University of Pisa

Master Degree in

Robotics and Automation Engineering

Range estimation and obstacle detection for unmanned aircraft vehicles: a stereo vision approach

Supervisors

Prof. Alberto Landi

Prof. Emanuele Crisostomi

Eng. Andrea Masini

Student

Leonardo Sestini

Academic Year 2012-2013

A nonna Bone, babbo Rolando e mamma Paola
Grazie

Abstract

The objective of this work is to implement algorithms for object detection and range estimation that can be applied to unmanned aircraft vehicles.
To tackle this problem, stereo vision theory and well-known object detection algorithms are investigated and compared.
Object detection algorithms are then combined with the stereo vision system to obtain two new algorithms that improve the performance of the simple vision system.
These two algorithms are finally tested and compared in real and simulated scenarios.

Sommario

Lo scopo di questo lavoro è stato quello di implementre algoritmi per la rilevazione di ostacoli e la stima della distanza da quest ultimi, da poter applicare a velivoli autonomi senza pilota.
A tal fine sono state studiate, confrontate ed esaminate le più importanti tecniche di object detection e di visione stereo.
Sono quindi stati implementati due nuovi algoritmi che combinano le tecniche studiate al fine di ottenere un sistema di rilevazione/stima accurato e preciso
Tali algoritmi sono poi stati testati e confrontati sia in un ambiente virtuale che in uno scenario reale.

Contents

CHAPTER 1

Introduction

In the last years, researches on unmanned air vehicles (UAV) have significantly increased, due to the variety of applications where UAVs can be used with different tasks. Small UAVs are primarily used for low altitude surveillance tasks, such as forest fire tracking, civilian search and rescues, military reconnaissances, convoy support and other military operations in urban terrain. Paths for small UAVs are planned by human operators or by automated path planners. Thanks to UAVs it is possible to complete tasks in very difficult environments without any risk for the people that control the UAVs. Unfortunately, there are lots of aspects and problems that need to be studied in order to have safe autonomous vehicles.

One of the most important problems about UAVs is how they can be integrated into a non-segregated airspace in a multi-aircraft and manned flight environment, e.g. an airport, without impacting airport safety and efficiency.
The system must necessarily satisfy:

- **Safety**
  - Ensure safe task execution using a Detect & Avoid (D&A) system compatible with existing safety nets and operating procedures, including air traffic management.

  - Airspace management procedures, including air traffic management strategies

- **Capacity and efficiency**

  - Exchange trajectory data and other information with Air Traffic Controller (ATC), takes into consideration the latencies and uncertainty of trajectories
  - Address alternative specific but interoperable surveillances, communications and navigation issues

- **Airport Integration and Airspace efficient throughput**

  - Demonstrate airport surface operations capabilities, including interactions with other aircraft on the surface as well as with ground vehicles and obstacles
  - Demonstrate take-off and landing capabilities without impacting airport throughput
  - Demonstrate D&A for ground operations, taking into consideration wind turbulence and weather conditions
  - Quantify minimum performance requirements for integration like speed, climb/descent and turn performance, requirements according to different airport complexity types.

- **Security**

  - Identify security threats to UAVs integration in non-segregated airspace, that can lead to hazards in terms of loss of control, communication, navigation or surveillance capabilities.

For all the previous reasons, research is very active, in order to find control algorithms and sensors that can satisfy the previous constraints.

A variety of sensors could potentially be used to navigate around unknown obstacles and non-structured environments. RADAR is one of the

most used sensors [1], that can be used in all weather operations with high resolution, but it is mainly used only for detecting obstacles within a close range due to its sparse measurements and its low scan rate. Furthermore, it is sometimes necessary to work with small UAVs (because it is often necessary to work in narrow environments), and light-weight and mechanically simple solutions are required. Additionally, passive sensors are preferable in military operations because active sensors signals can be detected and jammed or may interfere with nearby vehicles or similar sensor signals.

To reduce the possibility of mid-air collisions the Federal Aviation Administration has developed the Traffic Alert and Collision Avoidance System or TCAS [2]. This airborne system senses the presence of nearby aircraft by queering the transponders carried by these aircraft. This system only prevents aircraft collisions with other aircraft, but not with any other kinds of obstacles. Furthermore, TCAS system does not satisfy light-weight constraints.

An alternative solution is to use video cameras. Video cameras provide lightweight, simple, passive and inexpensive solutions, and are therefore an attractive sensor for small UAVs. However, significant computer video processing it is required, in order to obtain useful information. With video camera and image processing is possible to detect all kind of objects, calculate distance from obstacles and satisfy previous constraints. On the other hand this kind of implementation cannot be used in all weather conditions, but it is possible to improve its performance using image processing techniques (e.g. using infra-red cameras, video fusion and so on).

The main contribution of this work is to improve the classical techniques of object detection, applied in the UAVs case. This new approach combined together stereo vision methods and some image processing techniques. In this way it is possible to detect obstacles, from simulated or real scenarios, and to calculate the distance between UAVs and founded objects.

Chapter 2 presents the basic theory of stereo vision, and how it can

be used for range estimation. Chapter 3 describes some techniques that can be used in order to improve stereo vision basic systems, in order to extract obstacles from the acquired images and resolve some problems that may occur. The image processing techniques, previously illustrated, will be implemented in two new algorithms (called Ste.Vi.E and Ste.Vi.S) and compared in Chapter 5. Finally in Chapter 6 the obtained results will be discussed and possible future developments are outlined.

CHAPTER 2

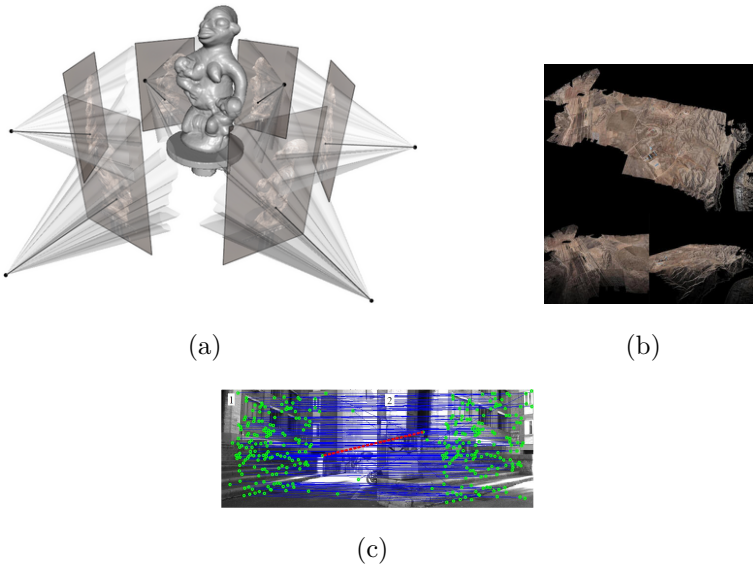
Stereo Vision

2.1 Introduction

This chapter presents the fundamental of stereo vision and describes the most important concepts and techniques used to build a stereo vision system.

Researchers in computer vision have been developing mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery. We now have reliable techniques for accurately computing a partial 3D model of an environment from thousands of partially overlapping photographs Fig. 2.1(b). Given a large set of views of a particular object, we can create accurate dense 3D surface models using stereo matching Fig. 2.1(a). It is also possible tracking something in the scene and so on. Computer vision tries to describe the world captured, in one, or more, images and to reconstruct its properties, such as shape, illumination, and color distributions. Computer Vision is now used in a wide variety of real-world applications, for example:

- **Optical character recognition:** reading handwritten codes on letters and codes in commercial products
- **Machine inspection:** rapid parts inspection in industrial and automotive product lines



(a) (b)

(c)

Figure 2.1: *Example of 3D resortuction, overlapping of more pictures and stereo matching*

- **3D model building:** automated construction of a 3D model of an object
- **Medical imaging:** that can help in medical operations
- **Automotive safety:** detecting unexpected obstacles such as pedestrians on the street, under conditions where active vision techniques such as radar not work well
- **Surveillance:** monitoring for intruders, analyzing highway traffic and monitoring pools for drowning victims.

Stereo vision is an important part of computer vision, that aims to capture information from a pair of images that can not be deduced with a single image. With a single camera, (configuration in figure 2.2), we have only an optical center, so two aligned points are projected in a same image point. If we are able to find corresponding (homologous) points in two, or more, images, is possible to infer depth by means of triangulation . One of the most important problems in stereo vision is the stereo correspondence. Sometimes

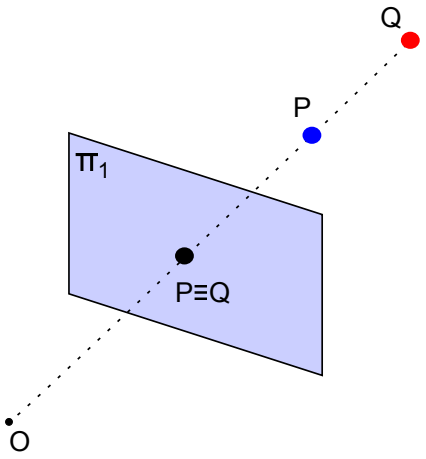


Figure 2.2: *Two aligned points real (P, Q) projected in the same image point, where "O" represent optical center and "π" is the image plane*

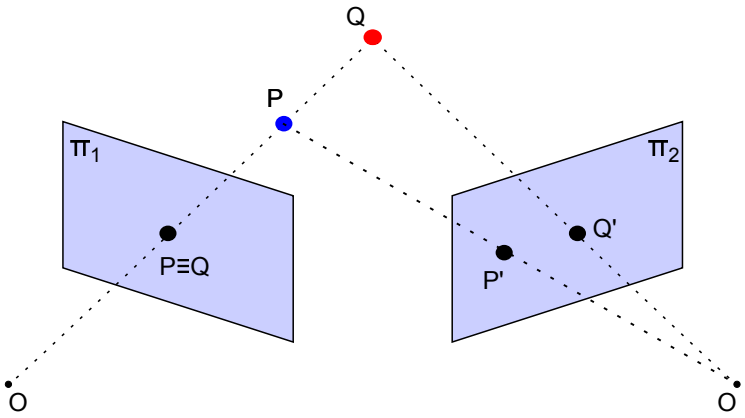
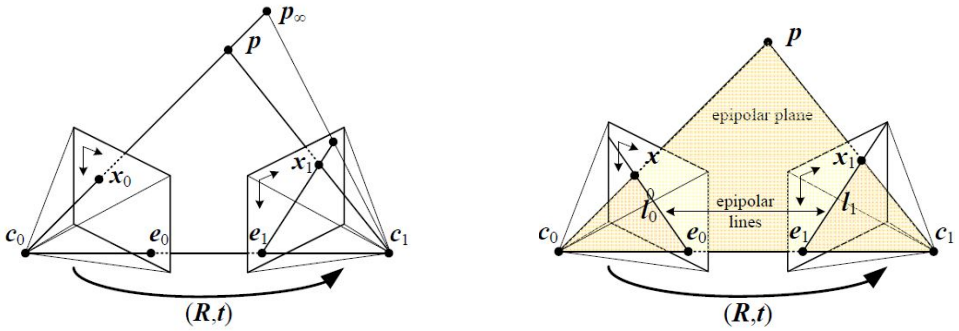


Figure 2.3: *Stereo vision with two cameras, and projection of two real point in different image plane*

can be difficult, or impossible, to find the projection of one real point in different image planes.

2.2 Epipolar Constraints

This problem can be resolved introducing *Epipolar constraints*. If we consider two points P and Q, in the same line of sight (on the same reference image), the epipolar constraints state that the points in a line of view of the first camera, lies in a particular line in the image plane of the second camera. Taking images at different time it is possible to use epipolar constraints with a single camera too. This information reduces the number of potential correspondences. Figures 2.4(a) shows how a real point (for example P),



(a) (b)

Figure 2.4: *Epipolar constraint (a) and epipolar plane (b)*

in a line of view from c_0 to p_∞, is projected in a particular line (*epipolar segment*) in the second image plane. The segment is bounded at one end by the projection of the original viewing ray at infinity p_∞ and at the other end by the projection of the original camera center c_0 into the second camera, which is known as the epipole e_1. This is valid for the first image plane obtaining epipole e_0. Extending both line segments to infinity, we get a pair of corresponding epipolar lines (Fig.2.4(b)), which are the intersection of the two image planes with the *epipolar plane* that passes through both camera

centers c_0 and c_1, as well as the point of interest P. In Fig. 2.4[1] the two image planes are referred at the same camera but after a roto-translation $(R, T) \in SE(3)$. The general formulation of them is the following:

$$y_t^{iT} T_{c_\tau c_t} \wedge (R_{c_\tau c_t} y_\tau^i) = 0 \tag{2.1}$$

where y_t^i is the vector containing the normalized coordinates of i^{th} feature of the image taken at time t, y_τ^i is the correspondent point in image taken at time τ (with $\tau < t$) and $(R_{c_t c_\tau}, T_{c_t c_\tau})$ represents the roto-translation matrix between camera position at time t w.r.t. its position at time τ.

2.3 Camera Model and Projective Geometry

One of the most important problems is to find the relationship between a point Q_i in the physical world, with coordinates (X_i, Y_i, Z_i), to the point q_i on the projection screen with coordinates (x_i, y_i). The relation that maps points Q_i with points q_i is called *Projective Transformation*. In this case, the image plane is the projective space and has two dimensions, so it is possible to represent points on that plane as three-dimensional vectors $q = [q_1, q_2, q_3]$ (because it is convenient to use homogeneous coordinates). All points having proportional values in the projective space are equivalents (due to homogeneous coordinates), so we can recover the actual pixel coordinates dividing through by q_3. This allows to arrange the parameters that define our camera into a single 3-by-3 matrix that will be call *Intrinsic camera matrix*.

