom anpassar den sig snabbare till förändringar i kamerans rörelse och är inte begränsad till typiska korridorsscener.

Ett ytterligare bidrag i detta examensarbete är en metod för att detektera objektpunkter som rör sig inom den segmenterade golvytan. Baserat på kamerans rörelse, beräknas den förväntade rörelsen hos punkterna på golvet och den används för att förkasta punkter som tillhör rörliga hinder. En viktig egenart hos den designade metoden är dess förmåga att hantera en godtycklig kamerarörelse.

De implementerade metoderna ska integreras i ett visuellt tröghetsnavigeringssystem som kombinerar visuell-och tröghetsinformation. Detta system kräver ett visst antal punktkorrespondenser på golvet för att korrigera data från en tröghetsmätningsenhet och uppskatta kamerans rörelse. Att segmentera golvyta och detektera rörliga särdrag, blir därför relevanta uppgifter fär att säkerställa att de använda punkterna tillhör golvet.

Resum

Segmentar el terra és un problema desafiant en processat d'imatge i té una gran varietat d'aplicacions. Detectar quins píxels pertanyen al terra és crucial en sistemes de navegació robot per a guiar el robot en l'entorn, definir la geometria de l'escena o eludir obstacles.

Aquest treball presenta un algorisme per a segmentar el terra en escenes d'interiors que treballa amb imatges en escala de grisos. La part del terra que està més a prop de la càmera és segmentada a partir d'unir judiciosament línies verticals i horitzontals, prèviament detectades. A diferència d'altres mètodes similars que es poden trobar en la literatura, la tècnica proposada no es basa en calcular el punt de fuga i conseqüentment, s'adapta més ràpidament als canvis en el moviment de la càmera i no està restringida a escenes típiques de passadissos.

Una segona contribució d'aquesta tesis és la detecció de punts característics en moviment que queden dins de l'àrea definida com a terra. A partir del moviment propi de la càmera, es calcula el moviment esperat del punts en el pla de terra, el qual s'utilitza per a descartar punts característics que formen part d'objectes en moviment. Un punt clau del mètode dissenyat és l'habilitat de treballar per a qualsevol tipus de moviment de la càmera.

Les tècniques implementades han estat dissenyades a fi de ser integrades en un sistema de navegació basat en imatge que combina informació visual i inercial. Aquest sistema de navegació requereix un cert número de punts característics situats al pla de terra per a corregir dades d'una unitat de mesura d'inèrcia i estimar el moviment propi de la càmera. Per tant, segmentar la regió del terra i detectar punts característics en moviment són tasques importants per a assegurar-se que els punts característics seleccionats pertanyen realment al terra.

Acknowledgements

There are a lot of people who have collaborated in this project, people with whom I have shared my time and who have suffered me all along this 9 months trip (yeap, like a pregnancy). Not only people, but also places, situations, feelings, etc. have been very important to bring me at the current point. Thanks to all in advance. I did not want to fall into clichés while writing these words, but I guess I am not able to get rid of them.

First of all, I would like to gratefully thank my examiner Prof. Magnus Jansson and my supervisor Ghazaleh Panahandeh. Magnus rescue me when I was sailing in a very confusing sea of Master Thesis proposals and put me in contact with Ghazaleh who offered me the opportunity to work with this amazing project. Thanks to him for answering all my mails and for his useful and valuable advices and feedbacks. I am thankful to Ghazaleh for guiding me and for her help and advices along these months. Both of you have decisively contributed to what hopefully is going to become my first publication.

To my family, for their long-distance encourage and availability to help with everything. I do know it has been not easy at all for you to have me far away from home for two years. I have also miss you, but I guess you can imagine how much have I enjoyed these 24 months in Sweden, so the remoteness has been completely worthy.

I would not like to start mentioning now the names of all friends who have somehow collaborated in this experience because not only would I miss some of you, but also I would not have enough space. I just want to let you know that I feel very proud of having you as friends, both the ones from my hometown and the ones I met in Stockholm. Among all the people, please let me specially thank Elena: my confidant, roommate and girlfriend. Sorry, but it would be hypocritical if I did not say it. You have been my main support these last months and regarding the project, you have been my severest reviewer. You know the Thesis, at least, as much as I do.

I really wish the best of the lucks to all of you.

Cheers! Skål! Salut!

Contents

List of Figures

Chapter 1

Introduction

In this introductory chapter, the thesis project and the report organization are described. First of all, an overview of the subject and the most prominent related work are provided so that the reader can grasp the context of the study and follow the rest of the thesis. After that, the proposed method is sketched stressing its strengths and contributions. Next, the adopted methodology and the prime material used are discussed. Finally, an outline of the thesis report, highlighting its structure, is presented.

1.1. Motivation

Robot navigation systems have been a cutting-edge research field during the past decades. One of the most promising challenges in robotics is the integration of mobile devices in indoor environments [1], [2]. In order to carry out its responsibilities, the robot must be able to navigate in the environment autonomously [3].

Historically, most of the approaches have been based on devices like laser range finders or stereo rigs. Not only are these systems very expensive, but also their setup and configuration difficult to manage [4]. Hence, visual-based techniques have increased their presence in robot navigation systems along the past years due to their simplicity and low cost. For the same reason, stereo systems are progressively migrating to approaches based on a single camera.

Vision-based indoor mobile robot navigation systems need to recognize the structure of the scene and avoid both static and movable obstacles. Segmenting the floor and detecting moving objects become significant tasks for guiding the robot within an environment [5], [6]. Specular reflections and textured floors are the main difficulties faced by floor segmentation algorithms [7]. Besides, accurate approaches must deal with changes in the illumination and structure of the scene.

Floor segmentation and moving features detection are becoming interesting to be used in ground plane-based ego-motion estimation in vision-aided inertial navigation systems (INS), such as [8] and [9]. Although the motion estimation in these methods is based on the ground plane features, they do not specifically address the problem of floor detection. This report presents a floor segmentation algorithm and a moving features detector, which have been conceived to be integrated into a particular INS, presented in [10]. Such system combines visual and inertial information, using a monocular camera and an inertial measurement unit (IMU).

1.2. Related Work

A large amount of work has been done during the past years related to obstacle avoidance and ground plane detection (e.g, [11], [12], [13]). Regarding single camera techniques, an interesting approach was proposed by Lorigo et al. [14] that uses a combination of color and gradient histograms to distinguish free space from obstacles. Wang et al. [15] presented a region-based obstacle detection method for indoor navigation, which works with a single color camera and provides a local obstacle map at high resolution in real-time. Ulrich and Nourbakhsh suggested system [16] is adaptive, since it learns the appearance of the ground during operation. Color appearance is used to classify each individual pixel as belonging either to an obstacle or to the ground. Kim and colleagues, [17] and [18], described two techniques that use homography to estimate the ground plane normal. Then, the floor is detected by computing plane normals from motion fields in image sequences.

The above mentioned methods require static environments and are not able to deal with movable obstacles. Most of the work done regarding this matter is focused on outdoor environments and detection of moving vehicles (e.g., [19], [20]). In Behrad et al. [21] technique, the background motion is estimated and compensated using an affine transform, while Klappstein et al. [22] developed method takes advantage of knowing the camera ego-motion and exploits spatial constraints to detect the motion in a second step. Odometric information is used by Braillon et al., [23] and [24], to model the expected motion of the points on ground plane. The location of the moving obstacles is determined by the points that do not follow this model.

There are also several interesting methods for general image segmentation. Normalized graph cuts are the state-of-the-art in this field. The original idea was presented by Shi and Malik [25]. It is based on representing the image as a graph where nodes are points in a certain measurement space and edges are given a weight representing the similarity between two nodes. Recent approaches achieve impressive results [26], [27] .

However, to the best of our knowledge, only few methods specifically face the problem of floor segmentation. Pears and Liang [28] developed a technique which combines multiple visual cues to detect and segment the ground plane. Lee et al. [29] presented an algorithm that is able to get a geometrical description of a single indoor image. Their method is able to distinguish the floor from walls and ceiling by the use of geometric constraints after edge detection.

The most similar floor segmentation method to the one presented in this report was implemented by Li and Birchfield [7]. They designed a technique, applied to single color images, that combines three visual cues for evaluating the likelihood of horizontal intensity edge lines to be part of the wall-floor boundary. Since their algorithm computes the vanishing point, it is restricted to typical corridor scenes and adapts slowly to camera movements.

1.3. Approach Overview

In this thesis report, a new floor segmentation algorithm that can be implemented in any type of ground plane-based ego-motion estimation system is introduced. Due to its simplicity, a single image method similar to [7], which aspires to draw a polyline defining the wall-floor boundaries, is proposed. Unlike other single image methods in the literature, such as [7] and [30], our proposed system works with grey-scale images and does not require to compute the vanishing point. Consequently, it adapts faster to changes in camera motion and is able to deal with all types of indoor scenes, even with the presence of different kinds of obstacles.

A floor polyline is defined, containing the wall-floor and floor-obstacles boundaries. In order to draw this polyline, an acute way of joining the most important lines from an edge detector [31] is applied. Finally, a mask describing the floor area in the image is generated. In ground plane-based ego-motion estimation approaches, the ground features closest to the camera have the main contribution to the motion estimation. Since our method is designed for such applications, it is not supposed to segment the whole floor, but only a sufficient part of it that is the closest to the camera. A relevant attribute of the new approach is that only the boundaries below the half image are considered, assuming that a sufficient part of the floor with enough feature points is within this region. Moreover, the computational cost of the algorithm is significantly reduced by decreasing the area under analysis.

The design and implementation of a movable features detection technique is also an object of this thesis. After segmenting the floor, feature extraction and matching [32] is performed between consecutive frames. Based on the estimated ego-motion of the camera, the homography matrix for the ground plane is calculated. In contrast to [23] and [24], which are restricted to forward motion, a noteworthy contribution of our method is that the homography is derived for general motion

and rotation of the camera. Then, the expected optical flow is computed for features within the floor mask at the current frame. Moving features are detected by comparing their expected and real motion, given by the estimated optical flow and the correspondences, respectively.

Both methods have been conceived to be part of the INS presented in [10]. Hence, the final system combines floor segmentation, movable feature detection and ego-motion estimation, which have normally been treated separately in the literature. Being able to join these three methods in a final system is also one of the main contributions of this thesis.

1.4. Methodology and Material

During the first weeks of work, the effort was put in getting familiar with the field to achieve a better understanding of the problem to solve. A deep research was carried out, starting with a literature review, with the aim of obtaining a state-of-the-art overview of the subject. This literature review provided a solid foundation upon which the proposed method is based on.

The algorithm has been implemented using MATLAB® [33]. General student license is enough for the floor segmentation algorithm, while feature extraction and matching require a higher-level license, which includes Computer Vision Toolbox.

The implemented algorithms have been tested using several recorded sequences, which reflect typical indoor scenes challenges, such as specular reflections, shadows or strong illumination. Sequences contain different illumination conditions, both textured and homogeneous floors as well as different kind of obstacles. Particularly, the sequences under analysis have some problems that complicate the task of any floor segmentation algorithm. The main ones are listed below:

- Illumination is not homogeneous in the scene.
- There are some burnt parts on the ground.
- The floor is not homogeneous neither in luminance, nor in texture.
- The wall-floor and floor-obstacles boundaries are not always clear.

The AVT Guppy monochrome camera used to record the forward-looking sequences generated images at resolution of **752x480** pixels, **8** bits [0-255] and **10Hz**. In order to maximize the floor area below the half of the image, it was rigidly mounted at the top of a trolley at **85cm** height and shifted **25°** towards the floor. A MicroStrain 3DMGX2 IMU, which is used for ego-motion estimation, is stuck at the bottom of the camera.

Figure 1.1 shows three examples of typical images from the tested sequences. Most of the problems mentioned in 1.4 can be noticed. Moreover, five examples of the recorded original videos can be found in the reproduction list named *"Original Sequences"* in the Youtube channel [34].

Figure 1.1. Typical images from the analyzed sequences

1.5. Outline

This thesis report has been structured as follows. Chapter 1 places the reader by providing a general introduction to the field, stating the problem to solve and giving an overview of the proposed method compared to similar previous work. Chapter 2 describes the implemented algorithm for segmenting a sufficient part of the floor, while the designed method to detect moving features is explained in Chapter 3. In Chapter 4, the role that the implemented methods play in a visual-inertial navigation system is detailed. For its part, Chapter 5 discusses the performance of the systems by presenting some relevant results. Finally, Chapter 6 reflects on the entire project, it contains the conclusions and suggests future work.

Chapter 2

Floor Segmentation Algorithm

This chapter describes the implemented algorithm for segmenting a sufficient part of the floor. Figure 2.1 illustrates a general block diagram of the proposed method. As it has already been pointed out, the designed method works on a single image. Neither stereo nor motion information are needed (see Section 1.3).

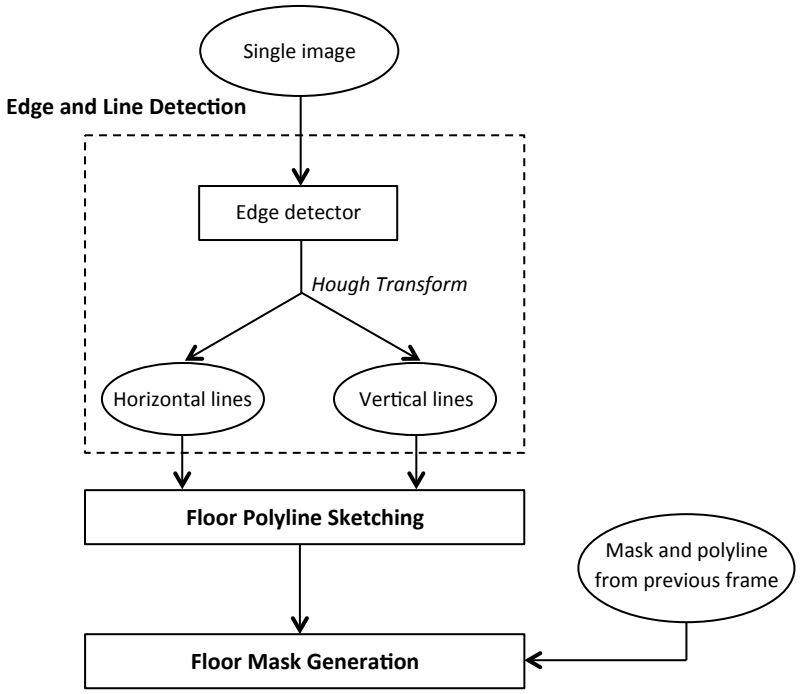


Figure 2.1. Block diagram for the implemented Floor Segmentation Algorithm

The final output of the algorithm is a mask defining the floor area in the image. The method has been divided into three main blocks: 1) edge and line detection, 2) floor polyline sketching and 3) floor mask generation. The goal of the edge and line detection part is to detect the main vertical and horizontal lines of the scene. These lines are then used in the floor polyline sketching block to define a polyline, which contains the wall-floor and floor-obstacles boundaries. In the end, a mask for the floor is generated.

The floor mask is a binary image where all the pixels belonging to the floor are set to white, while the rest of them remain in black. In addition, an image with a polyline defining the ground boundaries placed on the top of the original frame is also outputted. This second image is useful for evaluating the performance of the algorithm. Examples of these outputs are shown along this chapter as well as in the Results Chapter 5.

In the following sections, the three blocks of the designed method are described in detail and some partial results are shown with the aim of helping the reader to follow and understand the whole procedure.

2.1. Edge and Line Detection

The first block consists of detecting, identifying and describing the main lines that are to define the structure of the scene. At the end of this part, two lists are generated, which become the inputs for the floor polyline sketching block (see Section 2.2). One of the lists contains the relevant information from the horizontal lines, while the other does the same for the vertical lines. The structure of these two lists is detailed at the end of this section.

In the following pages, the different steps followed in order to generate the two lists are explained. Figure 2.2 reveals these steps. A single image is the input for this block. First of all, edge detection is performed. The edges are defined by all its points and listed before being transformed to a set of straight lines. Finally, Hough transform is applied in order to prune the lines in the transformed domain and generate the two lists, which are the outputs of this block.

The first procedure is to detect all the edges from the given picture. Canny edge detector [31], which is a common image processing tool, is applied. The edge detector generates a binary image, where the pixels of the detected edges appear in white.