So the projection of the points in the physical world into the camera can be now summarized by the following relation:

$$p = MP, \quad \text{where} \quad p = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

As we can see Camera Matrix M contains some important parameters that characterize the camera. There are two different *focal lengths*, f_x and

[1] Fig.Ref [3]

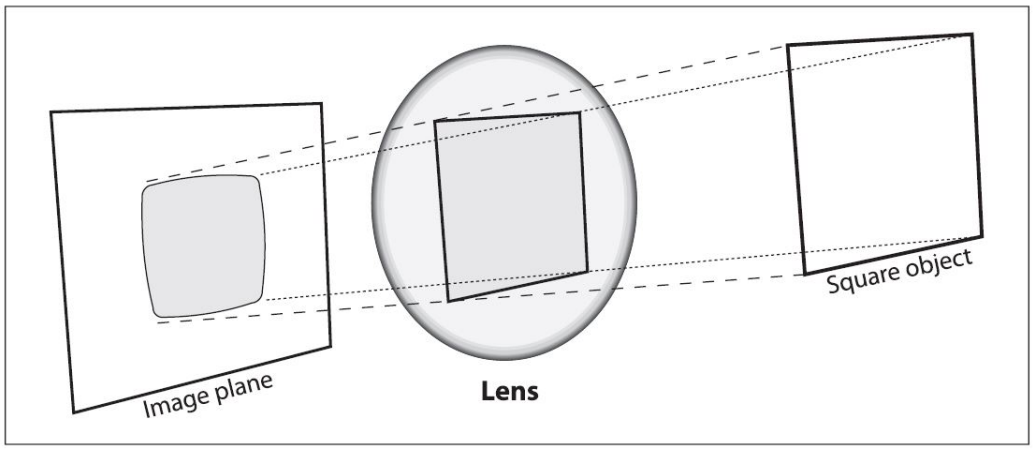


Figure 2.5: *Effect of radial distortion*

f_y, because the individual pixels, in a typical low cost camera, are rectangular rather than square. Each individual focal length is obtained by the product of the physical focal length F with the size s_x or s_y of the individual imager element (e.g. $f_x = F \cdot s_x$).

Sensor chip is usually not on the optical axis, due to unavoidable construction errors, so it is necessary to introduce two new parameters c_x and c_y that help to model a possible displacement (away from the optical axis) of the center of coordinates on the projection screen. The relationship between 3D point $Q_i = [X_i, Y_i, Z_i]$ and the projection on the screen (x_i, y_i) is given by:

$$x = f_x \left(\frac{X}{Z}\right) + c_x, \qquad y = f_y \left(\frac{Y}{Z}\right) + c_y \tag{2.3}$$

Another important aspect to consider, is the distortion introduced by camera lens, because no lens is perfect. It is possible to introduce two kinds of distortions: *Radial Distortion*, as a result of the shape of lens and *Tangential Distortion* arise from the assembly process of the camera.

The effect of radial distortion changes the pixels position near the edges of the image. This bulging phenomenon is the source of the "barrel" or "fish eye" effect. This effect is clearly shown in Fig 2.5[2]. The introduced distortion

[2] `Fig.Ref [4]`

is zero at the (optical) center of the image and increases moving toward the periphery. It's possible to characterized this effect by the first terms of Taylor's series expansion around optical center ($r = 0$). The corrected image coordinates will be:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 ...) \tag{2.4}$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 ...) \tag{2.5}$$

where ($x_{corrected}$, $y_{corrected}$) are the new pixel coordinates, r is the distance from the center and $(k_1, k_2, ..., k_n)$ the terms introduced by Taylor expansion (usually arrested at 3^{th} term).

The Tangential distortion derives from manufacturing defects, resulting from the lens not being exactly parallel respect the image plane. This effect is characterized by two additional parameters, p_1 and p_2, such that:

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)] \tag{2.6}$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 x] \tag{2.7}$$

All the effects introduced by distortions are described by these five coefficients $[k_1 k_2, k_3, p_1, p_2]$, that will be estimated by the calibration process as explained in section 2.4.

The 2D points determined in the previous steps can be also re-projected into three dimensions given their screen coordinates and the camera intrinsics matrix. The re-projection matrix is:

$$P = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & \frac{(c_{x_l} - c_{x_r})}{T_x} \end{bmatrix}$$

All the parameters contained in P are from the left image except for c_{x_r}, which is he principal point x coordinate in the right camera. Given the 2D homogeneous point and its associated disparity d, we can project the point

in the three camera dimension space as:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & \frac{(c_{x_l} - c_{x_r})}{T_x} \end{bmatrix} \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}$$

If the pose of the stereo rig, respect the real world coordinate system, is known is possible to georeference all the points in the world coordinate system.

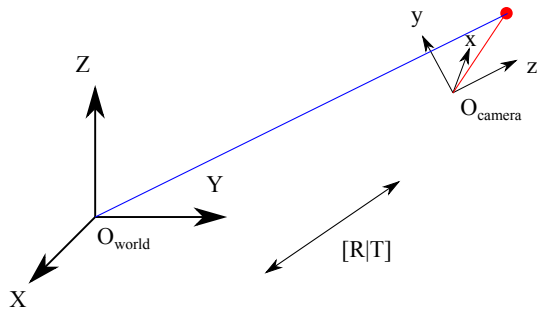


Figure 2.6: *Pose of camera coordinates system in the world coordinate system*

From IMU and GPS information is possible to calculate the transformation from world coordinates to camera coordinates:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = [R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

2.4 Calibration

Before starting acquisition is necessary to find some important parameters that characterize the stereo pair cameras. Calibration is an off-line technique that aims to find two kinds of parameters:

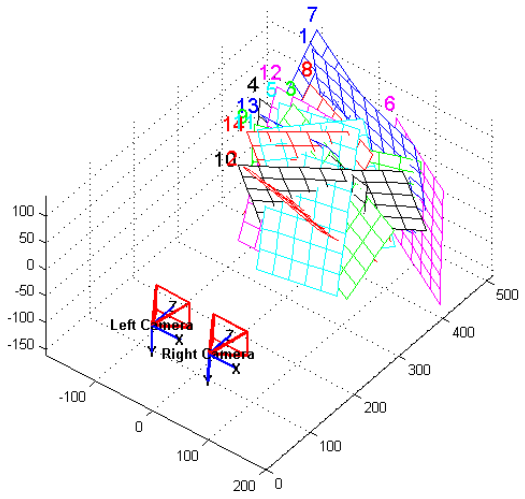


Figure 2.7: *Stereo calibration with* `Matlab` *Toolbox*

- **Intrinsic Parameters:** these parameters allow to find the transformation from a point in a 3D space in the image plane coordinates. For example focal length, image center, distortion and so on.
- **Extrinsic Parameters:** that represent the position of each camera respect a known reference system.

When intrinsic and extrinsic parameters are known is possible to find the 3D coordinates of a point P knowing the 2D projection coordinates of p in the two image planes. Literature presents lots of methods to calibrate the stereo rig and some algorithms are already implemented in dedicate toolboxes and libraries. MATLAB [5] calibration toolbox, based on [6], achieves good results but required manually points detection on the images used for calibration process, so it is preferable to implement calibration process in `C++` with OpenCV library [4] as described in Chapter 5. The process of calibration consists to target the camera on known structure (a chessboard pattern) that has many individual and identifiable points. By viewing this pattern from a variety of angles is possible to estimate the (relative) location and orientation of the camera and the intrinsic parameters.

With calibration process it is also possible to obtain the *Essential Matrix* E and *Fundamental Matrix* F. The Essential matrix E contains information about translation and rotation from the two cameras in the 3D space, and Fundamental matrix F contains the same information of E with the intrinsic parameters of each camera. Matrix F and E can help to derive the relation that connect the observed point P with the two points p_l and p_r in the two images. Essential matrix can be written as:

$$E = R \cdot S \tag{2.9}$$

where:

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2.10}$$

and R is a rotational matrix.
The relation between points in left and right image must satisfy the following relation:

$$p_r^\top E p_l = 0 \tag{2.11}$$

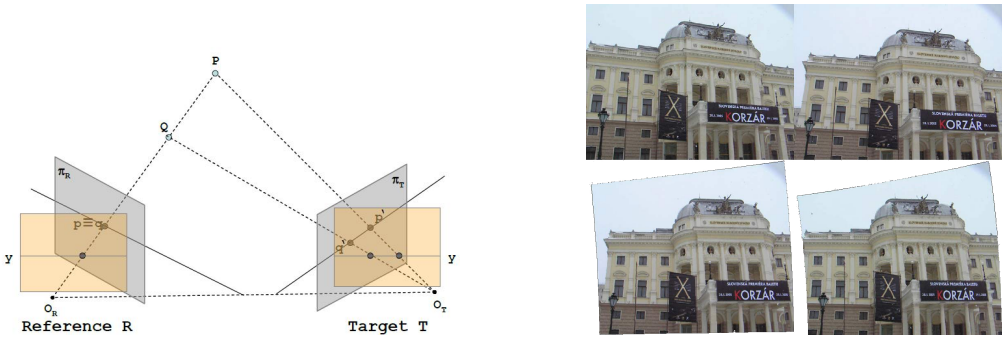
Fundamental matrix can be build from camera matrix M as:

$$F = (M_r)^{-1} E (M_l)^{-1} \tag{2.12}$$

2.5 Rectification

Since the search space for corresponding points can be narrowed from 2D to 1D (due to epipolar constraints), the images can be put (virtually) in a more convenient configuration, *standard form*, to have both images in the same plane. As described in [7], image rectification can be view as the process of transforming the epipolar geometry, of a pair of images, into a canonical form. As presented in [8] a simple way to rectify the two images is to first rotate both cameras so that they are looking perpendicular to the line joining the camera centers, c_0 and c_1. Next, to determine the desired twist around the optical axes, make the the y axis camera perpendicular to the camera

center line. This ensures that corresponding epipolar lines are horizontal and that the disparity for points at infinity is 0. Finally, re-scale the images, if necessary, to account for different focal lengths, magnifying the smaller image to avoid aliasing. One of the most importants and used rectification



(a) (b)

Figure 2.8: *Example of rectification process*

algorithm is *Bouguet's Algorithm* [5], that given the rotation matrix and translation vector (R, T) between stereo images, tries to minimize the amount of change reprojection produced for each image while maximizing common view area. The rotation matrix R, that rotate the right camera to overlap with the left image plane, is split in half between the two cameras, given two matrix r_l and r_r. Each camera rotates half rotation to have co planar alignment but not into row alignment. Starting with the direction of the epipole e_1 is possible to create the rectification matrix R_{rect} that will take the left camera's epipole to infinity and align the epipolar lines horizontally. Taking the principal point (c_x, c_y) as the left image's origin, the direction of the epipole is directly along the translation vector between the two camera center of projection.

$$e_1 = \frac{T}{||T||} \tag{2.13}$$

The second vector e_2 must be orthogonal to e_1 and it is possible to choose a direction orthogonal to the principal ray:

$$e_2 = \frac{[-T_y \quad T_x \quad 0]^T}{\sqrt{T_x^2 + T_y^2}} \tag{2.14}$$

Finally the third vector must be orthogonal to e_1 and e_2:

$$e_3 = e_1 \times e_2 \tag{2.15}$$

So R_{rect} can be construct in this way:

$$R_{rect} = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \tag{2.16}$$

The raw alignment of the two cameras is then achieved by setting:

$$R_l = R_{rect} r_l \tag{2.17}$$

$$R_r = R_{rect} r_r \tag{2.18}$$

In this way the two images lie in the same plane. All steps are resumed in Fig. 2.9[3].

The resulting standard rectified geometry is employed in a lot of stereo cameras setups and stereo algorithms and leads to a very simple inverse relationship between 3D depths Z and disparities d,

$$\frac{B}{Z} = \frac{(B + x_T) - x_R}{Z - f} \quad \Longrightarrow \quad Z = \frac{B \cdot f}{x_R - x_T} = \frac{B \cdot f}{d} \tag{2.19}$$

where $d = x_R - x_T$ is the *Disparity* that grows when the point P is closer to the cameras.

Since the camera is modelled as a pinhole camera, it is possible to calculate the aperture angles with the following equations:

$$FOV_x = 2 \cdot \arctan \frac{w}{2 \cdot f} \tag{2.20}$$

$$FOV_y = 2 \cdot \arctan \frac{h}{2 \cdot f} \tag{2.21}$$

[3] `Fig.Ref. [4]`

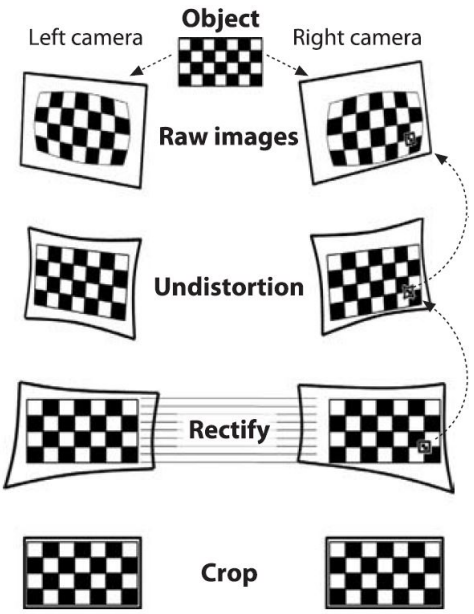


Figure 2.9: *Step doing in the rectification process*

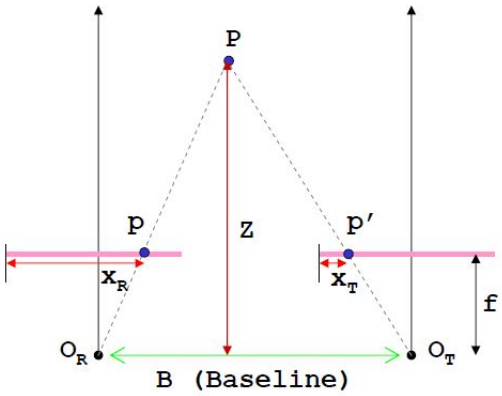


Figure 2.10: *Stero cameras after rectification process, B identifies the distance between the two optical center O_R and O_T, f is the focal length, Z is the depth and x_R and x_T are the x-coordinate projection of point p in the two image planes*

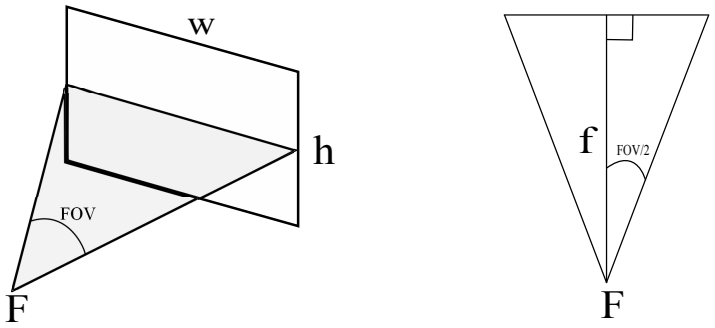


Figure 2.11: *Geometric representation of the aperture angle. Left diagram shows the spatial placement of image sensor, focal point F and horizontal aperture angle FOV. Right diagram shows the relation between image sensor width, focus length, and horizontal aperture angle*

where w and h represent the image sensor width and height respectively and f represent the camera focal length. Also it is possible to find the height H and width W of an object in the image as:

$$W = \frac{w \cdot Z}{f} \tag{2.22}$$

$$H = \frac{h \cdot Z}{f} \tag{2.23}$$

Given the smallest allowed disparity increment Δd, we can find smallest achievable depth range resolution ΔZ as:

$$\Delta Z = \frac{Z^2}{fB}\Delta d \tag{2.24}$$

and the smallest distance that can be recognized will be:

$$Z_{min} = \tan\left(90^\circ - \frac{FOV}{2}\right)\frac{B}{2} \tag{2.25}$$

where FOV is the Field of View of the cameras.

2.6 Stereo matching

The images captured from the two cameras are a set of points (pixels) and in order to find the distance from a point Q, in a 3D space and the re-projection points in the two image planes (q_l,q_r), is necessary to calculate

the disparity from the same points. For this reason, in stereo vision, is fundamental to recognize the same parts/points of the same scene from two different pictures. This is the problem of stereo matching. In literature there are lots of techniques of stereo matching, and it's possible to divide the stereo matching algorithms in two types:

- **Features based**
- **Dense**

The first kinds of algorithms allow to obtain disparity information for a finite number of *Features* (that are interest sets of points). This algorithms are so fast, because the number of features is smaller compared with the pixels in the image. Image features can be extracted in different ways, in [9] Lowe presents *Scale Invariant Features (SIFT)*, that is an algorithm that detects and describes local features in images. This method is based on a particular class of features that are scale, rotational and illumination invariant. These features are easily to find with a filter that identified stables points. These algorithms have good results but are so slow. *SURF (Speeded-up Robust Features algorithm)* [10] tries to resolve the SIFT speed and robustness problem. This method uses an approximation to the determinant of Hessian matrix that is approximate from integral image that is an intermediate representation of an image.
Unfortunately these algorithms are little used due to the poor disparity generated, that is limited only at the points that have distinctive features.

Dense algorithms try to generate disparity map for all the image points in agreement to a particular taxonomy [11]. It's possible to divide these algorithms in two different categories: *local* and *global*. The local algorithms seek similar points independently of one another, in a neighbourhood, of each examined point. Differently the global dense algorithms take assumption about same characteristics of the nature of the objects, for example smoothness, and seek a disparity map that minimize a global cost function. So the difference from these algorithms is the method used to minimize this

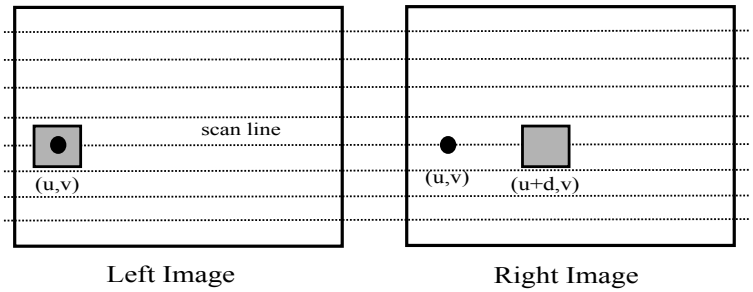


Figure 2.12: *Stereo correspondence starts by assigning points matches between corresponding rows in the left and right image. SAD windows moving along scan line. Match search starts at minimum disparity point and move to the left for the set number of disparity.*

function (e.g simulated annealing, probabilistic diffusion, graph-cut and so on). Algorithms based on graph-cut allow to obtain good results but with an high computational cost.

In this application a fast (real time) stereo matching is required, and for this reason is convenient to use the *Sum of absolutely differences* (SAD) cost aggregation.
SAD works by taking the absolute difference between each pixel in the original block and the corresponding pixel in the target block being used for comparison. These differences are summed to create a simple metric block similarity. The most similar block is lighter than the others blocks.

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)| \tag{2.26}$$

The difference is calculated for all points (i, j) in a set W. The size of W can be changed arbitrary to have more precision and reduce the search area. There are also some variants of this algorithm that change how to weight the difference of the pixels (e.g. Sum of Square Difference (SSD), Zero-mean Sum of Absolute Differences (ZSAD), Sum of Hamming Distances (SHD), and so on). There are three steps in the block matching technique that OpenCV uses: pre-filtering, correspondence search, and post-filtering. In the first stage, left and right images are normalized to have the same lighting levels.

A window of variable size is placed over each pixel and the central pixel is replaced as:

$$\min[\max(I_c - \overline{I}, -\hat{I}), \hat{I}] \tag{2.27}$$

where $\overline{I}$ is the average intensity value in the window, $\hat{I}$ is an upper limit (pre-set) and I_c is the intensity of the pixel in the center of the window, that will be changed.

After rectification, each row is an epipolar line, so the matching location in the right image must be along the same row (same x-coordinate) as in the left image (Fig.2.12); this matching location can be found if the feature has enough textures to be detectable and if it is not occluded in the right camera view. Post filtering removes false matches correspondences. Once the correspondences, for all common points, are known it is possible to calculate the disparity value for all pairs of pixels and to build a *disparity map* (usually encoded with grey scale image) that represents the disparity of all image and that is proportional of the 3D depth (Fig.(2.13)). Brighter intensity values

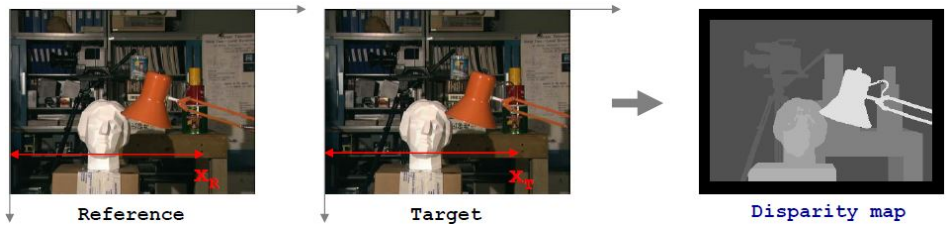


Figure 2.13: *Example of disparity map*

represents objects that are closer to the camera while darker objects are those farther away from the camera. Black pixels are those points where no correspondence was found between the images.

As described in [12] there is an alternative implementation of block matching that performs an optimization across the entire image, and not only a locally finite neighborhood. For this reason this technique is called *Semi-Global Matching*.

Semi-Global Matching performs a better result respect simple Block Matching but present an high computationally cost and in order to have good

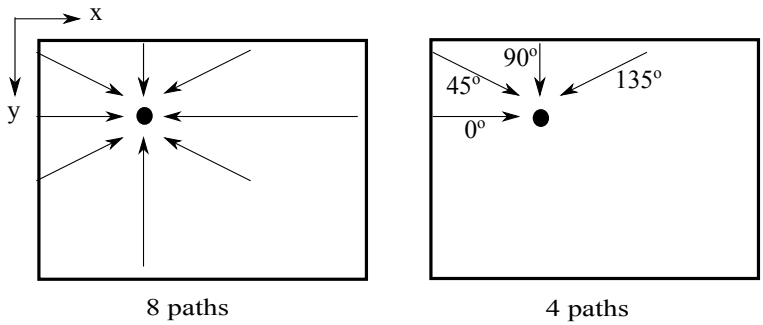


Figure 2.14: *Possible path orientation for 8 (a) or 4 (b) directions*

performances in real time system, dedicated hardware implementations is required. This algorithm aims to minimize the following global energy function, E, for disparity image, D:

$$E(D) = \sum_{p} \left[C(p, D_p) + \sum_{q \in N_p} P_1 \cdot I\left[|D_p - D_q| = 1\right] + \sum_{q \in N_p} P_2 \cdot I\left[|D_p - D_q| > 1\right] \right] \tag{2.28}$$

where $E(D)$ is the energy for disparity image, p, q represent indices for pixels in the image, N_p is the neighborhood of the pixels p, $C(p.D_p)$ is the cost of pixels matching with disparity in D_p, P_1 is the penalty passed by the user for a change in disparity values of 1 between neighboring pixels, P_2 is the penalty passed by the user for a change in disparity values greater than 1 between neighboring pixels and $I[\cdot]$ is the function which returns 1 if the argument is true and 0 otherwise. P_1 and P_2 governed the smoothness of disparity map (with $P_2 \geq P_1$). The semi-global matching function approximates the 2-D minimization by performing multiple 1-D minimization. The matching function aggregates costs on multiple paths which converge on the pixel under examination. Cost is computed for the disparity range specified by the minimum disparity and number of disparities. We can calculate costs in four or eight directions (Fig.2.14).

Semi-Global Block Matching introduces global consistency constraints by aggregating matching costs along several independents, one dimensional paths, across the image. As describe in [13] the minimization of (2.28) can be done by moving along one dimensional path L in r directions. The cost along a path is done by recursively computing:

$$\begin{aligned} L_r(P,d) =& C(p,d) + \min[L_r(p-r,d), \\ & L_r(p-r,d-1) + P_1, \\ & L_r(p-r,d+1) + P_1, \\ & \min_i L_r(p-r,i) + P_2] - \\ & \min_i L_r(p-r,l) \end{aligned} \tag{2.29}$$

where the first term describe the primary match cost (simple block matching). The second term adds the minimal path costs of the previous pixel $p-r$ with a penalty P_1 for disparity changes ($|\Delta d| = 1$) and P_2 for disparity discontinuities ($|\Delta d > 1|$) respectively. Quasi global optimization across the entire image is achieved by calculating path cost in all considered direction:

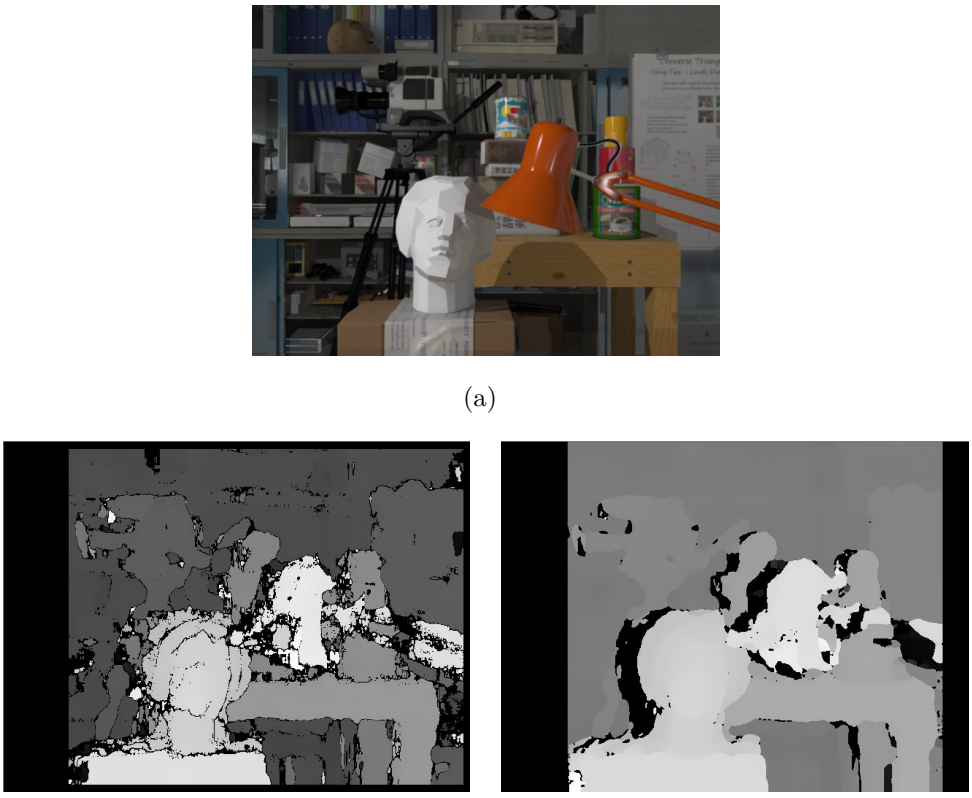
$$S(p,d) = \sum_r L_r(p,d) \tag{2.30}$$

Finally the disparity map is calculated by selecting the disparity with the minimal aggregation cost $\min_d S(p,d)$ for each pixel.