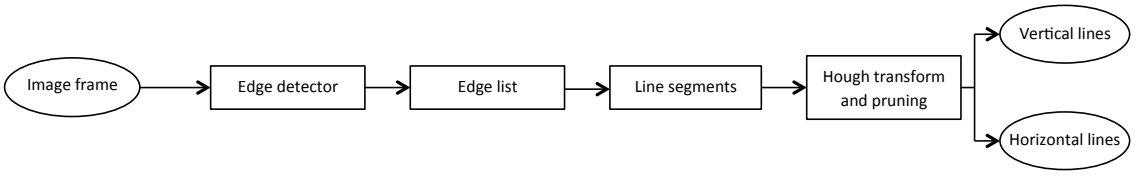


Figure 2.2. Block diagram for the Edge and Line Detection block

Figure 2.3 shows two examples of this mask. The top images are the original frames. The bottom images are the Canny edge detector outputs from the top images. Notice that, by just looking at the binary images, one can recognize the structure of the scene and identify the region belonging to the ground.

Figure 2.3. Two examples of masks from Canny edge detector

Edge detection is a critical step because the performance of the whole algorithm depends on it. If an edge is not detected at this point, there is no chance to recover it in the next steps. Moreover, it is also important to mention that the edge detector is very sensitive to illumination; consequently the whole system is sensitive to it as well.

By adjusting the hysteresis thresholds (th_{high}, th_{low}) and smoothing parameter (σ) one can improve Canny edge detector performance. The first step of the detector consists of smoothing the image by a Gaussian in order to remove high frequency

components. The standard deviation of this Gaussian filter is σ. Once the gradient is computed, a double thresholding is applied. As the gradient values are normalized, th_{high} and th_{low} are between 0 and 1. If the gradient for a pixel is greater than th_{high}, it is automatically defined to be part of an edge. The neighborhood of pixels with gradient values between th_{low} and th_{high} is checked in order to decide if it belongs or not to an edge.

These parameters can be optimized for every image. The developed method must deal with different types of images, since the characteristics of the different frames along a sequence, such as the number and position of obstacles or the illumination, might change significantly. Some sequences have been tested with the aim of finding the Canny edge detector parameters that give the best performance along the whole sequences. For the tested sequences, the optimal performance is achieved when the smoothing parameter is $\sigma = \mathbf{1}$ and the hysteresis higher and lower thresholds are $th_{high} = \mathbf{0.15}$ and $th_{low} = \mathbf{0.05}$.

After detecting all the edges, a list of all the points belonging to each of them is generated. Due to real world noisy conditions, short spurious edges might appear that are not important regarding the scene structure description. Thus, edges shorter than **60** pixels are removed at this point and will not be taken into account from now on. Besides, pruning these small edges simplifies all following handlings and reduces computational cost.

The floor polyline sketching block requires straight vertical and horizontal lines (see Section 2.2). Therefore, each of the edges in the list is fitted into a set of straight lines, called segments. A tolerance parameter (tol) controls the similarity between the original edge and the set of segments. This parameter ensures that the distance between any point in the new segment and its corresponding point in the analogous original edge is not greater than tol pixels. In order to simplify as much as possible every edge, but at the same time keep the similarity between the segments and the original contours, the tolerance parameter is set to $tol = \mathbf{10}$ pixels.

The difference between the original and the composed-by-segments edges description can be noticed in Figure 2.4. In all the cases, each of the edges in the list is plotted in a random different color. The images at the top show the original edges, directly extracted from the Canny edge detector mask. In the bottom images, edges are linearized and represented by a set of straight lines. Notice that the effect of linearizing is much more noticeable for curved edges.

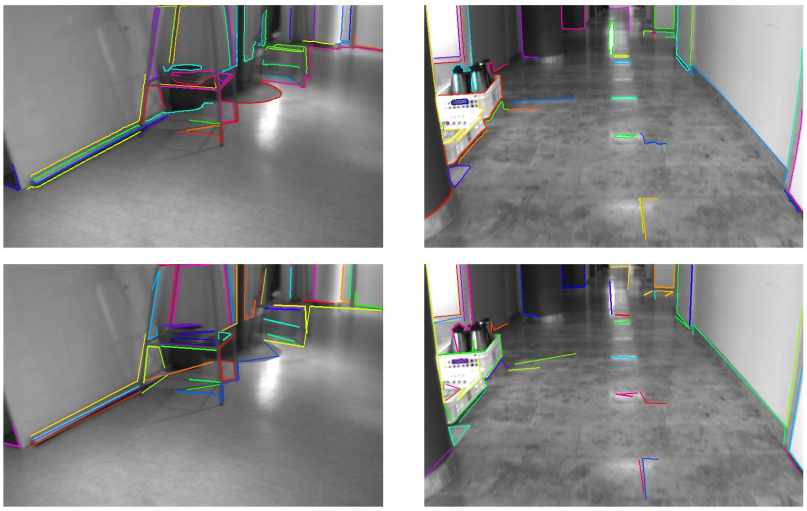


Figure 2.4. Edges and line segments from two different frames

In order to prune and classify the straight lines, Hough transform [35] is applied. It is a method for detecting straight lines from an image. The main idea is to consider lines in terms of their parameters. Polar coordinates (ρ, θ) are normally used. While ρ represents the distance between the line and the origin, the angle θ is the one described by the vector joining the origin and the closest point of the line with respect to the vertical. Equation (2.1) reveals the general expression for Hough transform.

$$\rho = x \cos\theta + y \sin\theta \tag{2.1}$$

Consequently, a line in spatial domain corresponds to a unique point in the Hough domain (ρ, θ), while each point in spatial domain corresponds to a sinusoidal curve. These two characteristics can be seen in Figure 2.5. On the top, a binary image shows the line segments of a scene. The graph on the bottom is the analyzed image in the Hough domain. Notice that sinusoidal curves can be appreciated. The red squares indicate the position of the maxima in the graph, which are in the curves intersections. These points correspond to straight lines in the spatial domain (see Figure 2.6 - top).

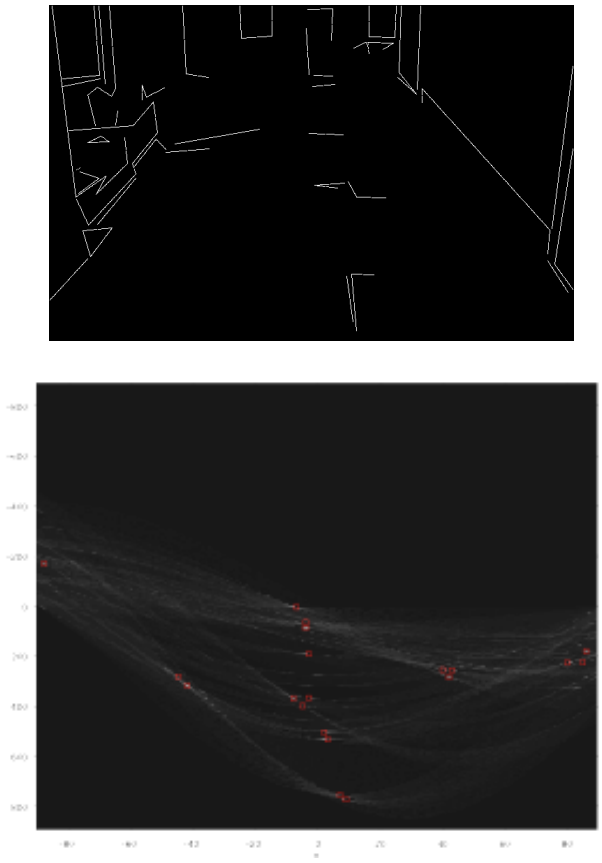


Figure 2.5. Example of Hough transform graph

In the transformed domain, it becomes much easier to filter lines by the angle they describe. Line segments are divided into two sets: vertical (V) and horizontal (H). Based on the tested sequences, a slope range to classify the lines is determined. A segment is classified as vertical if its slope is within $\pm\mathbf{10}^\circ$ of the vertical direction. Horizontal lines are given a wider slope range: $\pm\mathbf{65}^\circ$ of the horizontal direction. All the line segments describing an angle outside the mentioned ranges are removed. From now on, the vertical and horizontal lines are treated separately.

An inverse Hough transform is computed for both sets in order to get the lines in spatial definition again. As pointed out in Section 1.1 and Section 1.3, the floor segmentation method is designed for a specific ground plane-based ego-motion estimation approach [10]. In such systems, the ground features closest to the camera have the main contribution to the motion estimation, while far-away features are negligible. Consequently, the implemented algorithm is not supposed to segment the whole floor, but only a sufficient part that is the closest to the camera. Hence,

lines above the half of the image will not be taken into account. Besides, this constraint reduces the computational cost of the algorithm, since it frees the rest of the algorithm from dealing with a lot of lines that are less likely to be part of the sought boundaries. Applying half image constraint (see Figure 2.7 for points nomenclature) entails:

- V whose bottom points lie above $\frac{1}{2}$ of the image are removed.
- V with both top and bottom points below $\frac{1}{2}$ of the image are removed.
- H whose beginning and ending points lie above $\frac{1}{2}$ of the image are removed.
- H with just one beginning/ending point below $\frac{1}{2}$ of the image are cut. The new beginning/ending point is set at the point of the line that corresponds to the y coordinate value equal to $\frac{1}{2}$ of the image.

Figure 2.6 illustrates the effects of applying the above listed actions. The red line marks the half of the image. Horizontal lines are represented in black, while vertical in blue. The top images show all the lines after the pruning in the Hough domain. In the images at the bottom, the remaining lines after applying the half image constraint are represented.

Figure 2.6. Effects of applying the half image constraint

After all these steps, the sets of lines that are going to be used to draw the floor polyline (see Section 2.2) are already defined. The last part in this block is to generate two lists (named *listH* and *listV*), containing the relevant information of the lines for further handlings. The data recorded in these two lists is detailed below:

listH: [beginning point | ending point | length | orientation]
listV: [bottom point]

The length stored in *listH* corresponds to the original length of the line (before cutting it, if that is the case), while the orientation is with respect to the vertical.

2.2. Floor Polyline Sketching

In this section, an acute way of joining the line segments from the two lists generated in the previous block (*listH* and *listV*) is described. This is the main part of the implemented method. A polyline representing the wall-floor and floor-obstacles boundaries at the bottom half image is drawn by judiciously selecting and joining the lines. This floor polyline is the output of the current block.

Knowledge of the height and orientation of the camera, as well as typical structure of the scenes and geometric constraints have been taken into account in order to design the method to generate the floor polyline. The main idea is to draw a polyline, from left to right, connecting the endings of the vertical and horizontal lines one progressively encounters. Figure 2.7 introduces the nomenclature that is used for the end points of the lines along this section. Vertical lines are painted in blue, horizontal in black.

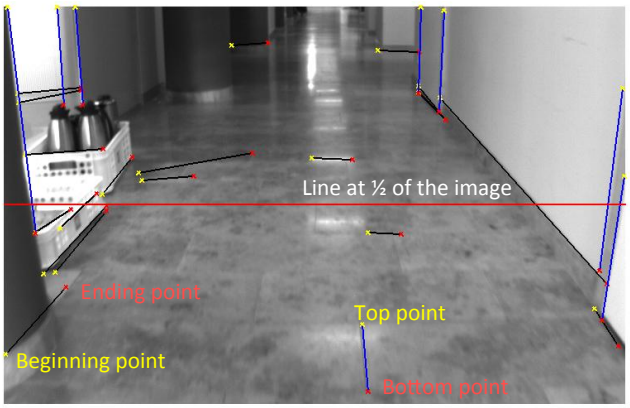


Figure 2.7. Nomenclature used for the end points.

For every iteration in the main algorithm (see Figure 2.8), the first step is to find which point, within $listH$ and $listV$, is most to the left. In order to define priorities, $listV$ and $listH$ are sorted before starting to draw the floor polyline. Since horizontal lines are more meaningful than vertical for defining the floor polyline, the assigned priorities seek to stress the use of horizontal lines. In addition, it is crucial to ensure that the segmented part of the image do belong to the ground plane, rather than segment a big area. Hence, lines that are closer to the bottom of the image are also stressed:

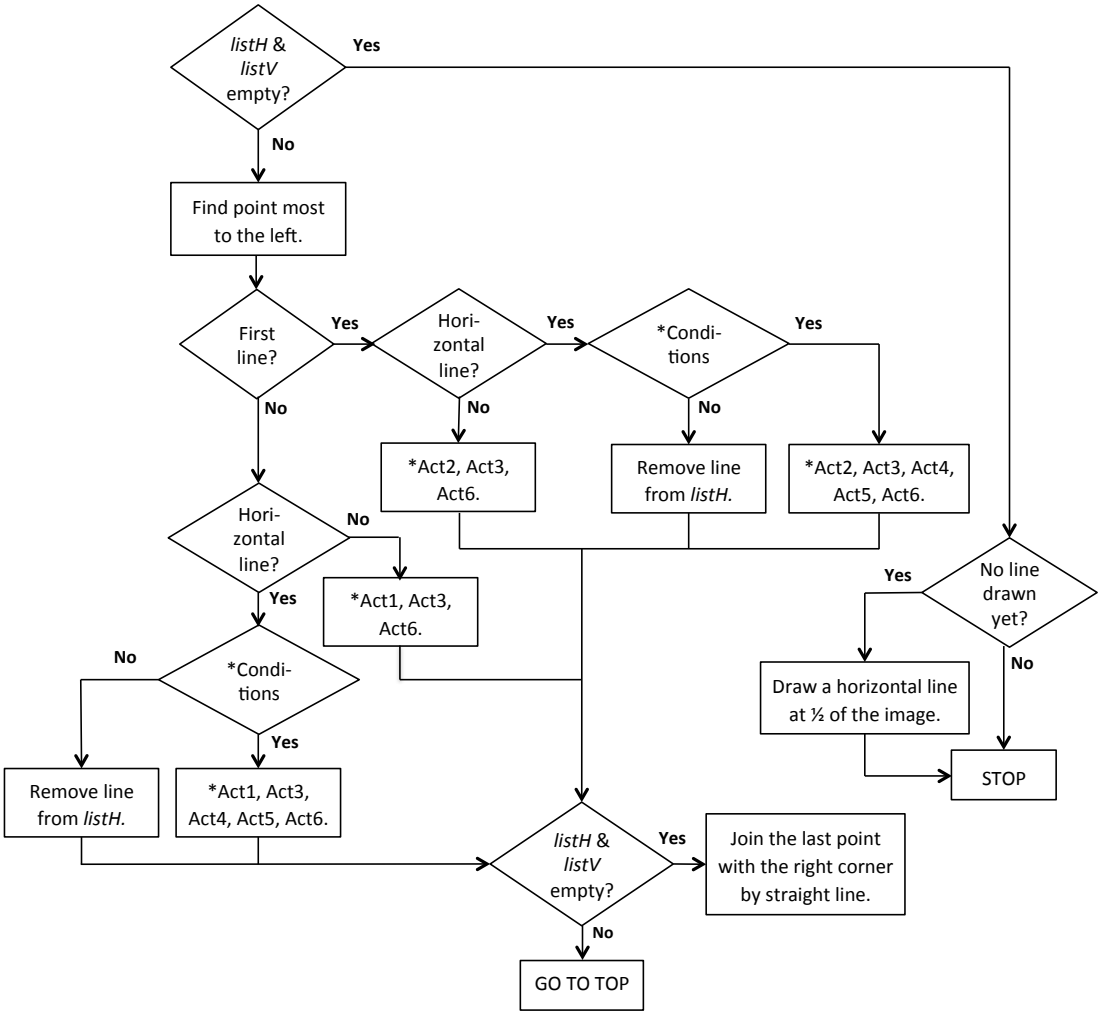
- $listH$ has priority over $listV$.
- In $listH$, beginning/ending points that are closer to the bottom of the image have priority.
- In $listV$, bottom points that are closer to the bottom of the image have priority.

After establishing priorities, the algorithm to draw the floor polyline can be applied. By watching its flowchart (Figure 2.8), one can follow the procedure and understand how the floor polyline is progressively drawn. In addition, the algorithm is described and justified in the next lines.

While there is still some element in $listH$ or $listV$, the line which has its beginning point (in the case of horizontal lines) or its bottom point (in the case of vertical lines) most to the left is selected. During all the procedure, the coordinates of the last point of the line segment that is being drawn are stored ($lastX$,$lastY$).

When a vertical line is selected, its bottom point is directly used for drawing the floor polyline. However, in order to avoid the effect of spurious edges that might appear because of textured floors or specular reflections, three conditions are checked when a horizontal line is selected. At least one of these conditions must be accomplished to consider the horizontal line as a segment of the floor polyline, otherwise it is removed from $listH$ and the algorithm moves to the next line in the lists.

The two first conditions are related to how the wall-floor boundary should look like, while the third one is to ensure obstacle detection. Horizontal lines that are part of the wall-floor boundary are expected to be long, since the original length of the line is considered. Due to the height and orientation of the camera, they are also expected to describe a certain angle with respect to the vertical. For possible obstacles, both the horizontal line of its base and the vertical lines of its edges are detected, so the third condition holds.



Act1: Join point to {*lastX,lastY*} by a straight line.
Act2: Join point to left corner by a straight line.
Act3: Update *lastX* and *lastY*
Act4: Follow the whole horizontal line.
Act5: Reset beginning points of horizontal lines at the left of *lastX* to (*lastX*+1,f(*lastX*+1)), f(x) refers to the equation of the modified line. Remove all lines at the left of *lastX.*
Act6: Remove line from list.

Conditions: *(at least 1 must be accomplished)*
- Larger than 150 pixels?
- Orientation <±60º AND beginning/ending point closer than 60 pixels from a corner?
- Beginning/ending point closer than 45 pixels from a bottom point of a vertical line?

Figure 2.8. Flowchart of the floor polylne sketching algorithm

The evaluated conditions for the horizontal are formally stated below:

- Is it larger than **150** pixels?
- Is its orientation $<\pm$**60**$^\circ$ with respect to the vertical and is its beginning/ending point closer than **60** pixels from a corner?
- Is its beginning/ending point closer than **45** pixels from a bottom point of a vertical line?

On the contrary, as only the vertical lines with their top point above the middle of the image and their bottom point below it are considered, there is no need to apply additional conditions in order to avoid the previously mentioned problems. When a vertical line is selected, this is how the algorithm proceeds:

- Join its bottom point to the left corner by a straight line (if it is the first line).
- Join its bottom point to ($lastX$,$lastY$) by a straight line (if other lines have been considered).
- Update $lastX$ and $lastY$ with the coordinates of its bottom point.
- Remove the analyzed line from the list.
- Search next line

For its part, when a horizontal line is considered to be part of the floor polyline, this is how the algorithm proceeds:

- Join its beginning point to the left corner by a straight line (if it is the first).
- Join its beginning point to ($lastX$,$lastY$) by a straight line (if other lines have been considered).
- Update $lastX$ and $lastY$ with the coordinates of its ending point.
- Follow the whole horizontal line.
- Reset beginning points of horizontal lines that are at the left of $lastX$. The new coordinates for the beginning points of these lines are set to ($lastX$+1,f($lastX$+1)), where f(x) refers to the equation of the horizontal line that is being modified.
- Remove all horizontal and vertical lines at the left of $lastX$.
- Remove the analyzed line from the list.
- Search next line

After analyzing all the lines, the last considered point ($lastX$,$lastY$) is joined with the right corner of the image by a horizontal straight line, giving the final floor polyline. If no line has been used so far, the whole bottom half part of the image is assumed to be part of the ground. Consequently, the floor polyline becomes a straight horizontal line at $\frac{1}{2}$ of the image.

Figure 2.9 shows two examples of the drawn polyline. The top images help the reader to get a better idea of how the algorithm works and which lines are used to draw the final polyline. The final floor polyline is painted in white. Horizontal lines are represented in black, while vertical in blue. A horizontal line at the half of the image is plotted in red. The two images at the bottom contain only the final floor polyline, painted in black, on the top of the original image frame.

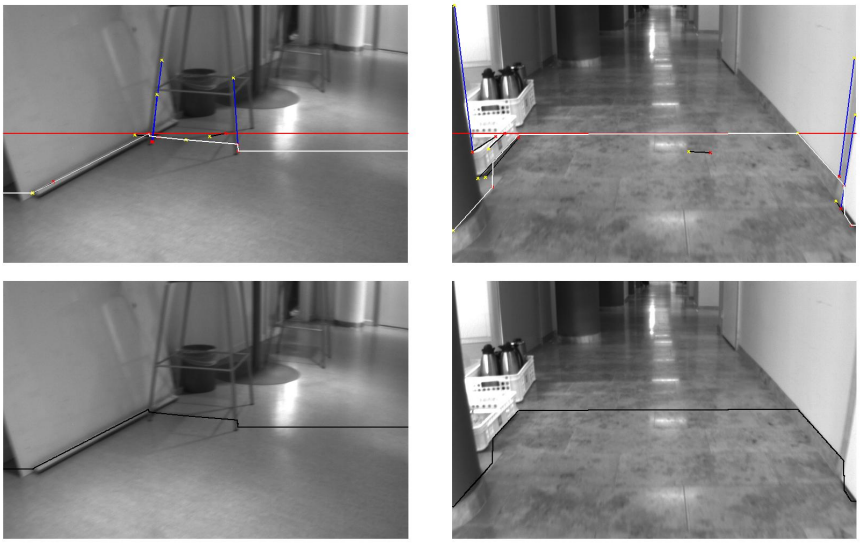


Figure 2.9. Lines used to draw the polyline and its final definition

2.3. Floor Mask Generation

Once the floor polyline has been defined, the floor mask, which is the final output of the algorithm, is generated. This is what the last block from the designed method consists of. Figure 2.10 illustrates its diagram. Notice that in this block, information of the previous frame (previous floor polyline and mask) is used.

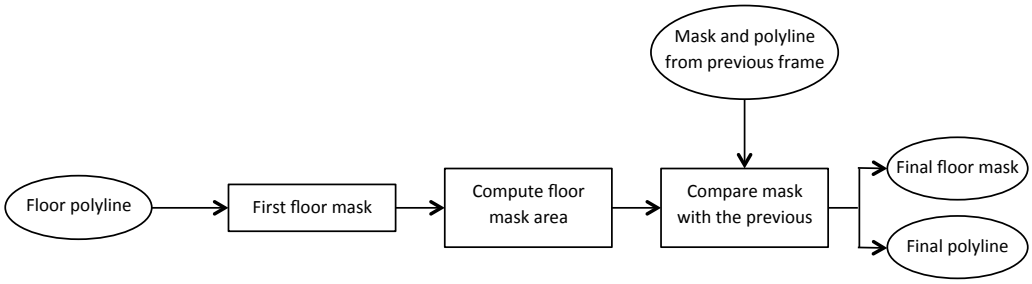


Figure 2.10. Block diagram for the Floor Mask Generation block

A first version of the mask is generated by setting the pixels below the polyline to white. The rest of the pixels in the image remain in black. Then, the area of the floor region is computed by summing-up the number of white pixels within the mask ($Npix_t$). In order to avoid sudden changes that might reduce dramatically the floor area, $Npix_t$ is compared with the area defined by the mask at the previous frame $Npix_{t-1}$. If the new area is more than **30%** smaller than the area of the floor region at the previous frame $Npix_t < 0.7Npix_{t-1}$, the method keeps the same floor polyline and the same mask as the previous frame. Real obstacles do not appear fast enough in order to reduce more than 30% of the region of the floor from one frame to the next one. On the contrary, changes in the illumination of the scene or textured floors can cause this effect and the system must ignore them.

Figure 2.11 shows the final masks for the two examples that have been used all along this chapter.

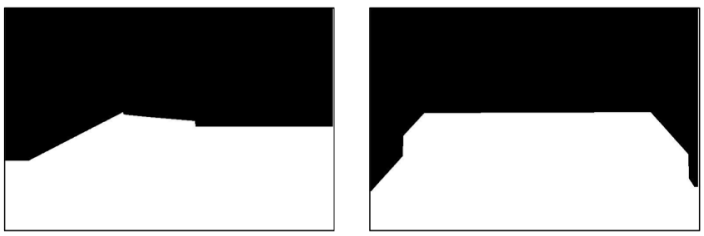


Figure 2.11. Final masks for the two analyzed frames

Chapter 3

Moving Features Detection

In this chapter, the designed method to detect moving features is explained. Its goal is to identify the features that belong to movable obstacles, from a set of feature correspondences between pairs of frames. To achieve this purpose, both the floor mask (see Chapter 2) and the estimated camera ego-motion are used. Figure 3.1 shows a general block diagram of how the detection of moving features is performed.

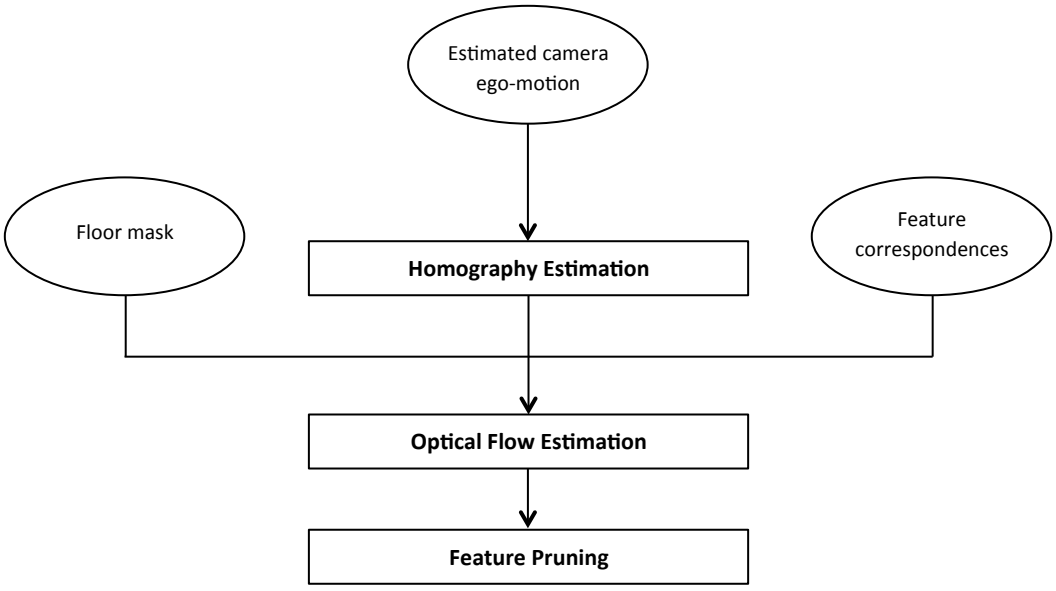


Figure 3.1. Block diagram for the Moving Features Detection method

The system has been divided into three main blocks: 1) homography estimation, 2) optical flow estimation and 3) feature pruning. In the first block, the estimated ego-motion of the camera is used to derive the homography of the ground plane.

Then, the optical flow is estimated from the derived homography. The mentioned optical flow is only computed for the feature points that lie inside the floor mask in the current frame. Finally, in the feature pruning block, the estimated optical flow and the feature correspondences are compared in order to remove the movable features.

When an obstacle moves fast, its contours appear blurred and thus, it is difficult for edge detectors to distinguish obstacle-floor boundaries. Movable objects with well-defined edges are rejected by the floor mask. However, irregular moving obstacles, such as feet, might partially lie within the mask. To avoid selecting features belonging to movable obstacles, a moving features detection method for features inside the mask has been designed.

3.1. Homography Estimation

In this section, the whole derivation to get the homography matrix, which defines a linear projection from one plane to the image, is discussed for the points on the ground plane. In the following lines, all the equations used to get the homography from the estimated camera ego-motion are presented. Unlike similar approaches like [23] and [24], which are restricted to forward motion, the developed method derives the homography matrix for general motion and rotation of the camera.

To simplify the motion model, projective geometry is applied. Under this assumption, the homography matrix projects points from one plane to another. Since homogeneous coordinates are used, a point in the 3D world space is expressed like $(X, Y, Z, 1)^T$, while a point in the image pixel coordinates becomes $(u, v, w)^T$. The projection equation for general motion of the camera of 3D points into the image pixel coordinates is:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}_{Image} = K R_b^c \begin{bmatrix} R_n^b & -R_n^b \vec{p} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}_{World} \tag{3.1}$$

where K is the matrix of the intrinsic camera parameters. It is defined (see equation (1.6) in [36]), considering a null skew factor, like:

$$K = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

R_b^c is the direction-cosine matrix that rotates a vector from camera to body and it is constant all along the sequence. Vector $\vec{p}$ contains the estimated position of the camera at the current frame:

$$\vec{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \tag{3.3}$$

and R_n^b is the matrix that rotates from body to navigation. It can be defined by the angles of the rotated coordinate system (equation (2.31) in [37]) as:

$$R_n^b = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{3.4}$$

where

$$R_{11} = \cos(\psi)\cos(\theta) \tag{3.5a}$$
$$R_{12} = \sin(\psi)\cos(\theta) \tag{3.5b}$$
$$R_{13} = -\sin(\theta) \tag{3.5c}$$
$$R_{21} = -\sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) \tag{3.5d}$$
$$R_{22} = \cos(\psi)\cos(\phi) + \sin(\psi)\sin(\theta)\sin(\phi) \tag{3.5e}$$
$$R_{23} = \cos(\theta)\sin(\phi) \tag{3.5f}$$
$$R_{31} = \sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\phi) \tag{3.5g}$$
$$R_{32} = -\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) \tag{3.5h}$$
$$R_{33} = \cos(\theta)\cos(\phi) \tag{3.5i}$$

The implemented method seeks to estimate the homography matrix for the points on the ground plane. Assuming that the ground is flat and located at $Z = 0$, its points can be expressed like $(X, Y, 0, 1)^T$. Thus, equation (3.1) is simplified by removing the third column of the $\mathbf{3} \times \mathbf{4}$ matrix that multiplies the 3D points. The resulting equation can be expressed as a homography, since it describes a projection between two planes, the ground and the image:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix}_{Image} = H \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}_{Ground} \tag{3.6}$$

where H is defined by:

$$H = K R_b^c J \tag{3.7}$$

and

$$J = \begin{pmatrix} R_{11} & R_{12} & -R_{11}p_x - R_{12}p_y - R_{13}p_z \\ R_{21} & R_{22} & -R_{21}p_x - R_{22}p_y - R_{23}p_z \\ R_{31} & R_{32} & -R_{31}p_x - R_{32}p_y - R_{33}p_z \end{pmatrix} \tag{3.8}$$

The motion of the points can be understood as their temporal derivative in the image plane. Therefore, equation (3.6) must be differentiated in order to have an expression for $(\dot{u}, \dot{v}, \dot{w})^T$:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}_{Image} = \dot{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}_{Ground} + H \begin{pmatrix} \dot{X} \\ \dot{Y} \\ 0 \end{pmatrix}_{Ground} \tag{3.9}$$

This last equation can be easily simplified because the temporal derivative for the points on the ground $\dot{X}$ and $\dot{Y}$ is equal to 0, since they do not change their position in the 3D world and remain static. The new relation is:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}_{Image} = \dot{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}_{Ground} \tag{3.10}$$

Finally, from equation (3.6) and (3.10), one can infer the relation between the homogeneous coordinates of a pixel on the ground and its temporal derivatives:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}_{Image} = \dot{H} H^{-1} \begin{pmatrix} u \\ v \\ w \end{pmatrix}_{Image} \tag{3.11}$$

Equation (3.11) is the expression needed to estimate the optical flow. Notice that, not only must H be estimated, but also its temporal derivative. The expressions defining this derivative are written below:

$$\dot{H} = K R_b^c \dot{J} \tag{3.12}$$

The derivative with respect to the time of J depends on $\vec{p}$, the estimated velocity of the camera $(v_x, v_y, v_z)^T$ and the temporal derivative of R:

$$\dot{J}_{11} = \dot{R}_{11} \tag{3.13a}$$
$$\dot{J}_{12} = \dot{R}_{12} \tag{3.13b}$$
$$\dot{J}_{13} = -(\dot{R}_{11} p_x + R_{11} v_x + \dot{R}_{12} p_y + R_{12} v_y + \dot{R}_{13} p_z + R_{13} v_z) \tag{3.13c}$$
$$\dot{J}_{21} = \dot{R}_{21} \tag{3.13d}$$
$$\dot{J}_{22} = \dot{R}_{22} \tag{3.13e}$$
$$\dot{J}_{23} = -(\dot{R}_{21} p_x + R_{21} v_x + \dot{R}_{22} p_y + R_{22} v_y + \dot{R}_{23} p_z + R_{23} v_z) \tag{3.13f}$$
$$\dot{J}_{31} = \dot{R}_{31} \tag{3.13g}$$
$$\dot{J}_{32} = \dot{R}_{32} \tag{3.13h}$$
$$\dot{J}_{33} = -(\dot{R}_{31} p_x + R_{31} v_x + \dot{R}_{32} p_y + R_{32} v_y + \dot{R}_{33} p_z + R_{33} v_z) \tag{3.13i}$$