Figure 2.15[4] shows the differences between the algorithms presented. Figure 2.15(b) shows the application of simple Block-Matching algorithm with SAD windows size of 21 pixels, number of disparity of 64 pixels, texture threshold of 0 and a pre-filter size of 5 pixels. Figure 2.15(c) shows disparity map calculated with $SGBM$ algorithm. Disparity map is denser respect BM but can be difficult to extract single objects from the map. In this case SAD windows size is 21 pixels, 4 paths direction and with the penalty weights: $P1 = 1000$ and $P2 = 2400$. The execution time for BM is $\simeq 100ms$ and for

[4]Dataset from http://cvlab-home.blogspot.it/2012/05/h2fecha-2581457116665894170-displaynone.html

$SGBM$ is $\simeq 600ms$. For this reason in real time application is preferable to use simple Block Matching.



(a) (b) (c)

Figure 2.15: *Application of Block Matching and Semi-Block Matching at the same simulated scenario. (a) is the original image, (b) is BM algorithm and (c) is SGBM algorithm*

CHAPTER 3

Techniques for objects extraction

This chapter presents some techniques that try to extract image information in order to find obstacles in the scene. Image object detection is a very important and difficult process. Literature presents a lot of methods, but that works only in particulars situations, or if some information about the obstacles are known (such as color, position, dimension, shape and so on). In our scenario obstacles can be various, with different colors, shapes, and dimensions and we don't have any information about it. For this reason three techniques, *Segmentation*, *Edge extraction* and *Saliency*,have been studied and implemented, because they work in many conditions and can be used to improve the simple stereo vision.

3.1 Segmentation

As presented in [14] segmentation can be defined as a "technique of labelling features". For each feature is possible to associate a label that contain some informations, and it is possible to assume that two features is similar if they have the same label. Segmentation could be very important in image processing because can help to reject stereo correspondence errors. This can be possible because in segmentation technique "similar is defined as smoothly". A group of features of the same object (or obstacle) are smoothly varying, but from two different objects the variation can be very high. Segmentation

divides the image in *k-regions* such that within group features are smoothly varying and between different group features are discontinuous. Smoothly varying features will generate poor stereo correspondence due to low contrast, so segmentation ignore disparity interior to the region. The problem of segmentation and grouping remain a great challenge due to the difficult task to merge high computational costs and fast performances. For our purpose the segmentation method must have the following proprieties:

- *Capturing perceptually important groupings or regions, which often reflect global aspects of the image.* It's important that the regions that we define with segmentation reflects the original information of the image. Segmentation should simplify the image without loss of information.
- *Be highly efficient, running in time nearly linear in the number of image pixels.* In our application we have a stream video, composed by sequentially frames, so the computational time must be smaller as possible (linear with the dimension of the images).

3.1.1 Intensity Segmentation

One of the simplest and fastest segmentation algorithm that can be used in our application is segmentation by intensity.

Segmentation by intensity aims to separate the background from the objects, that can be found within the image using the assumption that background is brighter than the objects. This segmentation can be used when we are at high quote and we have homogeneous and brighter background (only sky) and a darker objects (for example another aircraft or drone) that we want to detect. For this task, threshold is an appealing method because it is computationally inexpensive and fast. The success of background subtraction by thresholding is however solely depending on our knowledge of the background intensity properties. We can simply choose a global threshold and if the input pixel intensity is above the chosen threshold level, the corresponding output pixel is set to one and if the intensity is below the threshold

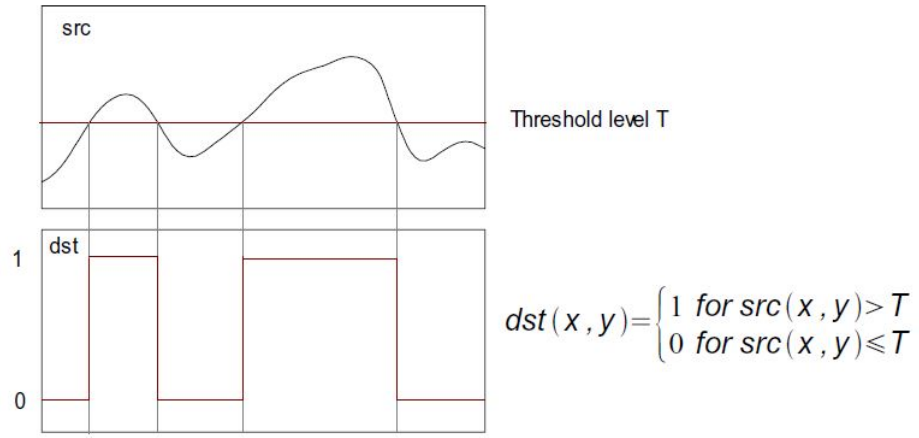


Figure 3.1: *Intensity segmentation by threshold*

level is set to zero. The threshold used to select the interest level of darker can be found with the histogram of the image as we can see in figure 3.2. Histogram is a graphical representation of the distribution of data. It plots the number of pixels for each tonal value and help to judge the entire tonal distribution at a glance.

As figure 3.2 shows, histogram has 3 peaks; the highest describe the sky and brighter color, with high color level (250-255), the second peak describe the drone and the little third peak is the helicopter in the background. With this method is possible to extract objects from background but only if the scene is poor of particulars and there is a lot of differences from the objects and the background. An example of object extraction with this technique is described in Fig.3.3.

If we have a complexity scene the histogram will not have isolated peaks, and will be so hard extract the objects information. Sometimes adaptive threshold can be used to have better result, but it's necessary to find a better algorithm that works in all conditions of image complexity.

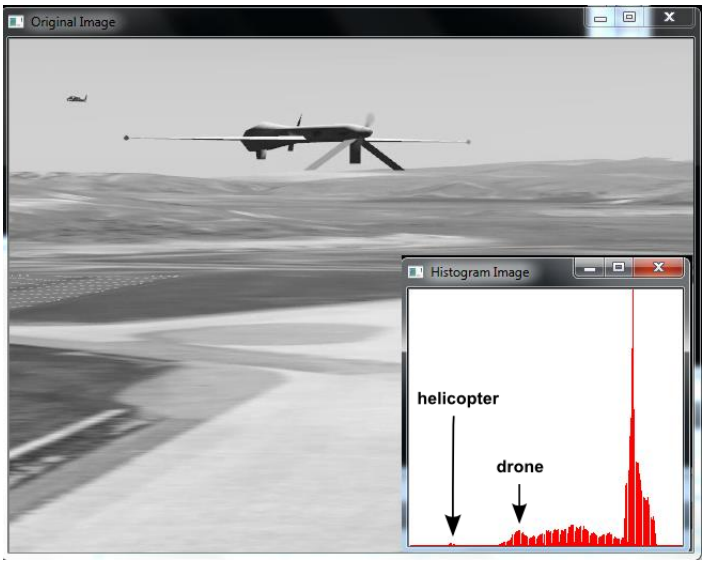


Figure 3.2: *Simulated scenario with grey-level histogram. We can see that we have a peak in correspondence of drone and the helicopter*

Figure 3.3: *Extraction of helicopter from background by intensity segmentation*

3.1.2 Colour Segmentation

Colour Segmentation is the simplest and the most utilized technique used to extract objects from an image.
If the obstacle colour is known can be very simple to remove the other colours and isolate the object. This technique doesn't work if the goal is to find all kinds of objects/obstacles in the scene, but if we want to extract a particular object, such as aircraft, roof, wall and so on, can be used because its characteristics are known.
The captured image is filtered with a Gaussian filter in order to remove the noise and converted in HSV image. Hue-Saturation-Value is the most common cylindrical coordinate representations of points, in an RGB color model. The purpose of this model is to aid selection, comparison, and modification of colours by organizing them into a cylindrical geometry which roughly corresponds to human perception. When this range is note a mask is applied at the image to remove all the other colours and it is all converted in a binary image. After that morphological filters (see section [3.3]) are applied to fill some empty regions coming from some colour intensity variations.
Due to its simplicity and robustness is also possible applied a Kalman filter to track the objects found, in case of occlusions or temporary loss of image informations.
An example of this technique is shown in Fig. 3.4.

In literature can be found a lot of segmentation algorithms, for example Watershed algorithm [15] where a grey-level image may be seen as a topographic relief, where the grey level of a pixel is interpreted as its altitude in the relief, or eigenvector based methods [16] and so on, that achieved excellent results but with high computational cost that doesn't makes them practical for many applications.

(a) (b)

Figure 3.4: *Example of colour segmentation. Figure (b) shows the binary image obtained by colour extraction and opening operator. Figure (a) shows the result on original image with a green rectangle on the object found.*

3.1.3 Graph-based Image Segmentation

In graph based image segmentation methods, the image is modelled as a weighted undirected graph $G = (V, E)$, where each node in V is associated with a pixel, or a group of pixels of the image, and edges in E connect certain pairs of neighbouring pixels. The weight $w(u, v)$ associated with the edge $(u, v) \in E$ describes the affinity (or the dissimilarity) between the two vertices u and v. The segmentation problem is then solved by partitioning the corresponding graph G, using efficient tools from graph theory, such that each partition is considered as an object segment in the image. Among graph based algorithms, Efficient-Based Image Segmentation by Felzenswalb and Huttenlocher [17] is worth to be mentioned due to its effectiveness and low computational complexity.

The Efficient Graph-Based algorithm is based on minimum spanning tree (MST) and it can preserve detail in low-variability image regions, while ignoring detail in high-variability regions adjusting the segmentation criteria. Starting considering each pixel as a component, it merges similar components using a predicate, which is based on measuring the dissimilarity between elements along the boundary of the two components. For this purpose some parameters are defined.

The *Internal difference* of a component $C \subseteq V$ is defined as the largest weight in the minimum spanning tree of the component, $MST(C, E)$. That is:

$$Int(C) = \max_{e \in MST(C,E)} w(e) \tag{3.1}$$

The difference between two components $C_1, C_2 \subseteq V$ is defined as the minimum weight edge connecting the two components. That is,

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)) \tag{3.2}$$

The region comparison predicate evaluates if there is evidence for a boundary between a pair of components by checking if the difference between the components, $Dif(C_1; C_2)$, is large relative to the internal difference within at least one of the components, $Int(C_1)$ and $Int(C_2)$. The pairwise comparison predicate is then defined as:

$$D(C_1, C_2) = \begin{cases} true & \text{if} \quad Dif(C_1, C_2) > MInt(C_1, C_2) \\ false & \text{otherwise} \end{cases} \tag{3.3}$$

where $MInt(C_1, C_2)$ is the *Minimal Internal Difference* defined as:

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{3.4}$$

We can modify $MInt(C_1, C_2)$ with τ that represents a threshold function based on the size of the component:

$$\tau(C) = \frac{k}{|C|} \tag{3.5}$$

where $|C|$ denotes the size of component C and k is a tuning parameter. Using this threshold function makes harder to create small components because for small components a stronger evidence for a boundary is required.

Colour images are split as three separate monochrome images (R,G,B planes). Each pixel corresponds to a node $v_i \in V$ and the edge set E is constructed by connecting pairs of pixels that are neighbours in an 8-connected senses (any other local neighbourhood could be used) as described in figure 3.5.

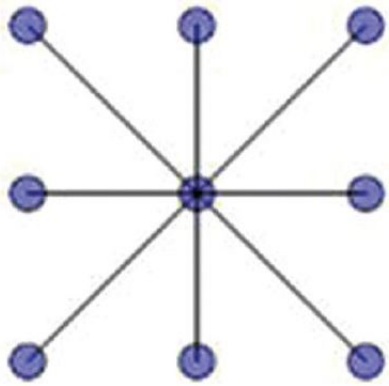


Figure 3.5: *Connection of each pixel with his neighborhood*

This yields a graph with $m = O(n)$ number of edges, and the running time of the segmentation algorithm is $O(n \log n)$ for n image pixels. We use an edge weight function based on the absolute intensity difference between the pixels connected by an edge:

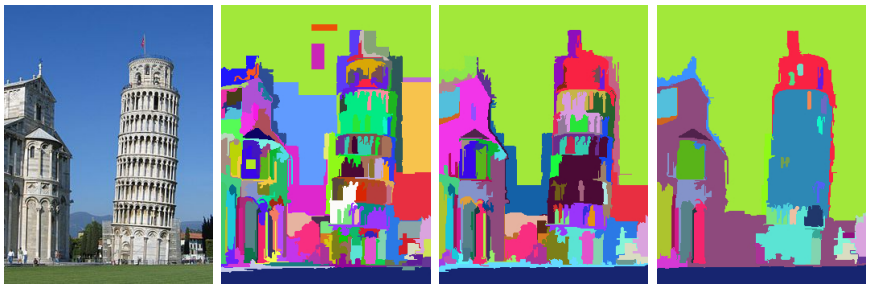
$$w((v_i, v_j)) = |I(p_i) - I(p_j)| \tag{3.6}$$

where $I(p_i)$ is the intensity of pixel p_i. In general is used a Gaussian filter to smooth the image slightly before computing the edge weights, in order to compensate for digitization artefacts, with $\sigma = 0.8$, which does not produce any visible changes to the image but helps remove artefacts. For colour images algorithm run three times, once for each of the red, green and blue colour planes, and then intersects these three sets of components. The algorithm for image segmentation is resumed in table 1.