For its part, the derivative with respect to the time of R depends on the angles of the rotated coordinate system and their rotational velocity $(w_\phi, w_\theta, w_\psi)^T$:

$$\dot{R}_{11} = -w_\psi \sin(\psi)\cos(\theta) - w_\theta \cos(\psi)\sin(\theta) \tag{3.14a}$$
$$\dot{R}_{12} = w_\psi \cos(\psi)\cos(\theta) - w_\theta \sin(\psi)\sin(\theta) \tag{3.14b}$$
$$\dot{R}_{13} = -w_\theta \cos(\theta) \tag{3.14c}$$
$$\dot{R}_{21} = -w_\psi \cos(\psi)\cos(\phi) + w_\phi \sin(\psi)\sin(\phi) - w_\psi \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)[w_\theta \cos(\theta)\sin(\phi) + w_\phi \sin(\theta)\cos(\phi)] \tag{3.14d}$$
$$\dot{R}_{22} = -w_\psi \sin(\psi)\cos(\phi) - w_\phi \cos(\psi)\sin(\phi) + w_\psi \cos(\psi)\sin(\theta)\sin(\phi) + \sin(\psi)[w_\theta \cos(\theta)\sin(\phi) + w_\phi \sin(\theta)\cos(\phi)] \tag{3.14e}$$
$$\dot{R}_{23} = -w_\theta \sin(\theta)\sin(\phi) + w_\phi \cos(\theta)\cos(\phi) \tag{3.14f}$$
$$\dot{R}_{31} = w_\psi \cos(\psi)\sin(\phi) + w_\phi \sin(\psi)\cos(\phi) - w_\psi \sin(\psi)\sin(\theta)\cos(\phi) + \cos(\psi)[w_\theta \cos(\theta)\cos(\phi) - w_\phi \sin(\theta)\sin(\phi)] \tag{3.14g}$$
$$\dot{R}_{32} = w_\psi \sin(\psi)\sin(\phi) - w_\phi \cos(\psi)\cos(\phi) + w_\psi \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)[w_\theta \cos(\theta)\cos(\phi) - w_\phi \sin(\theta)\sin(\phi)] \tag{3.14h}$$
$$\dot{R}_{33} = -w_\theta \sin(\theta)\cos(\phi) - w_\phi \cos(\theta)\sin(\phi) \tag{3.14i}$$

3.2. Optical Flow Estimation

This block aims to estimate the optical flow for the matched feature correspondences within the floor mask. The optical flow is defined as the apparent motion of image brightness patterns in an image sequence [38]. It is a vector that, for every pixel, describes the motion from one frame to the other. By definition, the optical flow can be expressed as:

$$\vec{f}(u,v,w) = \left(\frac{\dot{u}}{w}, \quad \frac{\dot{v}}{w} \right) = \left(\frac{\dot{u}w + u\dot{w}}{w^2}, \quad \frac{\dot{v}w + v\dot{w}}{w^2} \right) \tag{3.15}$$

As already pointed out, there is no need to compute the optical flow for all the pixels in the image, but only for the pixel coordinates of the feature points that lie inside the floor mask in the current frame. Therefore, the floor mask, generated by the floor segmentation algorithm (see Section 2.3), is required at this point.

Once the homography matrix and its temporal derivative have been derived, the optical flow can be computed. Below, the equations to get it are presented. Furthermore, based on the analysis of some examples, a discussion of the optical flow estimation performance is given.

The extra coordinate (w) in equation (3.15) is added because of the use of a homogeneous coordinate system, although its value is not known. Hence, the previous equation must be simplified in order to get an expression depending only on the known values. The following transformations are used:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \frac{1}{w} \begin{pmatrix} u \\ v \end{pmatrix} and \begin{pmatrix} \dot{u}' \\ \dot{v}' \\ \dot{w}' \end{pmatrix} = \frac{1}{w} \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} \tag{3.16}$$

Combining the transformations with equation (3.11), the expression below is obtained. It describes the relation between the temporal derivative of the projective coordinates of a pixel $(\dot{u}, \dot{v}, \dot{w})$ and the homography matrix (H), its derivative with respect to the time ($\dot{H}$) and the pixel coordinates (u', v').

$$\begin{pmatrix} \dot{u}' \\ \dot{v}' \\ \dot{w}' \end{pmatrix} = \dot{H} H^{-1} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \tag{3.17}$$

The final expression for the optical flow is then defined as:

$$\vec{f}(u', v') = \begin{pmatrix} \dot{u}' - u'\dot{w}' \\ \dot{v}' - v'\dot{w}' \end{pmatrix} \tag{3.18}$$

This optical flow model is valid only below the horizon line (equation (3.19)). Nevertheless, by applying the floor mask, which rises at most until the half of the image, it is ensured that all the analyzed features are valid.

$$h_l = v_0 - \alpha_v \tan\phi \tag{3.19}$$

At this point, one must notice that, since the homography matrix is derived only for the ground plane (see Section 3.1), equation (3.18) expresses the theoretical optical flow vector for each feature under the assumption that it belongs to the ground. Consequently, this theoretical flow is expected to represent the real motion (given by the feature correspondences) for the features of the floor, while it should significantly differ for features belonging to movable objects. This fact is exploited in the feature pruning block (see Section 3.3) in order to remove features that do not belong to the floor.

Some examples of the estimated optical flow and feature correspondences for different pairs of frames are shown below. In all the examples, the separation between the two frames is 3 samples. The position of the feature in the current frame is indicated by a red circle, while its counterpart in the previous, by a green cross. The inverse optical flow estimated vector for every feature (from the current frame to the previous) is plotted in blue, while the correspondence vector is represented in yellow. Figure 3.2 shows four examples where the estimation holds, while Figure 3.3 illustrates some examples where it fails.

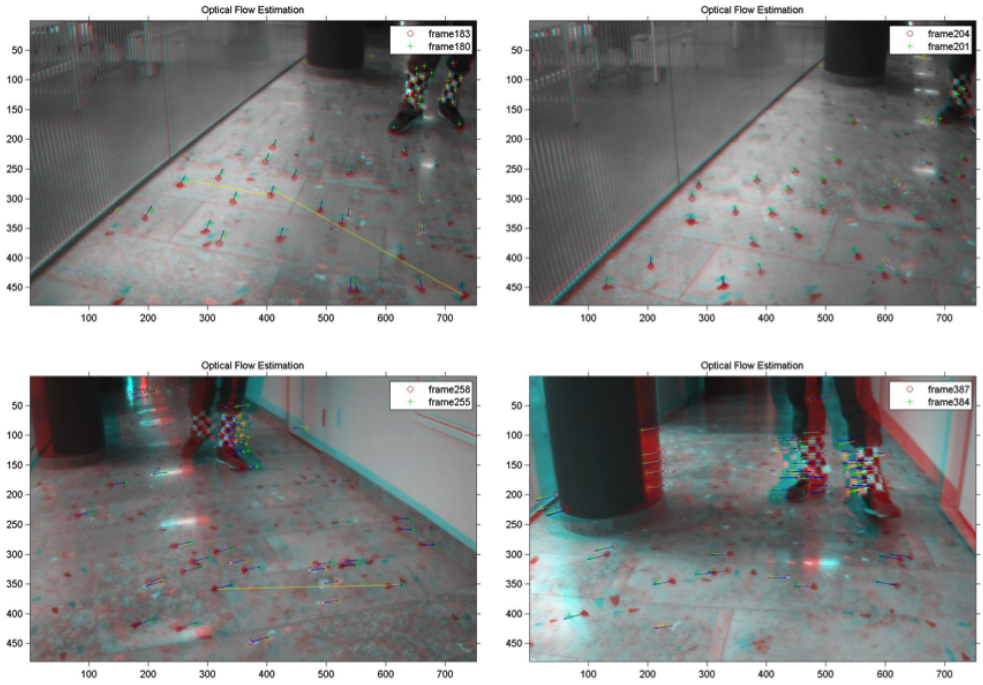


Figure 3.2. Four exemples where the optical flow estimation is correct

In both figures, regardless of the horizon line constraint, the optical flow is estimated for all the features. However, the following analysis of the performance is focused only on the features in the bottom half part of the image that are part of the ground. For these features, one expects the estimated vector to be close to the correspondence. It certainly does, for the examples in Figure 3.2. This figure also proves that the designed method is able to deal with general motion of the camera.

On the contrary, Figure 3.3 reveals that the estimation is not valid for all the cases. Despite carrying out a lot of tests (some can be seen in Section 5.2), no pattern is detected in the erroneous estimations. Notice that the optical flow estimation

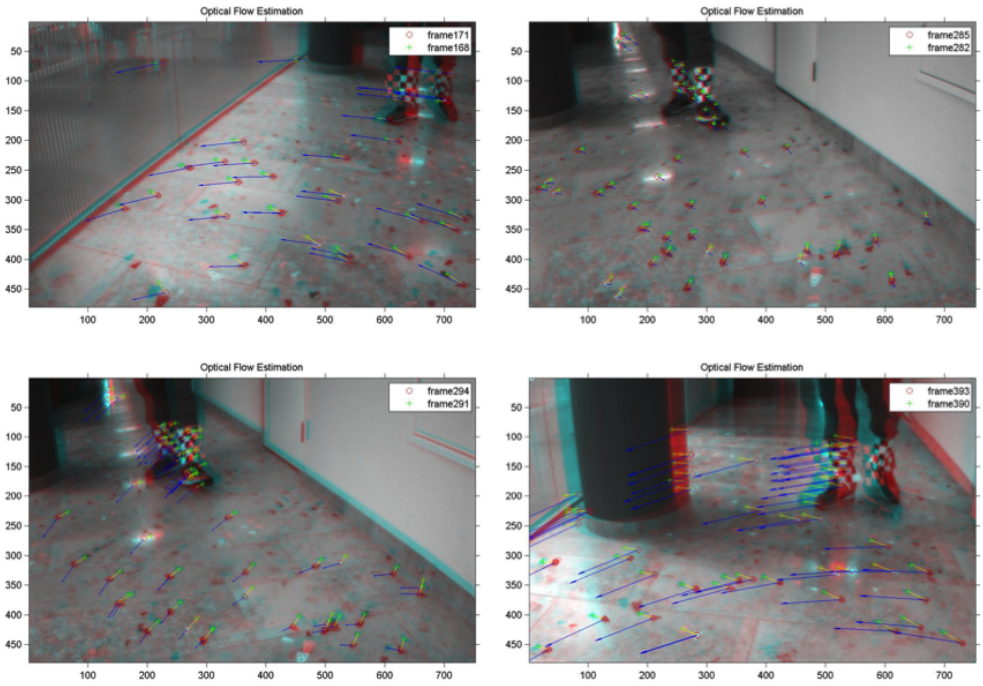


Figure 3.3. Four exemples where the optical flow estimation fails

changes from a correct value to a completely incorrect one in a very short period (see both bottom-left images from Figure 3.2 and Figure 3.3).

Nevertheless, a couple of observations that might help to understand this behavior can be mentioned. The method is extremely sensitive to the estimated ego-motion of the camera. Thus, a small deviation or error in it becomes critical for the optical flow estimation performance. Moreover, although the pixels are normalized before computing their estimated optical flow to avoid the effect of camera nonlinearities, these might have a strong effect on the method success as well. A deeper study of how these aspects influence the performance of the optical flow estimation, must be done as a future work.

3.3. Feature Pruning

In this section, the method used to prune the features that do not belong to the floor is described. So far, the procedure to calculate a theoretical optical flow vector for every feature has been explained. This last block computes the

difference between the theoretical motion, given by the optical flow vector, and the real displacement, given by the correspondences; with the aim of establishing the likelihood of each feature to be part of the ground and deciding which of them must be removed.

All the results and comments regarding the performance of this block are valid only under the assumption that a good estimation of the optical flow is previously achieved. That is to say that similar performances as Figure 3.2 are considered.

Figure 3.4 presents the procedures followed to accomplish the mentioned goal. The estimated optical flow and the feature matching correspondences are the inputs of this last block. After filtering the features by an angle criterion, a difference value is calculated for the remaining features. Finally, a similarity value, which is to represent the likelihood of each feature to be part of the ground, is computed. By thresholding this similarity value, the final set of features belonging to the floor is generated.

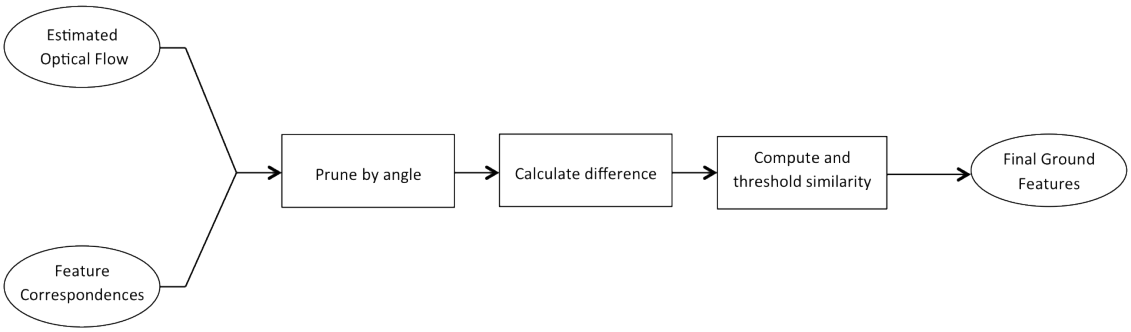


Figure 3.4. Block diagram for the feature pruning part

In the first step, features are removed if their optical flow vector and their correspondence vector angles differ more than a certain threshold. For the tested pairs of frames, this threshold is heuristically set to $\mathbf{20}^\circ$. Mismatches and features with a significant different motion direction from the motion of the camera are removed right at this point.

In order to prune the remaining moving features, a similarity value is calculated. To begin with, the intensity difference between every feature in the current frame and its expected position in the previous image is computed. This difference value is defined as the sum of squares differences (SSD) within a neighborhood region around the two positions:

$$D = \sum_{(i,j)\in W} [I_t(u+i, v+j) - I_{t-\Delta t}(\bar{u}+i, \bar{v}+j)]^2 \tag{3.20}$$

where W is a window defining the neighborhood to be analyzed; $I_t(u, v)$ is the intensity value of a feature at $(u, v)^T$ in the current frame; and $I_{t-\Delta t}(\bar{u}, \bar{v})$ is the intensity value of the feature in the previous image $(\bar{u}, \bar{v})^T = (u, v)^T - \Delta t \vec{f}(u, v)^T$.

Figure 3.5 shows two examples of the windows used to compute the difference value in equation (3.20). Both images are from the same plot. Zoom is applied in order to put the focus on two different regions. On the left image ground features are represented, while on the right image the features belonging to the moving foot and leg are highlighted. The feature position in the current frame (red cross) is in the middle of the red window, which is compared with the blue one that describes the neighborhood area around the expected position of the feature in the previous frame (blue cross). Feature correspondences are plotted in yellow. Notice that, since the optical flow is computed assuming features on the ground, the expected motion is much more similar to the real one for the examples in the left image.

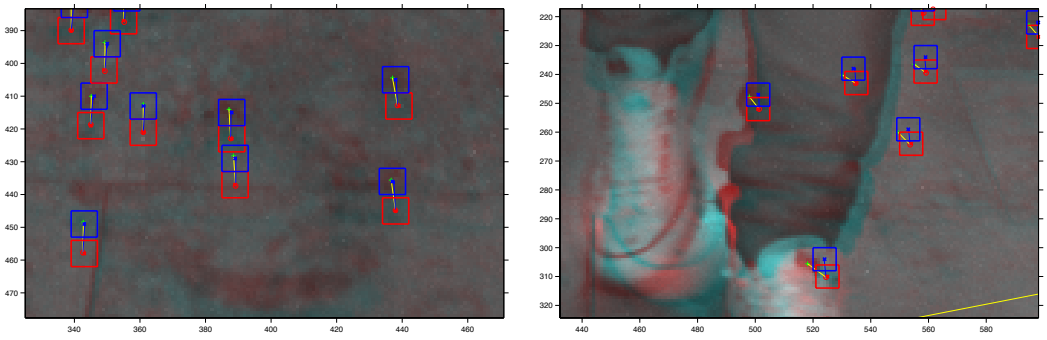


Figure 3.5. Example of windows used in equation (3.20)

The size of the analyzed windows might change the performance of the system. Larger windows tend to homogenize the area under analysis, and thus, are to allow greater deviations in the estimation. For the tested pairs of frames, the window size is set to $\mathbf{9 \times 9}$, which reaches a good compromise between performance and computational time.

Assuming constant illumination, the computed difference tends to zero for features belonging to the floor, while it has greater values for features on the movable obstacles. With the aim of avoiding too large difference values, all the single differences between pixels $[I_t(u', v') - I_{t-\Delta t}(\bar{u}', \bar{v}')]$ are normalized by dividing them by the maximum difference value within the window.

Then, a similarity value for every match is defined as $(1 - \bar{D})$, where $\bar{D}$ is the normalized difference. This implies that the similarity is between 0 and 1, giving

an idea of the likelihood of each feature correspondence to be part of the ground.

Finally, the similarity value is thresholded and all features whose similarity value is below a threshold are removed. This parameter regulates the amount of features that are rejected and can be optimized at every frame. For the tested sequences, it is set to **0.7**, which shows a good performance.

Figure 3.6 illustrates the final output of the method explained in this chapter. The two images at the top contain all feature matches, with their corresponding estimated optical flow and window. Bottom images show the remaining features after detecting and pruning the movable features and applying the floor mask.

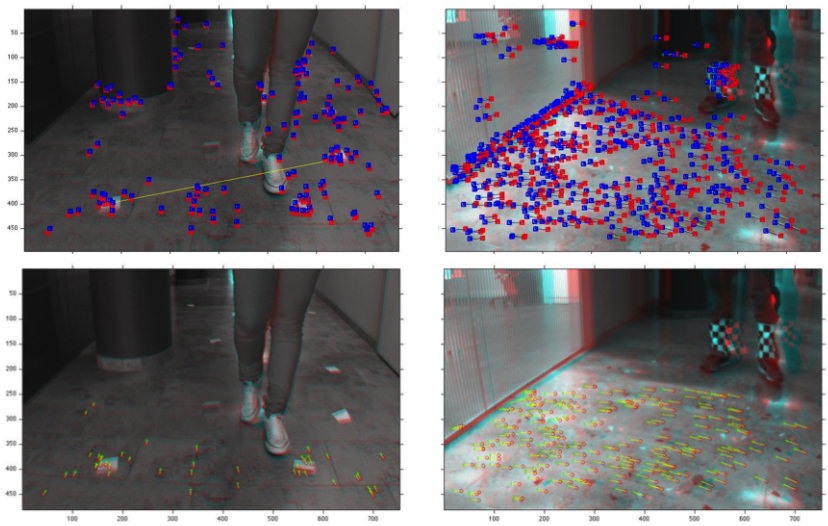


Figure 3.6. Final output for the moving features detection algorithm

The above figure reflects that a high performance is achieved, since almost all the remaining features do belong to the ground. Therefore, it is proved that when the optical flow estimation holds, the implemented method, together with the floor mask, is able to remove the moving features.

Chapter 4

Applying Floor Segmentation in Visual-Aided INS

This chapter details the role that the implemented methods play in a visual-aided inertial navigation system (INS). Floor segmentation and moving features detection are useful in such applications in order to decide whether to use or reject a particular feature. Specifically, both methods have been integrated into the ground plane-based ego-motion estimation block of the visual-aided INS described in [10].

The goal of the INS is to get an estimation of the camera ego-motion along a sequence. After initializing and calibrating the system, different procedures are applied frame by frame in order to get this estimation. Figure 4.1 illustrates a general block diagram of the system.

Four blocks can be distinguished from the diagram: 1) feature extraction and matching, 2) getting floor mask, 3) removing moving features and 4) positioning estimation. To begin with, feature points of the current frame are detected and matched with their corresponding ones in the previous frames. Afterwards, the floor segmentation algorithm computes the floor mask. At this point, the mask is applied in order to keep only the feature points within its region in the current frame. The ego-motion and position of the camera at the previous frame is then used to perform a detection and pruning of the features that belong to movable obstacles. Finally, the positioning estimation block corrects motion parameters from an inertial measurement unit (IMU) to update the ego-motion and position of the camera based on the remaining features.

Next sections describe the four blocks of the updated INS with the aim of showing the reader a real application where the designed methods, explained in Chapter 2 and Chapter 3, are exploited. The necessary adjustments applied to the algorithms in order to fuse them with the INS are also pointed out in the following sections.

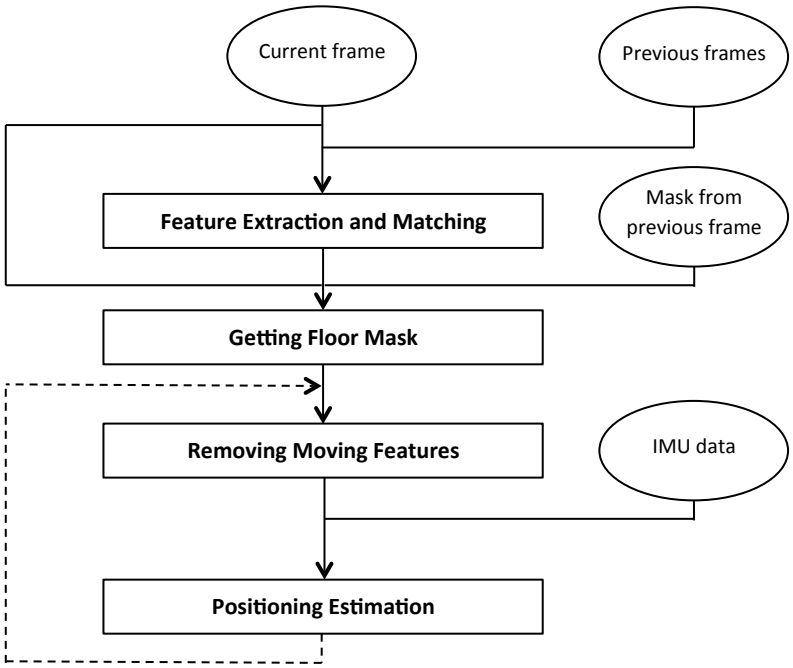


Figure 4.1. Block diagram for the updated INS

4.1. Feature Extraction and Matching

The first block in the analyzed INS consists of extracting feature points from the current frame and establishing correspondences by matching them with those from previous frames. In order to track the features along a sequence, a separation up to **5** frames is used . This block has not been modified from the original implementation in [10].

Speeded-Up Robust Features method (SURF) [32] is the selected algorithm for extracting feature points. MATLAB® Computer Vision Toolbox contains the implementation for both feature detection and matching algorithms.

SURF detector and matcher have parameters to regulate the amount of features extracted and the reliability of the matches. The more features extracted at each frame, the more time consuming the extraction, matching and all further handlings are. Similarly, the more correspondences established, the more likely it is to have mismatches.

The established matches are filtered along next blocks and only a part of them are used to update the camera ego-motion and position. If a correspondence is not established at this point, it cannot be used in the positioning estimation block. Hence, it is essential to get as many correspondences as possible, always keeping in mind the computational cost of processing them.

Figure 4.2 and Figure 4.3 depict examples of the SURF feature extraction and matching. In the generated images, a red circle represents the position of the features in the current frame. Their position in past frames is indicated by a green cross, while the matched correspondences between the pairs of frames are marked in yellow. Figure 4.2 shows four examples of feature correspondences with different separation between frames. The left image in Figure 4.3 contains the extracted features (red crosses) for one particular frame. In the right picture, the features correspondences between the current frame and an image from 3 samples before are illustrated.

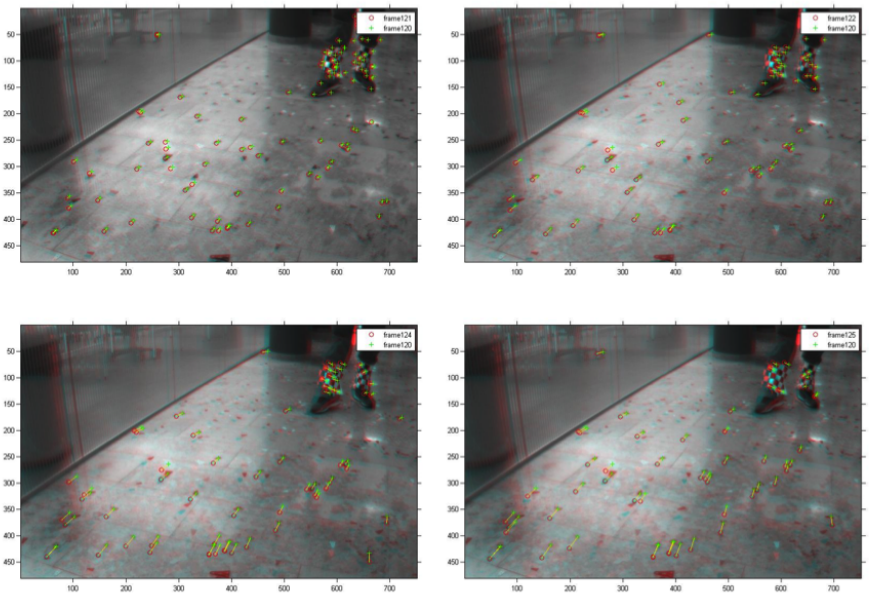


Figure 4.2. Four exemples of feature matching

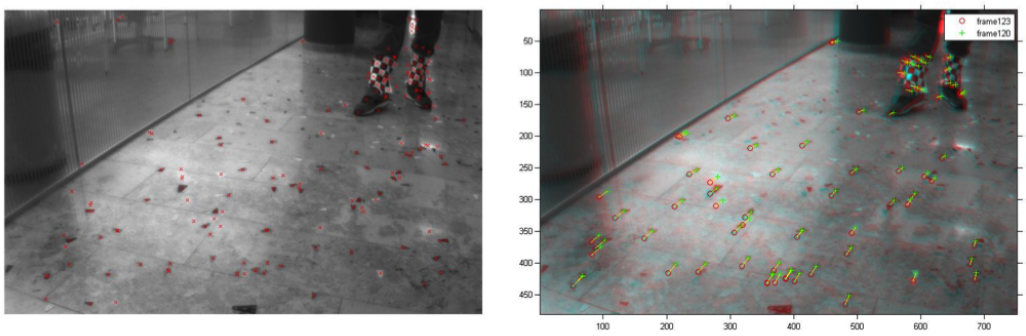


Figure 4.3. Example of feature extraction and matching

4.2. Getting Floor Mask

In this section, the necessity for using the floor mask in the ego-motion estimation part of the visual INS is argued. Besides, the integration of the floor segmentation algorithm in the system is detailed. Some results are presented revealing the improvements introduced by applying the mask compared to the method used before.

The positioning estimation block of the INS needs a certain number of ground feature correspondences in order to update the position and camera ego-motion. Using feature correspondences that are not on the ground plane is counterproductive. Thus, the system must ensure that the considered features do belong to the floor.

Before implementing the developed floor segmentation algorithm, a bounding box defining a square region in front of the camera (see Figure 4.4 - right) was used to decide which features were considered. Although the size of the bounding box could be adapted to every sequence, it was constant along it. It is normal to assume that no obstacle is placed in the region closest to the camera, since the robot navigation system tends to avoid them. Hence, one can find similar methods in the literature [39][40]. However, the use of the bounding box presents some problems:

- The bounding box is defined to be constant along the sequence and thus, it is not able to adapt to changes in the structure of the scene.
- If the region inside the box is too small, many ground features can lie outside it. The ones inside might not be enough to perform the positioning estimation.
- If the bounding box is too large, it might contain features belonging to walls, obstacles or moving objects.

Figure 4.4 illustrates an example where the bounding box is used. The left image depicts all the feature correspondences between a pair of frames. Only the feature correspondences inside the defined bounding box (delimited in red) are represented in the image on the right.

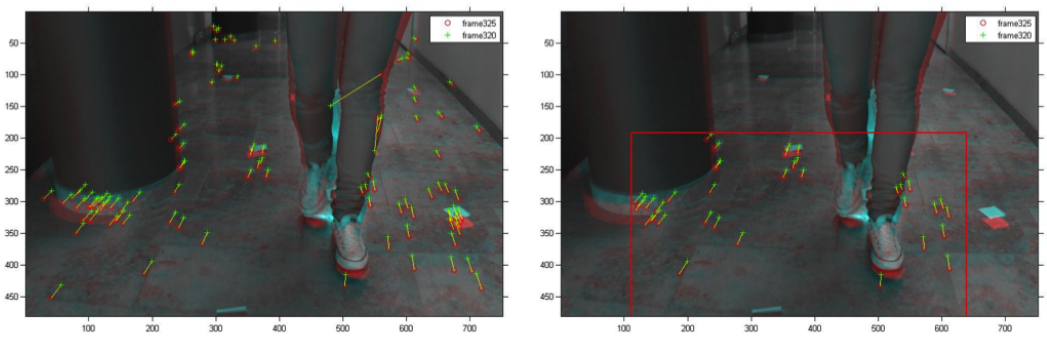


Figure 4.4. Example of using the bounding box

Figure 4.5 compares the performance of using a bounding box or a floor mask. Some of the mentioned problems (see list in 4.2) can be noticed. At the top, all the feature correspondences from two examples are shown. The middle images contain the features inside the defined box, while in the bottom images only the feature correspondences lying within the mask are represented.

The floor mask has been included in the updated INS. However, when the whole floor segmentation algorithm is applied frame by frame to get the mask, the system becomes very slow. A control parameter allows the user to choose whether to use:

- The old bounding box.
- Floor masks from a directory. This means that floor segmentation algorithm must be applied before for the whole sequence.
- Run the floor segmentation algorithm frame by frame and use the extracted mask.

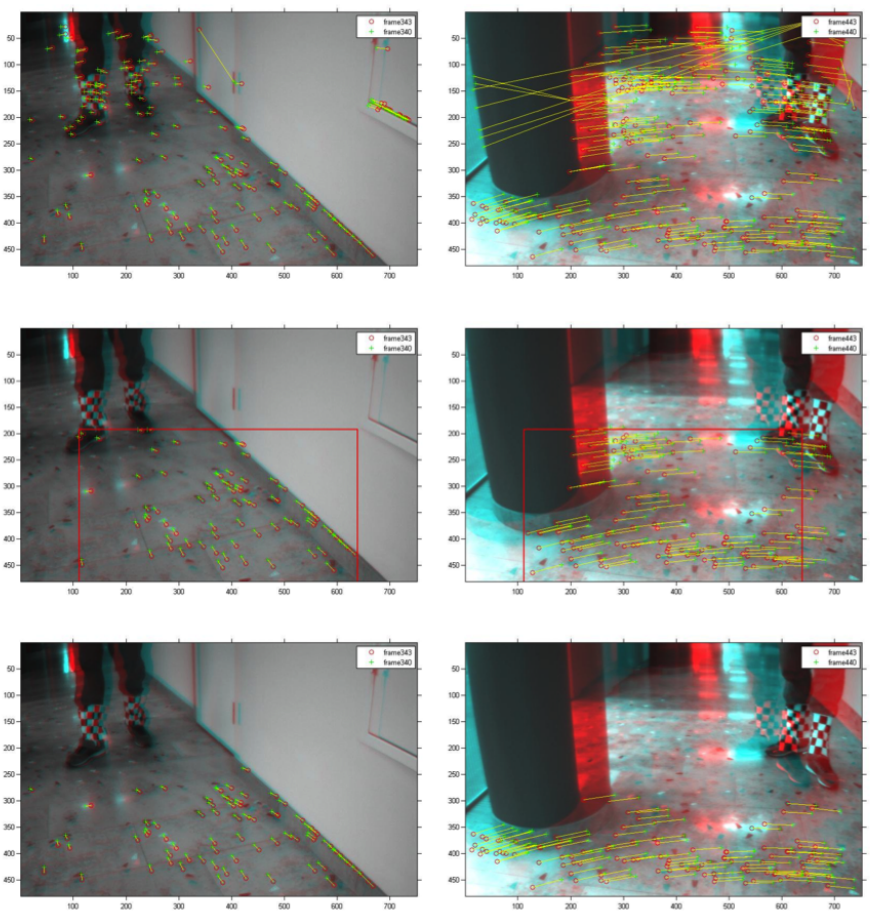


Figure 4.5. Comparison between the bounding box and the floor mask performance

4.3. Removing Moving Features

This section contains the explanation and relevant comments regarding the integration of the detection of movable features algorithm (explained in Chapter 3) into the INS described along this chapter. The goal of this block is to identify and remove movable features, from the set of correspondences between pairs of frames inside the mask in the current frame.