The algorithm sorts the edges set in non decreasing edge weight order and then starts with the real segmentation process.
Let v_i, v_j the vertices connected by q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$ and living in disjoint components of S^{q-1}, if $w(o_q)$ is smaller than the internal difference of both those components, it merges the two components otherwise do nothing. This comparison is repeated for all pair of connected vertices in disjoint components. The result of this segmentation algorithm is described in Fig. 3.6

Sort E into $\pi = [o_1, ..., o_m]$, by non decreasing edge weight
Start with segmentation S^0, where each vertex v_i is in its own component.
for *(q=1* ***to*** *m)* **do**
  Let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j
  if $C_i^{q-1} \neq C_j^{q-1}$ *and* $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ **then**
    S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}
  else
    $S^q = S^{q-1}$
  end
end

Algorithm 1: Segmentation Algorithm



(a) (b) (c) (d)

Figure 3.6: *Example of image segmentation with* $k = 100$ *[b],* $k = 300$ *[c] and* $k = 700$ *[d]*

As show in Fig. 3.6 increasing the tuning parameters k is possible to change and increase the area of each component, because small components are less considered, and more image informations are discarded.

3.2 Canny Edge Detector

The purpose of edge detection, in general, is to significantly reduce the amount of data in an image, while preserving the structural properties to

be used for further image processing. The aim of algorithm proposed by John F. Canny [18] is to find the optimal edges of image, where ”optimal” means satisfied the following criteria:

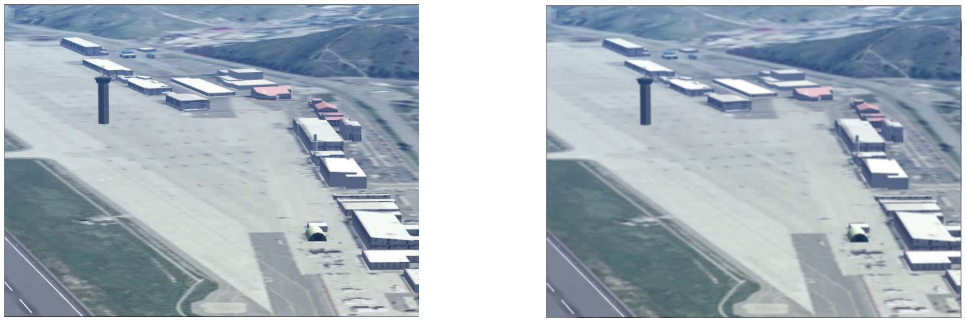
- **Detection**: the probability of detecting real edge points should be maximized while the probability of falsely detection non-edge points should be minimized. This correspond to maximizing the signal to noise ratio.
- **Localization**: the detected edges must be close as possible to the real edges.
- **Minimal Response**: one real edge should not result in more than one detected edge.

The algorithm is composed by 4 steps:

- Smoothing
- Finding Gradients
- Non-Maximum Suppression
- Double Thresholding
- Edge Tracking

The first step is called *Image Smoothing* where a Gaussian filter is applied on the image to reject the inevitable noise that the image present. The filter choosing is very important, because it has a strong influence on the final result. A filter with small dimension produces less blur and allows to recognize small details, whereas bigger filter produces high blur and allows to detect larger area. As we can see in Fig. 3.7 a 3×3 the Gaussian filter applied at the image doesn't produces any visible changes.

In the second step algorithm calculates the *Intensity Gradient* $G(x, y)$, and angle $\alpha(x, y)$ of the gradient vector. Intensity gradient is calculated



(a) (b)

Figure 3.7: *Example of Gaussian smoothing filter. Original image on the Left and smoothed image on the right*

applying *Sobel-Operator* and gradient magnitudes (also known as the edge strengths) can be determined as an Euclidean distance by applying the law of Pythagoras as shown in equation 3.7

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.7}$$

where G_x and G_y are the gradients in x and y respectively.

The angle direction of the edge is determinate as:

$$\alpha(x, y) = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.8}$$

obviously angle must be approximate at the possible pixel location (45^o,90^o...). The third step consists in the *Non-maximum Suppression*. For an edge pixel, the gradient value is bigger than the gradient values of its neighbours in the same direction. So for each points is necessary to detect the direction of the gradient and compare his module with the module of its neighbours. If at least one of the two pixels in the neighbour has bigger module respect the interest pixel, this pixel is rejected imposing his module to zero. In a 3-by-3 pixels region it is possible to define four directions for an edge passing troughs the central pixel (the horizontal,vertical and diagonal direction).

So the direction of edge is obtained by the direction of the gradient vector, which in turn, is obtained by angular value $\alpha(x, y)$.

In the fourth step, *edge thresholding*, the algorithm applies two thresholds

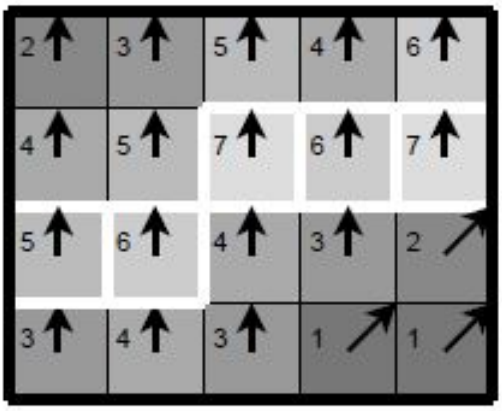


Figure 3.8: *Illustration of non-maximum suppression. The edges strengths (module) are indicated both as colours and numbers, while the gradient directions are shown as arrows. The resulting edge pixels are marked with white borders.*

(high T_H and low T_L) at the precedent image. Each pixel intensity gradient is compared with T_H and T_L and if $G(x, y)$ is:

- **Less** than T_L the pixel is discard
- **Greater** than T_H the pixel is part of edge
- **Between** T_L and T_H the pixel is accepted as part of edge only if it is contiguous of another edge point.

The result of Canny edge detector is show in Fig. 3.9.

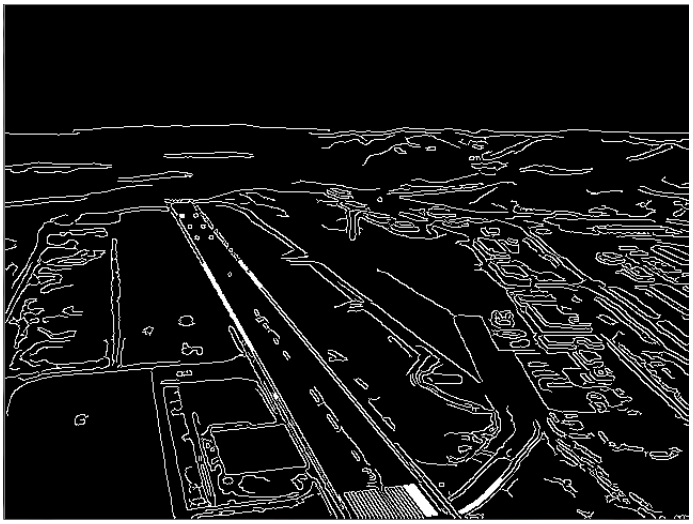


Figure 3.9: *Canny edge detector applying on 3D simulated environment*

3.3 Morphological Operations

Morphological Operations are a set of mathematical operations in image processing, that can help to extract or modify some components in the image. There are two fundamentals morphological operations: *Erosion* and *Dilatation*. The first can be used to remove or thin the objects on scenario, and the second to increase or fill the objects. A binary image is considered as a subset A of integer numbers. In this way objects into image are described by pixels with value 1 and the background is characterized with pixels set to 0. Objects are manipulated by a little set B called *Structural element* (SE) with a point that characterized its origin. Usually SE is symmetrical with the origin in the center. For not binary image, such as gray scale images, a function $f(x, y)$ as image and $b(s, t)$ as SE are used. Either function given an intensity value at every pair of internal coordinate (x,y). An example of structural element is display in Fig. 3.10.

1	1	1
1	1	1
1	1	1

1	0	1
0	1	0
1	0	1

1
1
1

Figure 3.10: *Different type of Structural Elements*

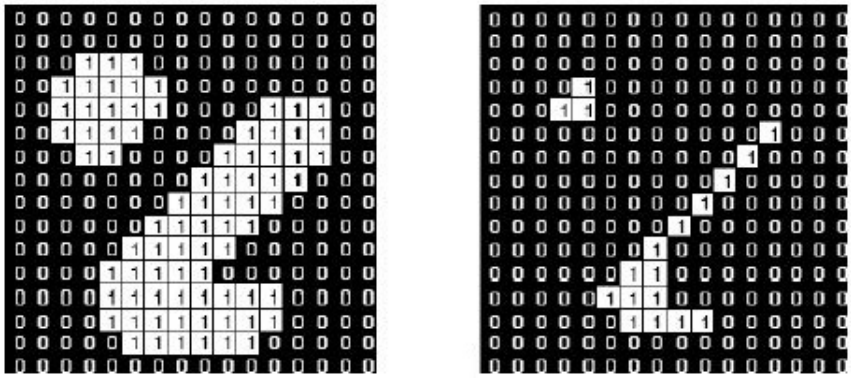


Figure 3.11: *Example of Erosion operation with a* 3×3 *symmetrical Structural Element*

3.3.1 Erosion

The *Erosion* operation of binary image A is defined by a translation of structural element B:

$$A \ominus B = \{z|(B_z) \subseteq A\} \tag{3.9}$$

in other words, $A \ominus B$ is the set of points z of image A such that B translated in z is completely contained in A. With this operation we can reduce the elements in the image. In this case Erosion operation is used on disparity map to separate objects that can be wrongly attached due to a big SAD windows size used. It is also possible to find contours with erosion operation, applying this operation at the image and later subtracting the original image with the image found. Naturally it is important to find an opportune SE. Generally a 3×3 symmetric SE is used. For a grey-scale image the erosion operation on $f(x, y)$ is applied in all points (x, y), in order to find the minimum value

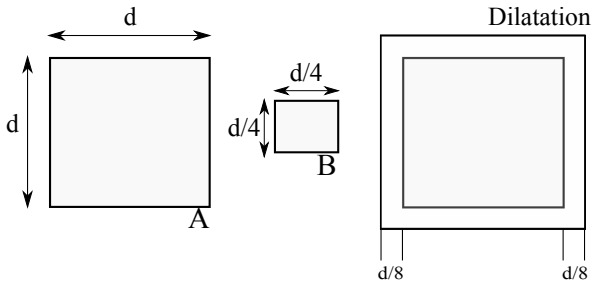


Figure 3.12: *Dilatation Operation*

in the region coinciding with $b(s,t)$ when the origin is set in (x,y).

$$[f \ominus b](x,y) = \min\{f(x+s, y+t)\} \qquad \text{with} \quad (s,t) \in b \tag{3.10}$$

3.3.2 Dilatation

Dilatation of a binary image A with a structural element B is defined as:

$$A \oplus B = \{z | (B')_z \cap A \neq \emptyset\} \tag{3.11}$$

The structural element it's reflected respect his origin and moved of z position by a translation. Figure 3.12 shows an example of dilatation of set A with a symmetrical SE ($B' = B$). The result is the set of all points such that B' and A overlap in at least one point. In this case dilatation is used to try to close boundaries in Canny edges detection and to remove holes in the disparity map that represents noise resulting from equal textures. In all cases a 3 symmetrical SE is used in order to expand the contours of object of one pixel in all directions.
For a gray-scale image the Dilatation operation with a reflected SE respect his origin $(-s,-t)$ is given by:

$$[f \oplus b](x,y) = \max f(x-s, y-t) \qquad \text{with} \quad (s,t) \in b \tag{3.12}$$

so dilatation finds the maximum value of image in the region coinciding with $b(s,t)$ when the origin is set in (x,y).

3.3.3 Complex Morphological Operation

The operation of dilatation and erosion for a same structural element allows to define a complex operator. If dilatation operation follows an eroding operation, with the same SE on the image A, the result operation is called *Opening*.
With opening operator it is possible to preserve regions with a similar shape of SE, makes more homogeneous objects contours and remove little interruptions.

$$A \circ B = (A \ominus B) \oplus B \tag{3.13}$$

On the other hand if erosion operator follows dilatation operator, with the same SE on the same figure A, the *Closing* operator is obtained.

$$A \bullet B = (A \oplus B) \ominus B \tag{3.14}$$

This kind of operation makes section of contour more homogeneous filling the empty areas. The Opening and Closing operator can be used as filters to remove noise as shown in Fig 3.13.

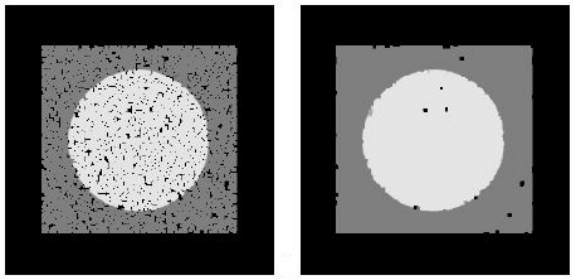


Figure 3.13: *Example of Opening operator as noise filter*

3.4 Saliency Map

Saliency Map contain information about where the interesting information can be found. These areas corresponds to features considered as rare or informative, depending on the definition of salience used. As analyzed in [19] there are many definitions of saliency that can be used, based on:

- Key-points criteria
- Absolute value of the local wavelet decomposition of the image
- Salient features having a low probability of being mis-classified with other features
- Select image fragments which maximize the natural information between the fragment and the object class.