Applying the floor mask is not always enough to ensure that all the remaining features belong to the floor. Although movable objects with well-defined edges are indeed rejected by the floor mask, irregular moving obstacles, such as feet, might partially lie within the mask (see Figure 4.6). To avoid selecting features belonging to movable obstacles, the moving features detection designed method is applied for the features inside the mask.

In Figure 4.6, the position of the features in the current frame is marked with a red circle, while its counterpart in the previous one, with a green cross. A yellow straight line represents the correspondences. The left image contains all the detected feature correspondences, while the one on the right only shows the remaining matches after applying the floor mask. Notice that some of the features in the right image lie on the right foot.

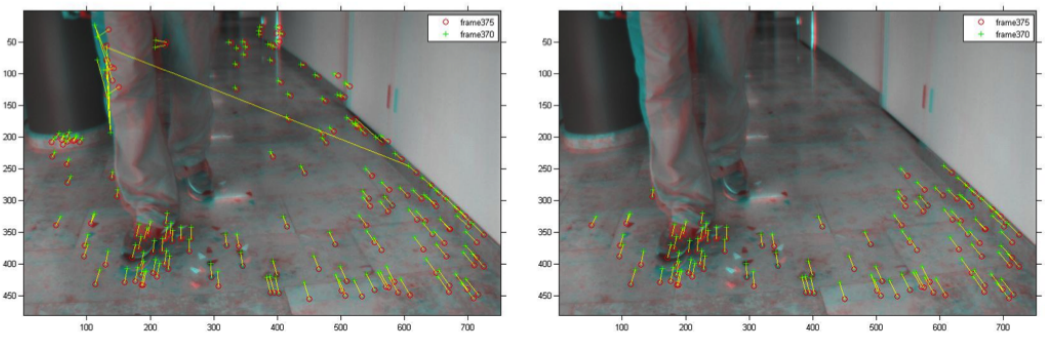


Figure 4.6. Example of features after applying the mask

Homography matrix (for the ground pixels) and optical flow vector (for all the features within the mask) are estimated following the steps explained in Chapter 3. In order to do it, the estimated camera position and ego-motion from the previous frame, as well as the feature positions, are used. The difference between the real and the expected feature position is then compared to detect and prune movable features (see Section 3.3).

As it has been pointed out in Section 3.2, the optical flow estimation has a low effectiveness and fails in almost half of the cases. Consequently, the method has not been introduced into the INS yet, since it might be counterproductive. Before implementing it, one must notice that the feature matching in the INS is not only done to the immediate previous frame, but also to other previous frames within a range (normally up to **5** frames).

Figure 4.7 depicts the final output and the performance of the method, when a correct optical flow estimation is achieved. On the left, the matches with their corresponding optical flow vector (in blue) are plotted; on the right, the remaining features after pruning are represented. Notice that all the features on the foot (see Figure 4.6 - right) have been removed.

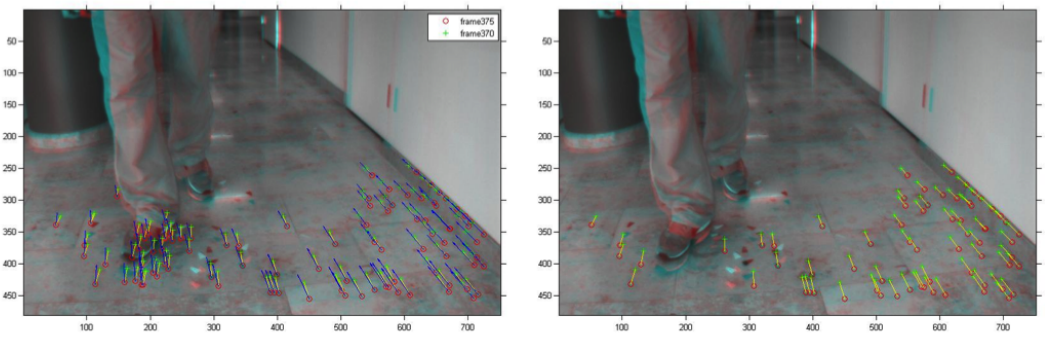


Figure 4.7. Remaining features after detection and pruning

4.4. Positioning Estimation

The last block of the INS exploits the feature correspondences to update the motion parameters from an IMU and estimate the current position and ego-motion of the camera. This part of the system has not been modified, so it works as it did in the original implementation [10]. In order to solve the ego-motion estimation problem, a state-augmented Sigma-point Kalman filter [41] is used.

Figure 4.8 and Figure 4.9 show the final estimated position of the camera along two sequences. The selected coordinate system defines the x-axis pointing to the back of the camera, the y-axis to its sides and the z-axis to the ground. The camera is rigidly mounted on a trolley (see Section 1.4) and thus, the motion in the z axis is negligible. Therefore, only the position in the x and y-axes is plotted.

In both figures, the top image shows the estimated position using the features from the bounding box, while in the bottom image, the features inside the floor mask are considered. Although some small differences can be appreciated, the estimation is very similar. In Figure 4.8, the camera almost describes a straight forward trajectory. On the contrary, Figure 4.9 reveals that the camera is moving forward, but also going from one side to the other. The graphs correspond to sequences *FloorSegmentationFeet2* and *FloorSegment-LateralMotion* in [34], respectively.

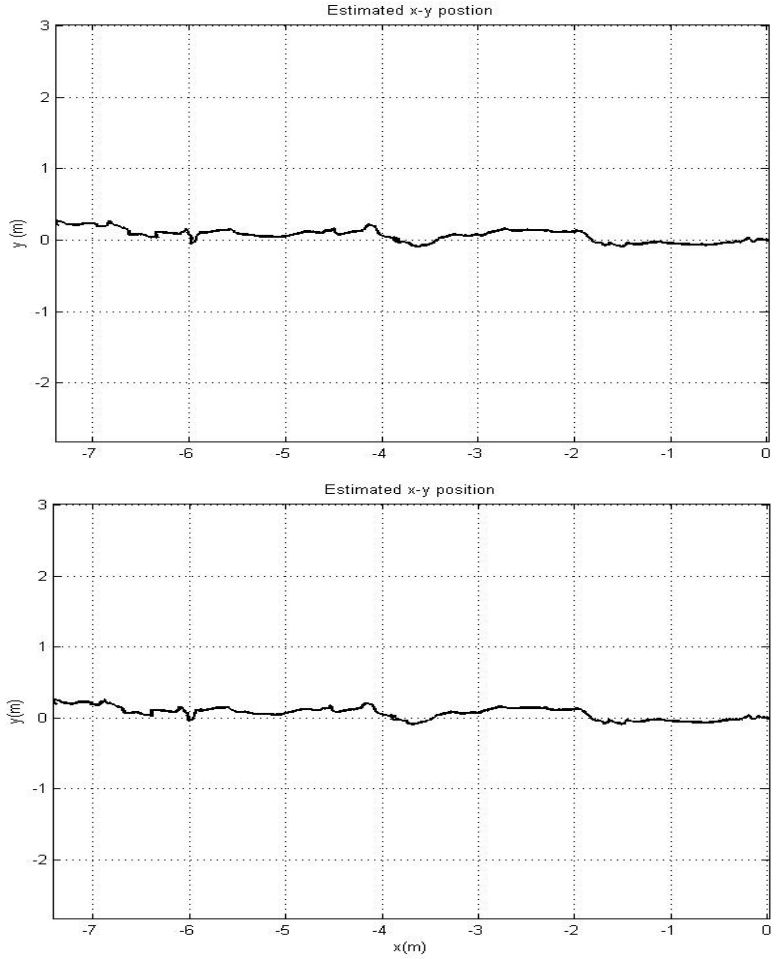


Figure 4.8. Positioning estimation example. Straight motion

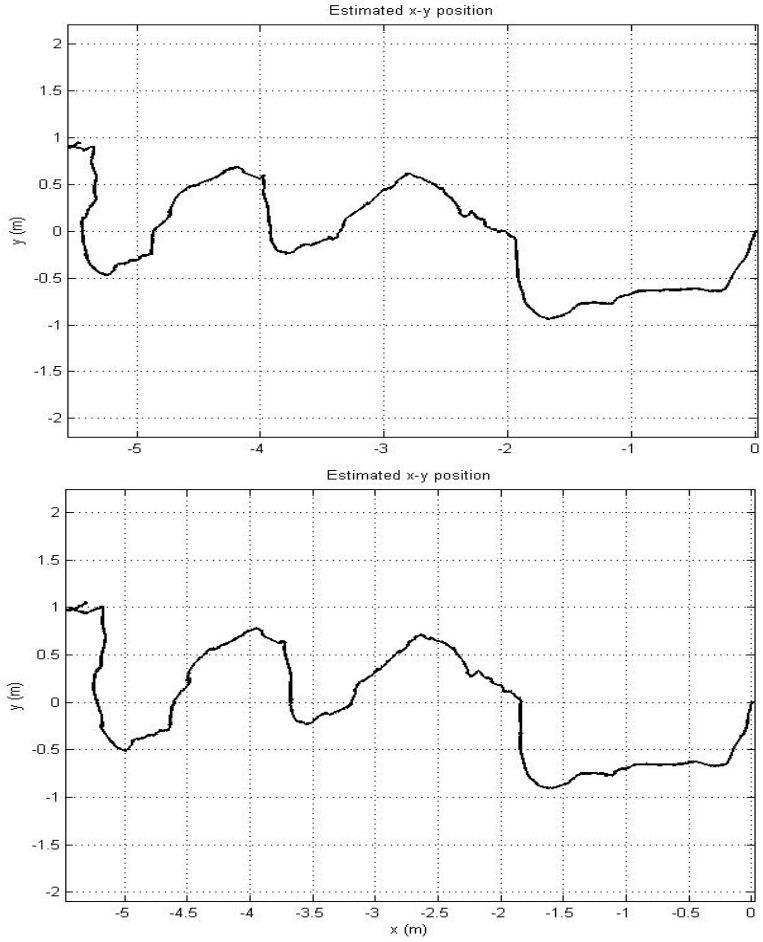


Figure 4.9. Positioning estimation example. Motion on x and y axes

Chapter 5

Results

This chapter shows examples of outputs from the designed methods. Although some results have been revealed all along the previous chapters, the aim of this part is to generalize by presenting more examples together. This way, the reader can qualitatively evaluate the performance of the implemented algorithms.

The presented results are divided into three groups: 1) floor segmentation, 2) optical flow estimation and 3) feature pruning. Since general comments about the performance of each method have already been discussed at the corresponding sections, the analysis here is focused on the behavior for every single sequence.

5.1. Floor Segmentation

In this section, the results of the floor segmentation algorithm for six different sequences are presented. The outputs for several frames along the selected sequences are plotted. In order to easily evaluate the performance of the algorithm, the floor polyline on the top of the original image is displayed. The output videos for all the tested sequences can be found in the *"Floor Segmentation Algorithm"* reproduction list in [34]. The reference in italics that appears at the end of each figure caption corresponds to the assigned name of the tested sequence in the mentioned list.

Figure 5.1 illustrates the results for a sequence that can be considered as a general case. In this example, the main problems faced by the algorithm are the non-homogeneous floor and the specular reflections. These last ones make the ceil lights be strongly reflected, which can cause spurious edges. Moreover, both the geometry of the scene and the camera direction are not constant along the sequence. Despite the difficulties, the performance of the algorithm for the entire sequence is very good.

The floor polyline for a second sequence is shown in Figure 5.2. There are two main interesting characteristics to notice in this example: the detection of static

obstacles (frames 130-145 and 260-280) and the ability to quickly adapt to changes of direction, even when a wall is close to the camera (frames 230-245).

Figure 5.3 reveals that the floor segmentation algorithm is not enough when there are small irregular obstacles, such as feet, at the bottom half of the image. Notice that, even when the person is static; that is during the first frames of the sequence, part of the feet lies below the floor polyline. On the other hand, Figure 5.4 illustrates that, when the person is crossing from one side to the other, the feet are more likely to be detected and thus, rejected from the floor area. In this example, the color of the shoes also helps to detect them easily.

Finally, the two last figures (Figure 5.5 and Figure 5.6) show that the algorithm is able to deal with well-defined objects, even if they are moving. Notice that the box is correctly detected in most of the cases, no matter if it is very close to the camera like in Figure 5.6. For some examples in this last figure (frames 210-230), the box shadow is rejected. Although the floor area is reduced, this can not be considered as a bad behavior, since the shadow might confuse the feature matching algorithm.

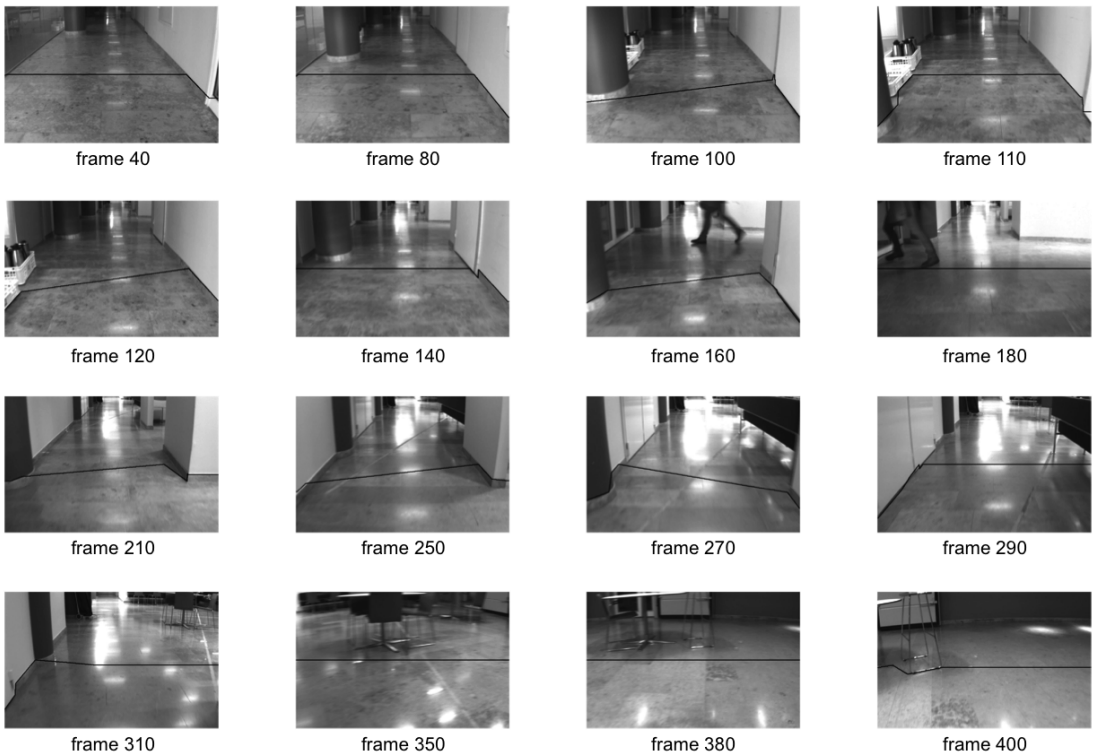


Figure 5.1. Example floor polyline for general indoor sequence - *FloorSegmentSeq1*