High saliency regions correspond to objects or places that are most likely to be found, while lower saliency is associated to the background. In this way saliency can help to better define contours of objects to extract with segmentation process as described in section 3.1.
In [20] Frintrop introduce the *Visual Object detection with a CompUtational attention System* (VOCUS) that is a new fast methodology to find saliency map.
VOCUS takes the original colour image and converts it in gray-scale image. Then, a Gaussian image pyramid is created, applying a 3×3 Gaussian Filter to the gray-scale image and scaling it down by a factor of two on each axis. The filtering and scaling are repeated four times, yielding five images i_0, i_1, i_2, i_3, and i_4. Algorithm works now with only the information present in the smallest scales images (i_2, i_3, i_4). The system calculates now the "on-center" and "off-center" differences in the three images. Center-surround differences are used in order to calculate every features map. A pixel is candidate as center and two surround values, σ, are used. Therefore is obtained 12 intensity sub maps. The intensity value of center and surround is obtained as:

$$center(x, y, s) = i_s(x, y) \tag{3.15}$$

$$surround(x, y, s, \sigma) = \frac{\sum_{x'=-\sigma}^{x'=\sigma} \sum_{y'=-\sigma}^{y'=\sigma} i_s(x + x', y + y') - i_s(x, y)}{(2\sigma + 1)^2 - 1} \tag{3.16}$$

So every pixel of each intensity sub map is obtained as:

$$Int_{On_{s,\sigma}}(x,y) = \max(center(x,y,s) - surround(x,y,s,\sigma), 0) \quad (3.17)$$

$$Int_{Off_{s,\sigma}}(x,y) = \max(surround(x,y,s,\sigma) - center(x,y,s,), 0) \quad (3.18)$$

where $s \in \{2,3,4\}$ is the scale image considered and $\sigma \in \{3,7\}$. After that, on-center intensity map is calculated. This is possible scaling the six center intensity sub maps into largest scale i_2 and than summing pixel by pixel. The same process is applied for off-center intensity map.
It is possible to modify the previous algorithm introducing the concept of "Integral of image", that is an algorithm for quickly and efficiently generating the sum of values in a rectangular subset. So it is possible to replace equation (3.16) with integral. At the end the two intensity maps are mixed to obtain saliency map.

(a) (b)

Figure 3.14: *Example of VOCUS algorithm applied on simulated scenario. (a) original image, (b) saliency map*

CHAPTER 4

SteViE and SteViS Algorithms

This chapter presents the new improved stereo vision systems, implemented for object detection. After a brief description of a canonical stereo vision system, will be presented the two new algorithms implemented:

- **Ste.Vi.E** (STEreo VIsion Edges) where stereo vision is merged with edges extraction
- **Ste.Vi.S** (STEreo VIsion Saliency) where stereo vision is merged with saliency map and blobs tracking.

4.1 Canonical Stereo Vision system

Canonical stereo vision system is based on the theory described in Chapter 2. The first step is calibration process, that can be made with MATLAB or `C++` OpenCV library. In this work, calibration process is made with OpenCv library that performs faster and better results. When stereo parameters (such as intrinsic and extrinsic parameters) are known, the rectification process forces the two images to stay on the same plane. After that, there is the stereo matching step, with Block Matching technique and SAD cost aggregation, to obtain a fast process. Setting a threshold on disparity map pixel intensity, is possible to detect objects closer to the correspondent depth value. So

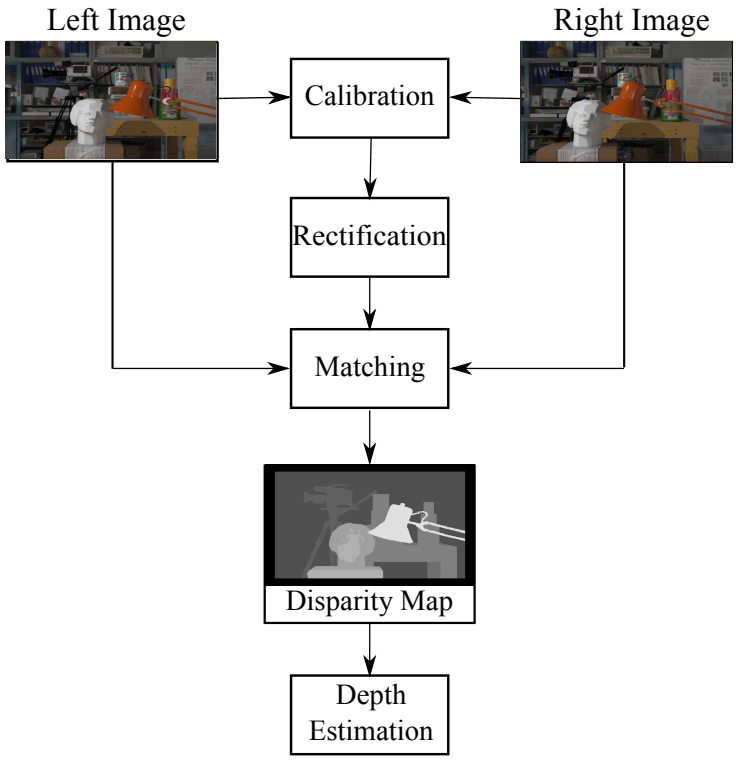


Figure 4.1: *Stereo Vision Algorithm*

there isn't a real objects detector. All the stereo vision process steps are summarized in Fig. 4.1.

4.2 SteViE Algorithm

Stereo Vision Edges is an improvement of the simple stereo vision system. This algorithm merges together stereo vision with edges extraction. The edges extraction process is obtained applying Canny edges detector (explained in Sec. 3.2) at the left image. As shown in Fig. 4.2, a pair of images are filtered with a Gaussian filter and the Canny edges extractor is applied at left image. When the contours of the objects in the image are defined, they will be filled and modified with morphological operators to

separate each pixels blob. All pixels blobs[1] correspond to an object that is highlighted by a red square and associated with depth value estimated by the disparity map.

4.3 SteViS Algorithm

Stereo Vision Saliency is an improvement of stereo vision system. This algorithm merges the SteVIE algorithm with Saliency map described in section 3.4. Saliency map extracts the most informative features and build a gray-scale image where the most informative features are brighter respect the other. So saliency map is an high contrast map and it is simpler for segmentation process to extract the different region. With saliency map is possible to extract the most important feature that with high probability are the objects features. The following steps are the same of SteVIE algorithm. At the end there is a the tracking blobs step, in which, each blob is tracked with a Kalman filter to remove occlusion problems.
All the SteVIS steps are show in Fig. 4.3.

[1]For Blobs extraction was used `CvBlobLibs` library. For more information visit `http://opencv.willowgarage.com/wiki/cvBlobsLib`

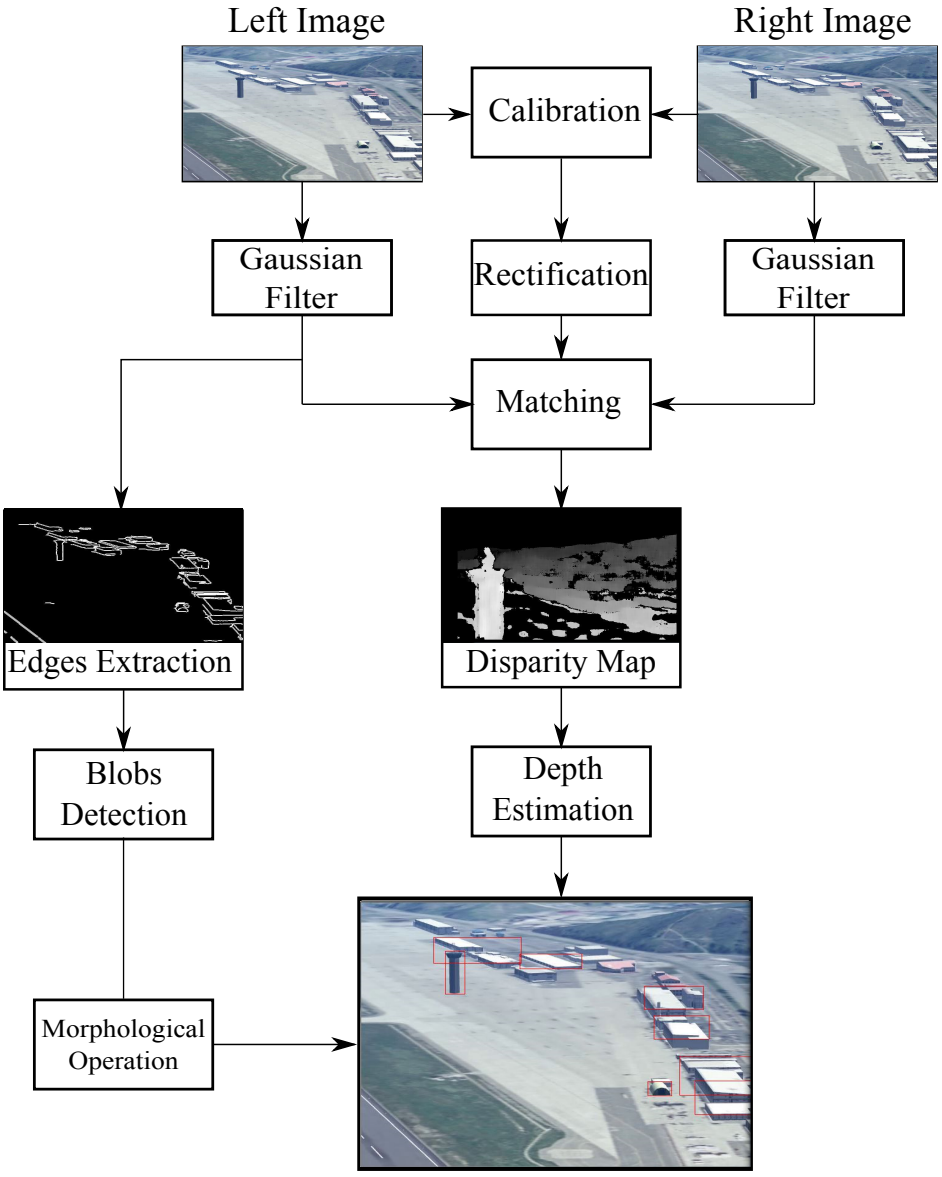


Figure 4.2: *Stereo Vision Algorithm with edges extraction (STEVIE)*

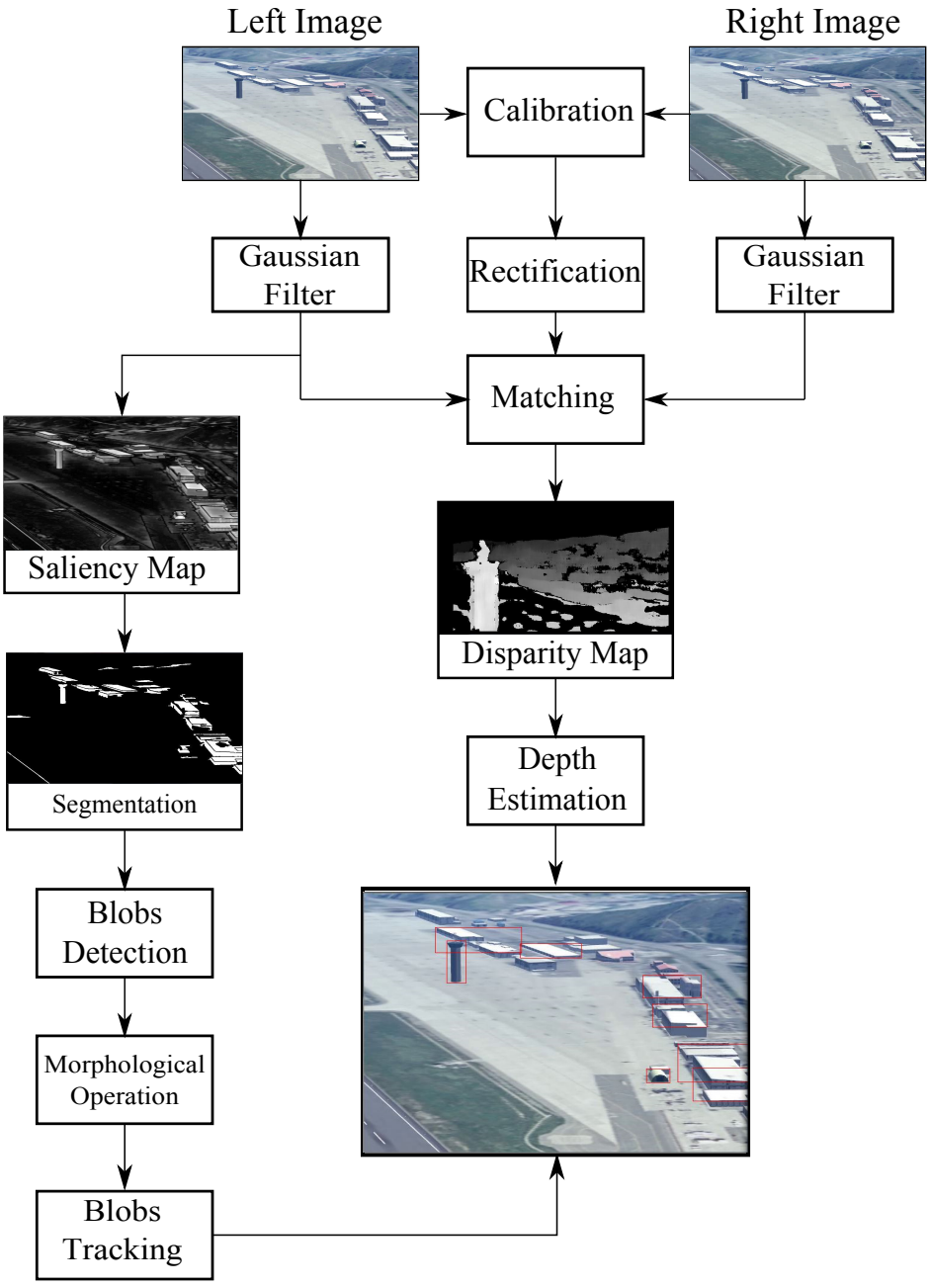


Figure 4.3: *Stereo Vision Algorithm with saliency map STEVIS*

4.4 Objects Tracking

Sometimes it is possible to lose the information about the objects found with the objects detection algorithms described previously. In this case can be useful to track or estimate the motion of the obstacles in case of partial or total occlusion.
For this reason has been developed a Kalman filter, that can be helpful in this case, with the *Hungarian Algorithm* that can be useful to recognize multiple objects in the scene.

4.4.1 Kalman filter with occlusion

Kalman filter can be used to cope with problems of estimating the state in linear and gaussian conditions.
Our process can be described with the following equations:

$$\begin{cases} s_k = As_{k-1} + w_{k-1} \\ z_k = Hs_k + v_k \end{cases} \tag{4.1}$$

where s_k and z_k represent the state and the measurements at time k, A is the system evolution matrix and H is the measurements matrix.
The random variables w_k and v_k represent the process and measurements noise that can be assumed with normal distributions:

$$E[w] \sim \mathcal{N}(0, Q) \tag{4.2}$$

$$E[v] \sim \mathcal{N}(0, R) \tag{4.3}$$

and independents:

$$E[w_i v_k^\top] = 0 \qquad \forall k, i \tag{4.4}$$

where Q is the process noise covariance matrix and R is the measurements noise covariance matrix.

Kalman filter allows a posteriori states estimation with the information about the state at previous step and the measures at actual step. The state

update is given by:

$$\hat{s}_k = \hat{s}_k^- + K_k(z_k - H\hat{s}_k^-) \tag{4.5}$$

where K_k is the kalman filter gain:

$$K_k = P_k^- H^\top (H P_k^- H^\top + R)^{-1} \tag{4.6}$$

$$P_k^- = A P_{k-1} A^\top + Q \tag{4.7}$$

$$P_k = P_k^- - K_k H P_k^- \tag{4.8}$$

The equations of Kalman filter fall into two groups: time update equations and measurements update equations. The time update equations are responsible for projecting forward the current state and the measurements update equations are responsible for the feedback step. The time update equation can also be thought as predictor equations, while the measurements update equations can be thought as corrector equations.
For simplicity we assumed that the objects found have linear trajectory and constant velocity, so the vector state will be:

$$s_k = [x_k, y_k, V_{x_k}, V_{y_k}] \tag{4.9}$$

where (x_k, y_k) is the object pixel blob coordinates, in the image plane and (V_{x_k}, V_{y_k}) represent the x and y blob velocity.
The evolution and measurements matrix will be:

$$A = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.11}$$

If we have a low noise system can be easy to estimate the objects position. Given covariance matrix of process and measurements Kalman filter, can normally tracks the objects. This can not be true in case of high noise system and can be possible to consider an occlusion as an high noise in the

measurements.
If an object is occluded, the result of detection will be disabled and the object cannot be indicated in the correct way. By increasing the measurements noise covariance matrix to infinite, diminishing the process noise covariance matrix to zero, we can prevent the system from using wrong measurements to update.

$$Q = \begin{cases} diag[10^{-2}, 10^{-2}, 10^{-2}, 10^{-2}] & \text{No Occlusion} \\ diag[10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}] & \text{Occlusion} \end{cases} \tag{4.12}$$

$$R = \begin{cases} \begin{bmatrix} 10^{-1} & 0 \\ 0 & 10^{-1} \end{bmatrix} & \text{No Occlusion} \\ \begin{bmatrix} 10^{7} & 0 \\ 0 & 10^{7} \end{bmatrix} & \text{Occlusion} \end{cases} \tag{4.13}$$

In occlusion case the system will be set in free prediction.

4.4.2 The Hungarian Algorithm for the assignment problem

In our scenario it is possible to find more than one objects or obstacles. For this reason multiple objects tracking and multiple Kalman filter should be implemented.
For each detected object in the image it is necessary to create a single Kalman filter, so it is very important to recognize the same objects (blob of pixels) in different frames.
Literature presents some algorithms that help to resolve this problem, for example:

- **Sift/Surf** that seek for some informative and equal points (key-points) in the two different images.
- **Template Matching** that tries to find a part of image in a bigger image.

- **Cam-shift** that creates a confidence map in the new image based on the color histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position.

All this techniques can't be applied in our case because the image returned by the objects extraction process is a binary image (composed by blobs pixels) and is impossible find informative and different points (there are only pixel with value 1 or 0).

In order to resolve this problem a dataset has been created; containing some important informations about each blob, such as: center coordinates, blob area, blob perimeter, blob first and second moment and so on. This information will be used to applied some controls between blobs pixels in different frames and to build a particular weight cost used in the Hungarian algorithm process.

The Hungarian algorithm [21], is a combinatorial optimization algorithm that solves the assignment problem in a polynomial time $\mathcal{O}(n^3)$.
The assignment problem can be easily formulated in this way: *"if we have n workers for n jobs and the cost w_{ij} to assign the i^{th} worker at the j^{th} job, what is the best assignment that minimize the global cost?"*. So given a set of workers S, a set of jobs T and a cost function $J : [S \cup T] \rightarrow \Re$ the minimization problem can be formulate as:

$$\begin{aligned} \underset{x_{ij}}{\text{minimize}} \quad & J = \sum_{i=0}^{N_{work}} \sum_{j=0}^{N_{job}} c_{ij} x_{ij} \\ \texttt{subject to} \quad & \sum_{i=0}^{N_{work}} \sum_{j=0}^{N_{job}} x_{ij} = \min(N_{job}, N_{work}) \\ & \sum_{i=0}^{N_{work}} x_{ij} \leq 1 \quad \forall j \\ & \sum_{j=0}^{N_{job}} x_{ij} \leq 1 \quad \forall i \end{aligned} \tag{4.14}$$

where c_{ij} is the cost aggregation of worker i at job j.

Equations 4.14 can be resolved a standard *linear binary programming*, however, the Hungarian algorithm provides a simpler way to solve this optimization problem.
This problem can be transformed in a matrix form, building a *Qualification* matrix. Qualification matrix is composed by 0 and 1, in which horizontal rows stand for workers and vertical columns for jobs; a qualified worker is marked by a 1 and unqualified worker is marked with 0. The qualification matrix indicates the optimal assignment and can be obtained as:

1. Find the cost weight for each job-worker assignment.
2. Put the cost obtained at the previous step in a matrix with the same form of qualification matrix
3. Subtract the smallest entry in each row
4. Subtract the smallest entry in each column
5. Find the minimum number of rows and columns so that all the zero entries of the cost matrix are covered
6. *Test for Optimality*: (i) If the minimum number of covering lines is n, an optimal assignment of zeros is possible. (ii) If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.
7. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

In our case the workers will be the pixels blobs at previous frame and the jobs will be the pixels blobs at actual frame. The cost function will be:

$$C(i,j) = \sqrt{x^2 + y^2} \tag{4.15}$$

with

$$x(k) = x_{meas}(k) - x_{pred}(k-1)$$
$$y(k) = y_{meas}(k) - y_{pred}(k-1)$$

where x_{meas} and y_{meas} are the blob center coordinates measured and x_{pred} and y_{pred} are the blob center coordinates predicted by Kalman filter.

Hungarian algorithm can confuse near objects in the image plane. For this reason can be helpful to use the depth information that stereo vision provides. Using the depth information the update step will be done only if the depth value, from previous and actual frame, is included between a setting threshold. For a robust matching, area and perimeter are compared too. After the matching the new parameters founded are stored in the blobs database, which also contain the Kalman filter information.

CHAPTER 5

Validation and Results

5.1 Simulated Environment

In order to test the algorithms described in Chapter 4, a virtual scenario has been created. The environment describes a typical military airport, with some obstacles that are represented by control tower, hangars and other aircraft. With Presagis Vega Prime[1] 3D environments simulator it is possible to create a stereo vision system. This simulator allows to control some stereo vision parameters such as the baseline from the two cameras, field of view, and relative handling between the two image planes. An example of the 3D environment obtained with Presagis Vega Prime is shown in figure 5.1.

The simulator creates a video for each camera. In this case the calibration process is not required, because some camera parameters are already known and there isn't any distortion effect introduced by camera lens; analyzed in Sec. 2.3.

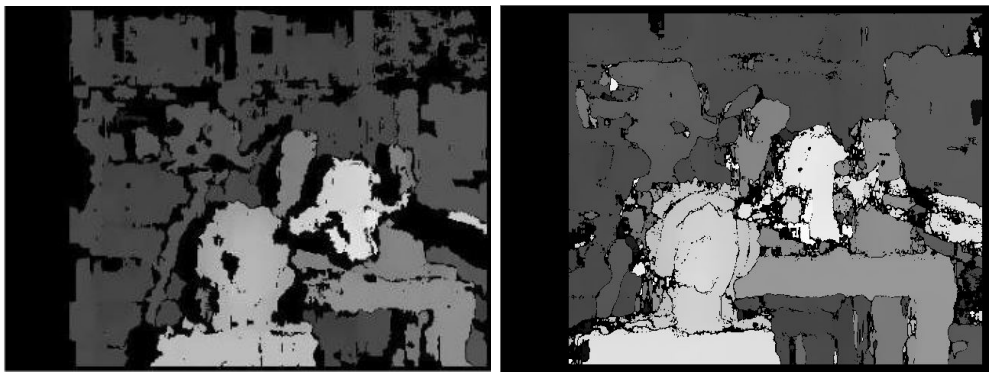
The simulator doesn't allow to set or calculate the focal length of each camera, which is a fundamental parameter to obtain the depth in the scene. An empirical value of focal length has been calculated knowing the distance of an element in the picture (control tower) from observation point. For this

[1]For other informations visit http://www.presagis.com/products_services/products/modeling-simulation/visualization/vega_prime/

Figure 5.1: *3D virtual environment created with Presagis Vega Prime Simulator*

reason the aim of these simulations will not be testing the stereo system, but testing the objects extraction algorithms described in Chapter 4.
One of negatives aspects of this simulator is the impossibility to record a raw video. This simulator allows to record only videos with MPEG compression format. With this compression format, the video stream is sampled and reduced into segments that will be analysed to extract changing information from two consecutive frames. For this reason the two cameras will be not perfectly synchronized and this error produces a noisy disparity map.
In order to validate the algorithm that creates the disparity map, described in Chapter 2, has been used an appropriate dataset[2].
Figure 5.2 shows a comparison between the disparity map obtained in dataset article [22] and the disparity map obtained in this work. The implemented algorithm obtains a better result respect the disparity map obtained by the author of dataset. The disparity map is created with the same parameters used in the article (16 number of disparity).

[2]Tsukuba dataset http://cvlab-home.blogspot.it/2012/05/h2fecha-2581457116665894170-displaynone.html



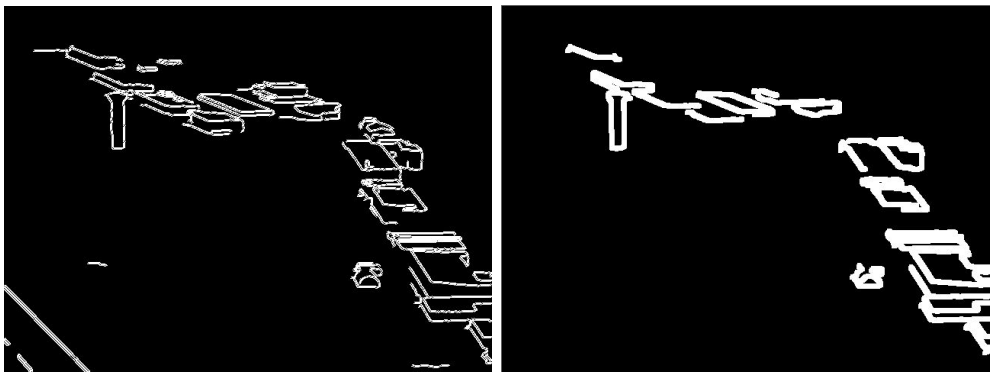
(a) (b)

Figure 5.2: *Comparison between disparity map obtained in [22] and the disparity map obtained in this work 5.2(b)*

5.2 SteViE algorithms on simulated environment

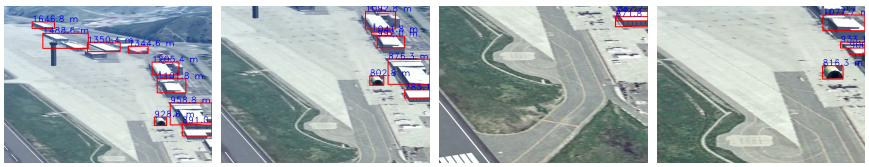
As described in section 4.2, this algorithm merges stereo vision with objects extraction thanks to Canny edges detector (Sec.3.2). The two videos created with Presagis Vega Prime simulator are decomposed in a set of frames. This is possible because the recording frame rate is known.
The acquired images are converted in gray scale and filtered with Gaussian filter 3×3. After that block matching algorithm creates the disparity map and Canny edges extractor finds the edges of the image. This edges are filled and modified with morphological filter as shown by figure 5.3.