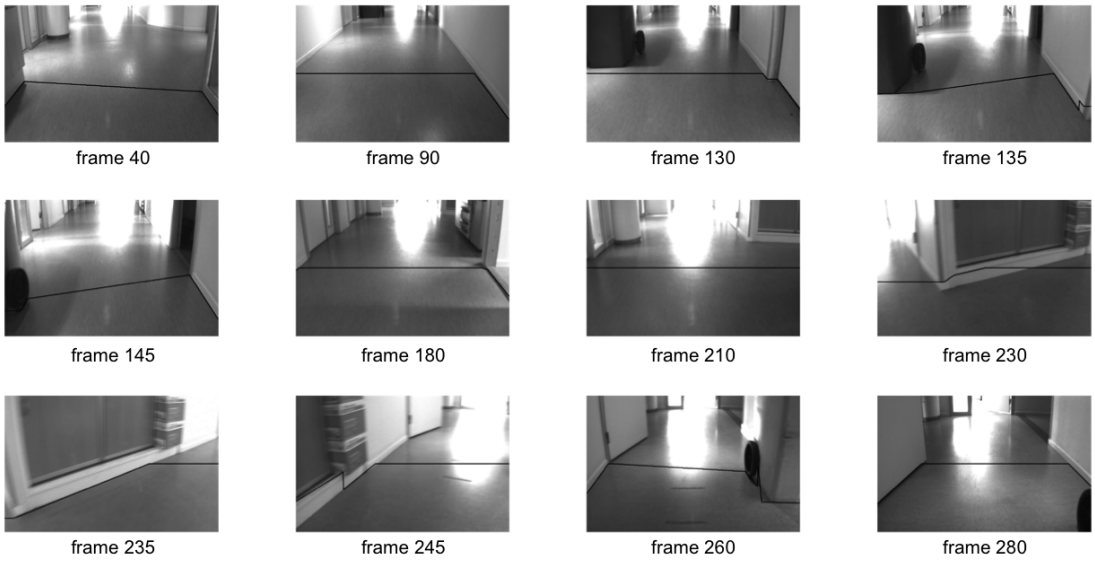


Figure 5.2. Example floor polyline for static obstacles - *FloorSegmentSeq3*

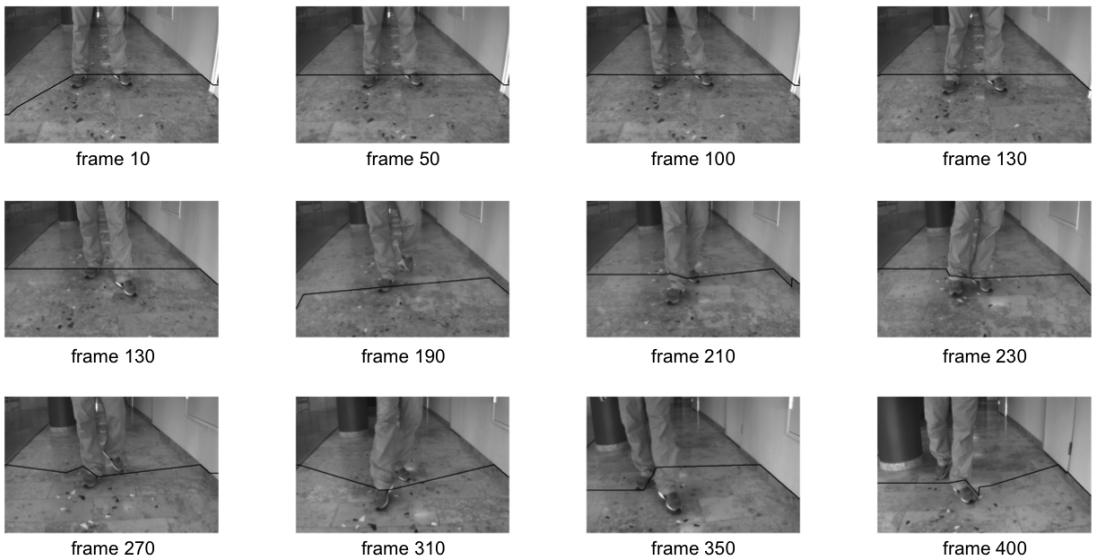


Figure 5.3. Example floor polyline, feet not detected - *FloorSegmentationFeet2*

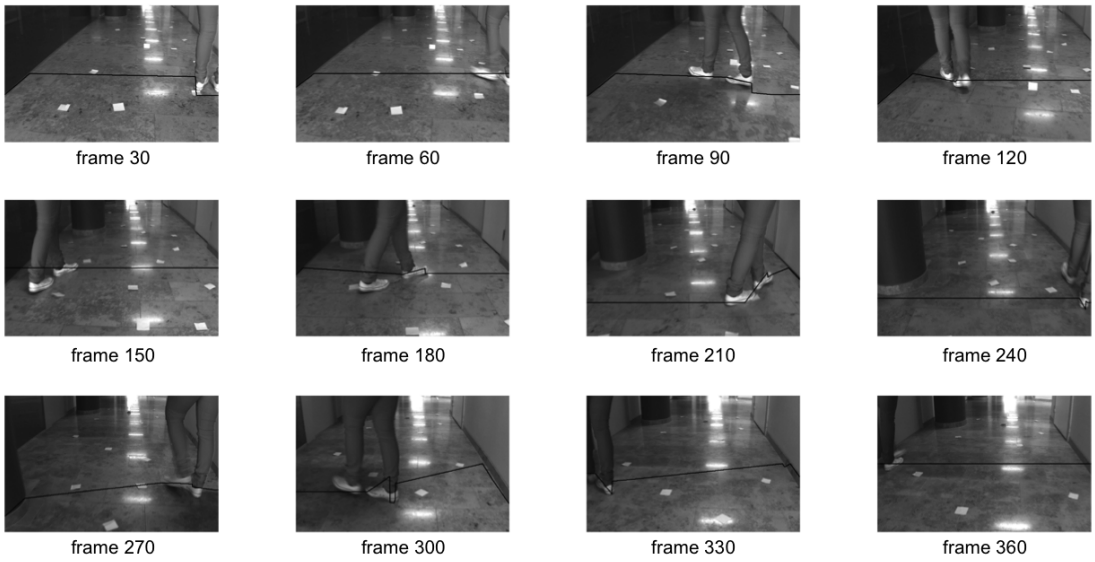


Figure 5.4. Example floor polyline, feet detected - *FloorSegmentationFeet5*

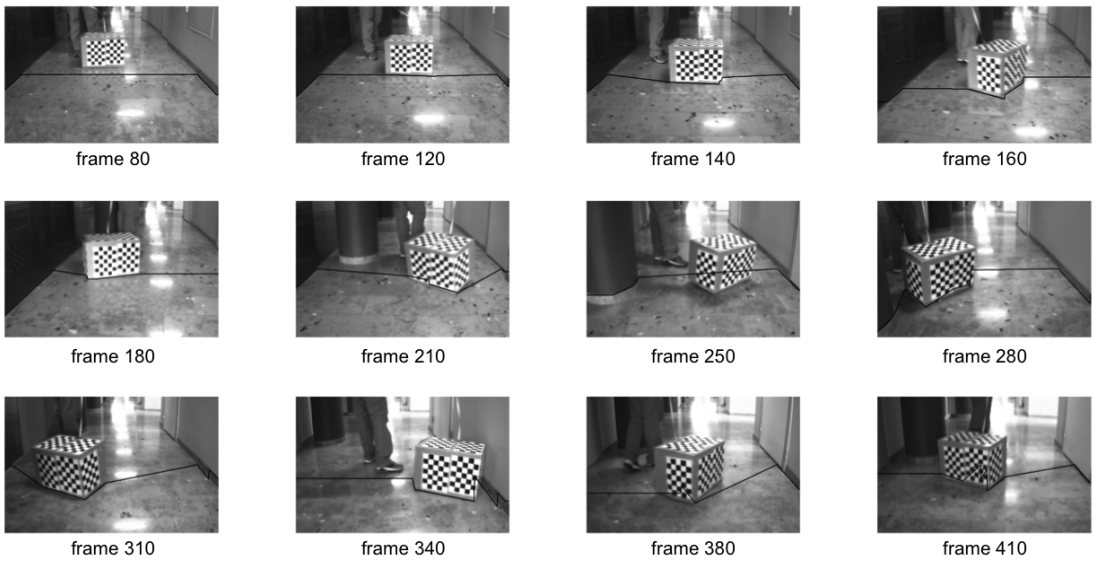


Figure 5.5. Example floor polyline moving box - *FloorSegment-Box2*

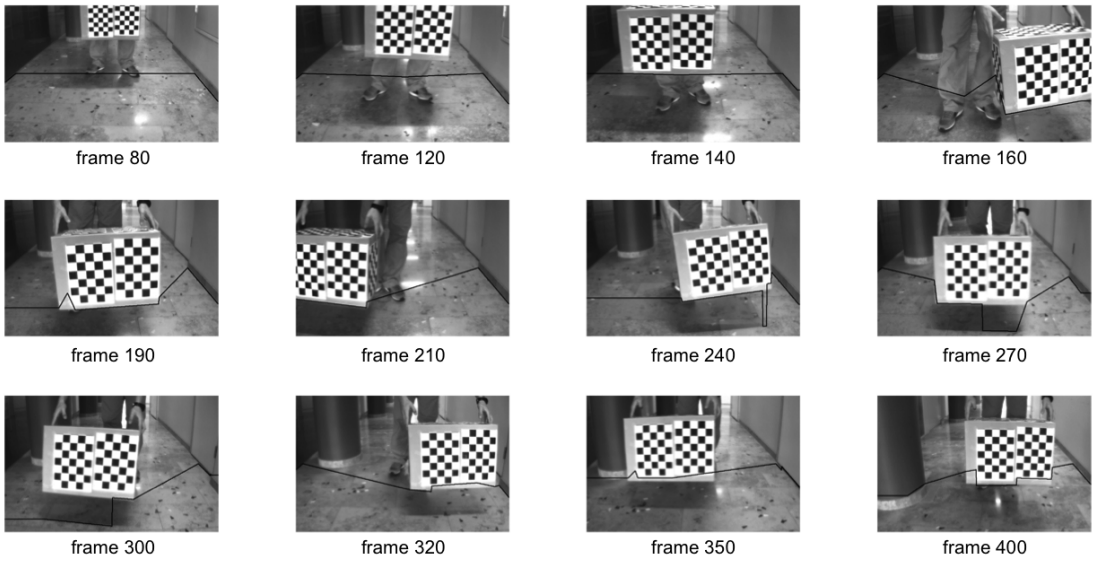


Figure 5.6. Example floor polyline moving box close - *FloorSegment-Box3*

5.2. Optical Flow Estimation

Three sequences are selected as example for optical flow estimation. These sequences are also used in the feature pruning part (Section 5.3) to test the removing moving features method. The separation between the pairs of frames where the feature correspondences are computed is **3** frames in all the cases. Notice that the floor mask is applied before estimating the optical flow and thus, only the features inside the area defined by the mask are considered.

For each image, the analyzed frames appear in the legend at the top right corner of the picture. Red circles indicate the position of features in the current frame, while features position in the previous frame is represented by green crosses. The correspondence vector, joining the current and the past position is plotted in yellow. The estimated optical flow vector is depicted in blue.

Figure 5.7 corresponds to a sequence where the camera moves from one side of the corridor to the other, while for Figure 5.8 the camera motion is mainly straight forward. The sequence represented in Figure 5.9 is chosen because the feet are most of the time inside the floor mask. This way, the ability of the pruning algorithm to remove moving features can be tested in the feature pruning section. The performance of the optical flow estimation is similar in all the cases; around **60%** of the frames are correctly estimated.

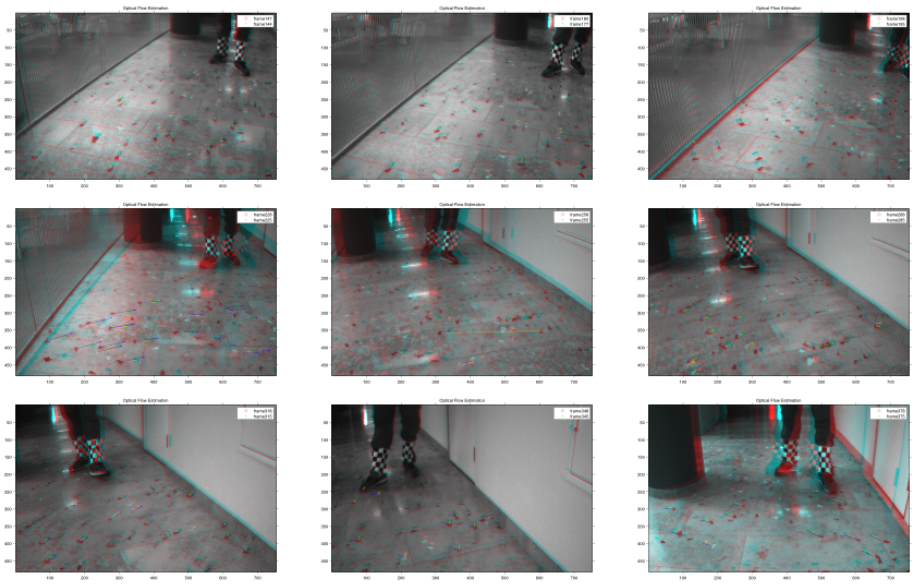


Figure 5.7. Example optical flow estimation sequence 1

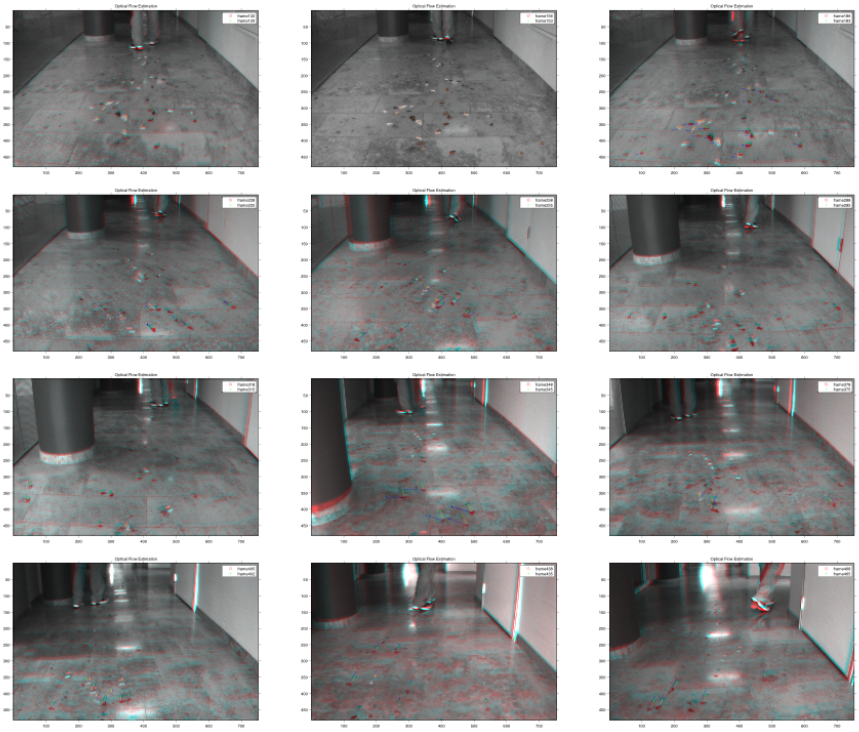


Figure 5.8. Example optical flow estimation sequence 2

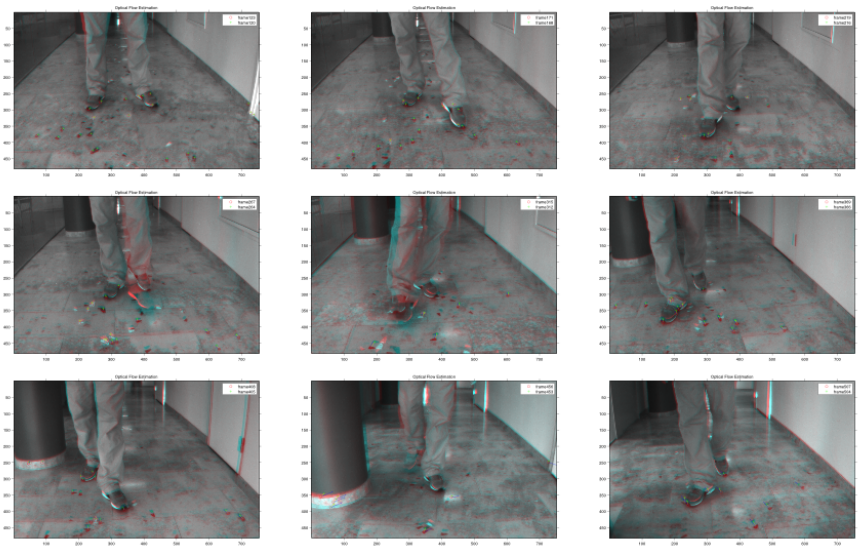


Figure 5.9. Example optical flow estimation sequence 3

5.3. Feature Pruning

This section presents the remaining features after applying the pruning method explained in Setion 3.3. Notice that the sequences used in this section are the same as in the optical flow estimation part (see Section 5.2)

For each image, the analyzed frames appear in the legend at the top right corner of the image. For some few examples, where no feature is left after pruning, the legend does not appear. However, one can recognize the analyzed frames because they are the same as in Figures 5.7, 5.8 and 5.9 from the optical flow estimation section.

Moving features are removed in Figure 5.10 and Figure 5.11 because they are clearly out of the floor mask. Last sequence (Figure 5.12) illustrates that, when the optical flow estimation holds, the pruning algorithm is able to remove most of the movable features, keeping those on the ground plane. On the contrary, if the optical flow estimation is incorrect, notice that the system tends to remove all the features.

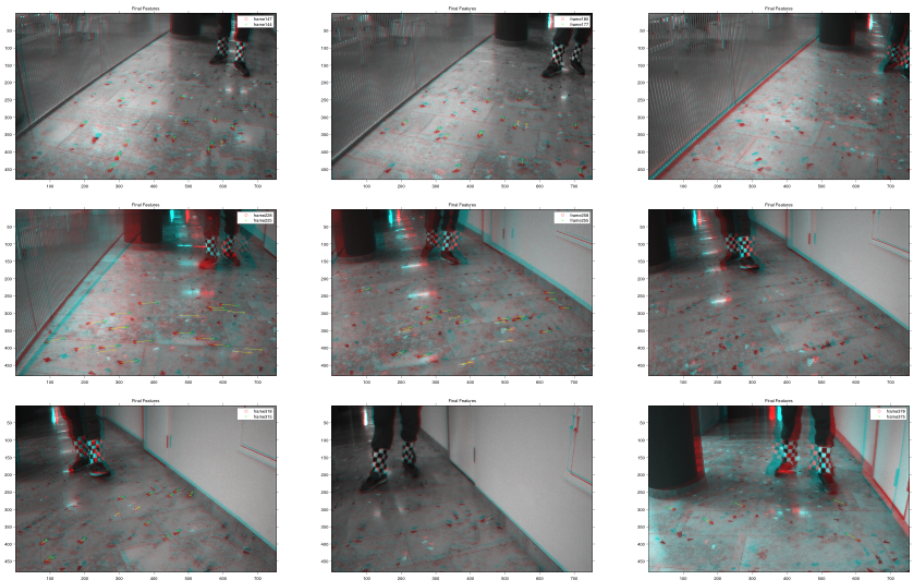


Figure 5.10. Example final features sequence 1

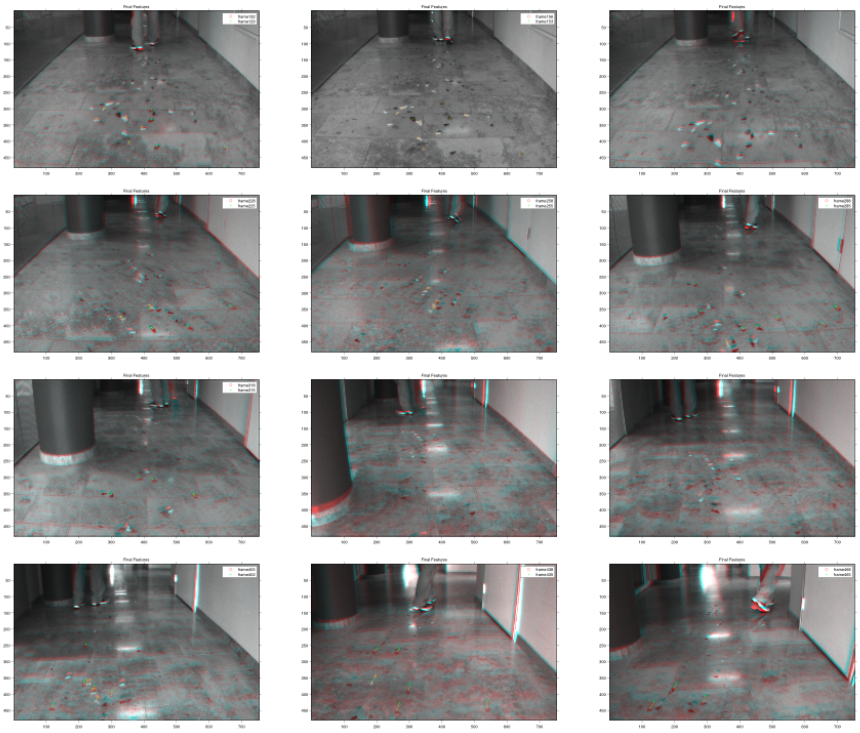


Figure 5.11. Example final features sequence 2

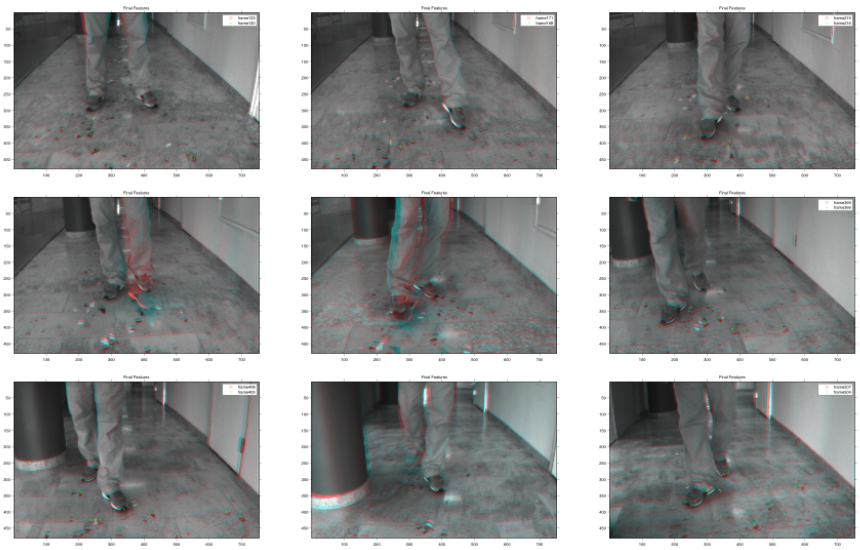


Figure 5.12. Example final features sequence 3

Chapter 6

Conclusions and Future Work

This chapter aims to be a general reflection of the whole Master Thesis project. It is divided into two sections: 1) conclusions and 2) future work. In the conclusions section, the thesis is summarized, its main contributions are stressed and a discussion of the obtained results is presented. Finally, the future work section suggests some ideas for further research.

6.1. Conclusions

This report has presented two new methods to segment the floor and detect moving features. Both algorithms have been integrated into the ego-motion estimation block of a particular visual-aided inertial navigation system (INS). This system combines both visual and inertial information, using a monochrome camera and an inertial measurement unit (IMU).

A noteworthy attribute of the designed technique is its simplicity, since it works with single grey-scale images. In order to define a region containing a sufficient part of the ground with enough feature points, a polyline defining the wall-floor and floor-obstacles boundaries below the half image is drawn. To achieve this goal, an acute way of joining the most important line segments from an edge detection method is applied.

Results have shown that the implemented method succeeds in most of the tested sequences. It can deal with difficult illumination conditions, specular reflections and static obstacles. Moreover, it has been proved to be able to adapt quickly to changes in the scene and motion of the camera. However, it is not sufficient to reject small irregular movable obstacles, such as people feet.

Odometric data from the IMU as well as the estimated position and ego-motion of the camera are used for the designed moving features detector. The movable features are identified because they do not follow the model of the expected optical

flow for the points on the ground plane. This expected motion comes from the derivation of the homography of the ground plane.

One of the main contributions of the implemented method is that no restrictions on the camera motion are introduced in order to estimate the homography. Nevertheless, results have proved that the optical flow estimation only holds in around the 60% of the tested examples. So far, no pattern is detected in the erroneous estimations. Besides, it has been noticed that the optical flow estimation changes from a correct value to a completely incorrect one in a very short period.

6.2. Future work

There is plenty of room for future work related to the research area the project focused on. However, only advances in the two developed methods will be considered in this section. The discussion for the two algorithms is done in separate subsections.

6.2.1. Floor Segmentation Algorithm

One of the strengths of the floor segmentation algorithm presented in this report is that it works for a single image. This simplifies very much the computation, but at the same time, implies that all the operations must be done at every frame of the sequence. With the aim of not repeating the same calculations and reducing the computational cost, information from one frame to the other can be used. Taking into account that the analyzed image is likely to be very similar from one frame to the next one, this proposal is suitable.

Furthermore, extra intelligence could be given to the algorithm for the purpose of time adapting the half image constraint. Depending on the structure of the scene in the past frames, the method would be able to raise or lower this line in order to optimize the floor segmentation.

6.2.2. Moving Features Detection

There are several improvements that can be applied to the moving features detection technique. The most important one is mentioned at the end of Section 3.2. Due to the fact that the optical flow estimation fails in around 40% of the cases, a deep analysis must be carried out to find the causes of this low percentage of effectiveness. One suggestion is to use temporal information in the homography estimation. That is to say that the homography would not be recomputed at every instant, but it would instead be adapted from one frame to the other.

In addition, a test should be realized with the aim of finding the best parameters for SURF detector and matcher (see Section 4.1). It is essential to detect enough features correspondences right at the first stage in order to increase the performance of the positioning estimation.

Finally, once the problems in the optical flow estimation are solved, the designed method is to be integrated to the INS [10] and thus, ensure that no moving features are used to estimate the position of the camera (see Section 4.3).

Bibliography

[1] Yan Meng. A dynamic self-reconfigurable mobile robot navigation system. In *IEEE Advanced Intelligent Mechatronics (ASME), 2005*, pages 1541 –1546.

[2] H. Blaasvaer, P. Pirjanian, and H.I. Christensen. Amor-an autonomous mobile robot navigation system. In *IEEE Systems, Man, and Cybernetics, 1994*, volume 3, pages 2266 –2271.

[3] Siyao Fu, Zeng-Guang Hou, and Guosheng Yang. An indoor navigation system for autonomous mobile robot using wireless sensor network. In *IEEE Networking, Sensing and Control (ICNSC), 2009*, pages 227 –232.

[4] F. Nashashibi, M. Devy, and P. Fillatreau. Indoor scene terrain modeling using multiple range images for autonomous mobile robots. In *IEEE Robotics and Automation, 1992*, pages 40 –46.

[5] D. Conrad and G.N. DeSouza. Homography-based ground plane detection for mobile robot navigation using a modified EM algorithm. In *IEEE ICAR*, pages 910–915, 2010.

[6] Ghazaleh Panahandeh, Nasser Mohammadiha, and Magnus Jansson. Ground plane feature detection in mobile vision-aided inertial navigation. In *IEEE Intelligent Robots and Systems (IROS)*, 2012.

[7] Yinxiao Li and S.T. Birchfield. Image-based segmentation of indoor corridor floors for a mobile robot. In *IEEE Intelligent Robots and Systems (IROS), 2010*, pages 837 –843.

[8] Xiaojing Song, L.D. Seneviratne, and K. Althoefer. A kalman filter-integrated optical flow method for velocity sensing of mobile robots. *IEEE Mechatronics*, 16(3):551 –563, 2011.

[9] C. Hide, T. Botterill, and M. Andreotti. Low cost vision-aided imu for pedestrian navigation. In *IEEE Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010*, pages 1 –7.

[10] G. Panahandeh, D. Zachariah, and M. Jansson. Exploiting ground plane constraints for visual-inertial navigation. In *IEEE Position Location and Navigation Symposium (PLANS), 2012*, pages 527 –534.

[11] T. Wekel, O. Kroll-Peters, and S. Albayrak. Vision based obstacle detection for wheeled robots. In *IEEE Control, Automation and Systems (ICCAS), 2008*, pages 1587 –1592.

[12] P. Lombardi, M. Zanin, and S. Messelodi. Unified stereovision for ground, road, and obstacle detection. In *IEEE Intelligent Vehicles Symposium, 2005*, pages 783 – 788.

[13] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *IEEE Robotics and Automation (ICRA), 2004*, volume 1, pages 592 – 597.

[14] L.M. Lorigo, R.A. Brooks, and W.E.L. Grimsou. Visually-guided obstacle avoidance in unstructured environments. In *IEEE Intelligent Robots and Systems (IROS), 1997*, volume 1, pages 373 –379.

[15] Hui Wang, Kui Yuan, Wei Zou, and Yizhun Peng. Real-time obstacle detection with a single camera. In *IEEE Industrial Technology (ICIT), 2005*, pages 92 – 96.

[16] Iwan Ulrich and Illah Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *Proceedings of AAAI 2000*, 2000.

[17] Young geun Kim and Hakil Kim. Layered ground floor detection for vision-based mobile robot navigation. In *IEEE Robotics and Automation (ICRA), 2004.*, volume 1, pages 13 – 18.

[18] Xue-Nan Cui, Young-Geun Kim, and Hakil Kim. Floor segmentation by computing plane normals from image motion fields for visual navigation. *International Journal of Control, Automation and Systems*, 7:788–798, 2009. 10.1007/s12555-009-0511-2.

[19] Boyoon Jung and Gaurav S. Sukhatme. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. In *in International Conference on Intelligent Autonomous Systems*, pages 980–987, 2004.

[20] J.M. Odobez and P. Bouthemy. Detection of multiple moving objects using multiscale mrf with camera motion compensation. In *IEEE Image Processing (ICIP), 1994*, volume 2, pages 257 –261.

[21] A. Behrad, A. Shahrokni, S. A. Motamedi, and K. Madani. A Robust Vision-based Moving Target Detection and Tracking System. In *Image and Vision Computing conference*, 2001.

[22] J. Klappstein, F. Stein, and U. Franke. Monocular motion detection using spatial constraints in a unified manner. In *IEEE Intelligent Vehicles Symposium, 2006*, pages 261 –267.

[23] C. Braillon, C. Pradalier, J.L. Crowley, and C. Laugier. Real-time moving obstacle detection using optical flow models. In *IEEE Intelligent Vehicles Symposium, 2006*, pages 466 –471.

[24] C. Braillon, K. Usher, C. Pradalier, J.L. Crowley, and C. Laugier. Fusion of stereo and optical flow data using occupancy grids. In *IEEE Intelligent Transportation Systems Conference (ITSC), 2006*, pages 1240 –1245.

[25] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Computer Vision and Pattern Recognition (CVPR), 1997*, pages 731 –737.

[26] S. Maji, N.K. Vishnoi, and J. Malik. Biased normalized cuts. In *IEEE Computer Vision and Pattern Recognition (CVPR), 2011*, pages 2057 –2064.

[27] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 26(5):530 –549.

[28] N. Pears and Bojian Liang. Ground plane segmentation for mobile robot visual navigation. In *IEEE Intelligent Robots and Systems (IROS), 2001*, volume 3, pages 1513 –1518.

[29] D.C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *IEEE Computer Vision and Pattern Recognition (CVPR), 2009*, pages 2136 –2143.

[30] Zhichao Chen and S.T. Birchfield. Visual detection of lintel-occluded doors from a single image. In *IEEE Computer Vision and Pattern Recognition (CVPR), 2008*, pages 1 –8.

[31] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, 1986.

[32] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[33] Matlab - MathWorks official website. Available from: http://www.mathworks.se/products/matlab/.

[34] Youtube channel with tested sequences. Available from: http://www.youtube.com/channel/UC3NXaWEQBIZXsJAtNvBXSeg/videos?flow=grid&view=1.

[35] Paul Hough. Method and means for recognizing complex patterns. U.S. Patent 3.069.654, dec 1962.

[36] H. Aghajan and A. Cavallaro. *Multi-Camera Networks: Principles and Applications*. Number ISBN: 978–0–12–374633–7. Elsevier, 2009.

[37] Matthew Barth Jay A. Farrell. *The Global Positioning System and Inertial Navigation*. McGraw-Hill Professional, 1st edition, 1998.

[38] S. Negahdaripour. Revised definition of optical flow: integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Pattern Analysis and Machine Intelligence*, 20(9):961 –979, 1998.

[39] P. Lombardi, M. Zanin, and S. Messelodi. Unified stereovision for ground, road, and obstacle detection. In *IEEE Intelligent Vehicles Symposium, 2005.*, pages 783 – 788.

[40] Iwan Ulrich and Illah Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *Proceedings of AAAI 2000*, 2000.

[41] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. pages 182–193, 1997.