The information from Canny step and disparity map are merged to create the final image, where the objects are highlighted with a red rectangle and distance value.
Figure 5.8 shows a frames sequence extracted from the final result. As we can see from the frames sequence, the algorithm finds an high number of obstacles into the scene. The best results are obtained when the objects presents high contrasts. Unfortunately when the objects are so closer between them, or

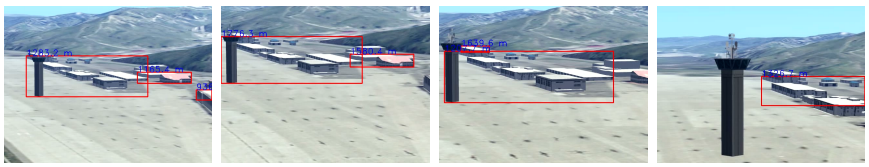


(a) (b)

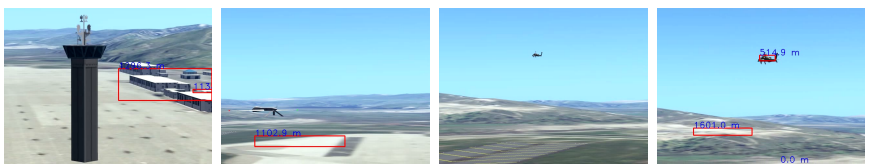
Figure 5.3: *Canny edges detection applied at simulated environment without morphological operations (a) and with morphological operations (b)*



(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150



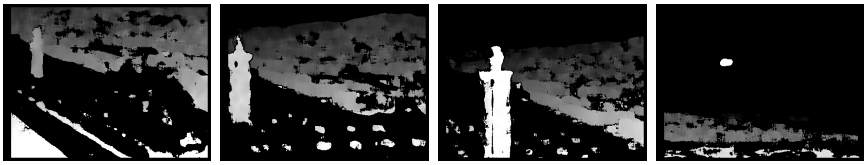
(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350



(i) Frame 400 (j) Frame 450 (k) Frame 500 (l) Frame 550

Figure 5.4: *Frames sequence of SteViE algorithm applied on simulated scenario*

closer to the observer, the performance of this algorithm get worse. As we can see in the frame 5.4(k) and in the frame 5.4(l) there are some false objects detected. This false detections are due to the hills color variation on the background.
Sometimes different edges are recognized as only single edge and for this reason two different obstacles are merged into a single bigger obstacle as figures 5.4(e), 5.4(f) and 5.4(g) show. In order to remove this effect is possible to use morphological operator (sec. 3.3), but this operator can compromise the correct detection of little obstacles far from the observer.
This algorithm presents low computational time, more or less $200ms$ of which 100 used for block matching step. For this reason SteViE algorithm can be used when fast processing time is the main constraint. Analysing each frame, we can see that in mean 60% of obstacles are correctly recognized.
In figure 5.5 are shows some disparity map frames of simulated scenario. As



(a) Frame 180 (b) Frame 320 (c) Frame 360 (d) Frame 560

Figure 5.5: *Disparity map frames of simulated scenario*

previously described, disparity map, is not correctly generated due to the compression format that the simulator uses. Despite this fact, we can see that the control tower and the other aircraft in the scene are clearly detect. The error that stereo system presents in the depth estimation is so high due to the noising disparity map generated.
At the first frame the error in the depth estimation from observer and control tower is about 50 meters. Table 5.1 shows the performance parameters of this algorithm:

SetViE Algorithm Parameters	
Computational time	200 ms
Correct obstacles recognised	60% [20%-100%]
Depth error	about 50 m (with high variation)

Table 5.1: *Performance parameters SteVieE algorithm*

5.2.1 SteViS algorithm on simulated environment

SteViS algorithm is applied at the same scenario in order to find the differences from the two algorithms.
SteViS adds saliency map at the previous algorithm. The simulated images are converted in gray-scale and used to create the disparity map. The same images are used to create the saliency map where the most important features are brighter respect the others. In this way it is possible to segment the saliency map and extract the most saliency parts of an image, that with high probability will be the obstacles in the scene.
An example of disparity map is shown by figure 5.6.

As we can see the saliency map highlights the most important parts of the

Figure 5.6: *Saliency Map of simulated environment*

image, making darker the uninformative pixels and brighter the other pixels. So it is possible to apply a segmentation process to extract the only white pixels as shown in figure 5.7. Figure 5.8 shows a set of frames extract from

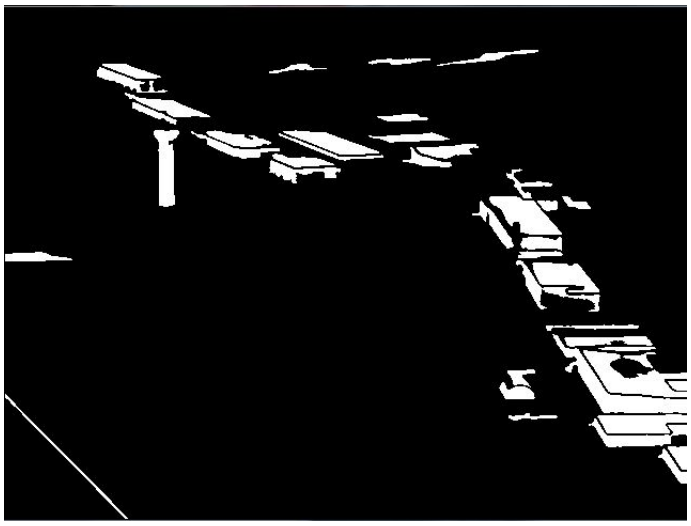
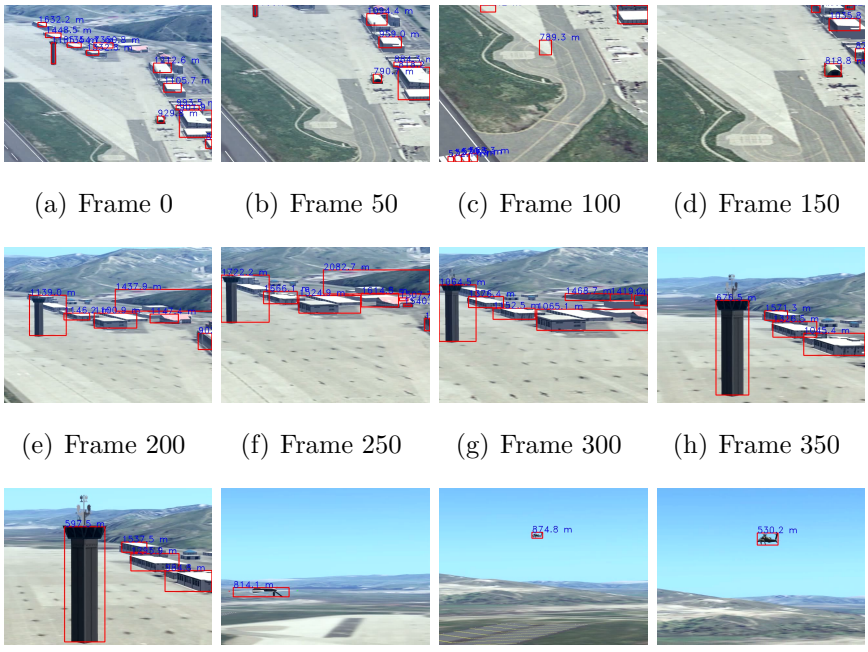


Figure 5.7: *Color segmentation applied at saliency map*

the final result. As we can see from simulation frames this algorithm presents a better result respect SteViE algorithm, finding more objects.
With the segmentation of saliency map the fusion objects problem, that the previous algorithm shows, is now removed and clear results are obtained. The objects are recognized and tracked in all frames as we can see in frames 5.8(e), 5.8(f) and 5.8(g), where the control tower and the other buildings are clearly detect. The other aircraft are recognized too, both in complex background (Fig. 5.8(j)) and in monochromatic background (on the sky Fig. 5.8(k) and 5.8(l)).
With SteViS is possible to obtain a better result respect SteViE algorithm (see Fig. 5.4(k) and 5.4(l)).
The performance parameters are resumed in table 5.2

One of the most negative aspects of this algorithm is the possible false objects detection. This problem occurs when there are high intensity color variations, as we can see in Fig. 5.8(c) where the white stripes on the runway



(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150

(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

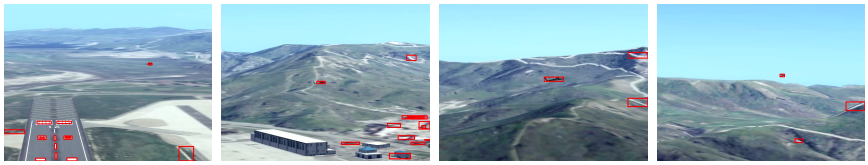
(i) Frame 400 (j) Frame 450 (k) Frame 500 (l) Frame 550

Figure 5.8: *Frame from SteViE algorithm applied on simulated scenario*

SetViS Algorithm Parameters	
Computational time	600 ms
Correct obstacles recognised	80% [50%-100%]
Depth error	about 50 m (with high variation)

Table 5.2: *Performance parameters SteVieS algorithm*

(a) SteViE Fr.0 (b) SteViE Fr.100 (c) SteViE Fr.200 (d) SteViE Fr.500



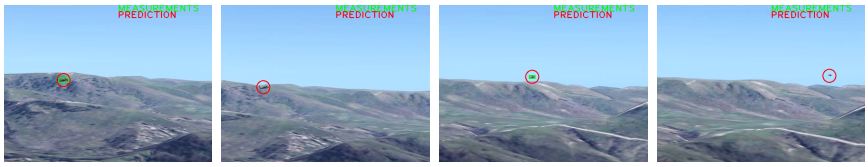
(e) SteViS Fr.0 (f) SteViS Fr.100 (g) SteVis Fr.200 (h) SteViS Fr.500

Figure 5.9: *Helicopter tracking*

are confused with possible objects. This effect is not presented in SteViE algorithm due to the Gaussian filter utilized.

Another important negative aspect is the high computational time that SteViS required. For each frame this algorithm requires $600ms$ that is no good results for real time constraint because there are loss of informations. When, for example, we have fast moving scenario, the salient features are not clear and some of this features are merged. To analyse this effect an helicopter tracking has been simulated. As shown in figure 5.9 SteViS algorithm finds the helicopter when it is so small to see at a glance (Fig. 5.9(e) and 5.9(h)) but it finds some false objects too, as we can see in 5.9(f).

SteViE algorithm doesn't have this problem but it doesn't find the helicopter when it is so small (5.9(a) and 5.9(d)).



(a) frame 100 (b) Frame 150 (c) Frame 200 (d) Frame 250

Figure 5.10: *Tracking with Kalman filter*

To remove the problem of occlusion or temporary loss of detected objects, a Kalman filter has been implemented as described in section 4.4.1.
Figures 5.10 show the differences between the tracking with Kalman filter and without Kalman filter.

5.2.2 Comparison from SteViE and SteViS on simulated scenario

As described in the previous sections the two algorithms present good results, but to maximize each performance can be useful to know some information about the environment where we want to work.
In this way the control parameters of these algorithms can be tuning in the correct way. For a static scene, high attitude surveillance or high attitude aircraft detection, SteViS obtains better result, but SteViE can be useful in take-off or landing operations when fast environment information must be extract.

5.3 Real Scenario

In order to test SteViE and SteViS algorithms a real stereo system has been created.
The system is composed by two cameras mounted on a sliding bar. In this way it can be possible to change the baseline from 50 mm to 700 mm and consequently the maximum depth achievable.
In our test two different IDS cameras have been used, with different charac-

teristics as indicated in Appendix A.
In order to obtain the best results, in a stereo vision system the two cameras should be as similar as possible. For this reason it has been necessary to build a dedicated `C++` code that manages the two cameras and that forced the captured areas and the frame rate to be the same as possible. The frame rates are set at 15 fps but the algorithm acquires the images at each iteration and this time changes with the utilized algorithm.

The color camera has a resolution of 3840×2748 pixels and the gray camera of 1280×1024 pixels. To obtain the same acquired image has been selected a specific region of interest (ROI) centred in the same point with the same resolution of the gray image. After that, another ROI of 640×480 pixel has been selected to have smaller image that help to reduce computational time and lens distortions.
As described in Chapter 2.4 before starting acquisition it is necessary to calibrate the stereo system. This step has been implemented taking 20 pairs of specific pattern in different positions and orientations. The pattern is composed by a rectangular chessboard (in order to determinate the orientation) with 7×9 squares of 28 mm. For a good calibration the chessboard should cover over the 40% of the entire image.
The algorithm (implemented in `C++` with the OpenCv library [4]) starts searching the corner points in a known chessboard using a sub-pixel corners detection. After that, the calibration parameters and the informative matrix have been found and the image points are remapping to obtain the two new rectified images.

The matching process has been realized with Block Matching technique with sum of absolute differences cost aggregation that respects the real-time constraints.
The software created allows to tune all the block matching parameters to achievable the best results in all conditions. The tuning parameters are:

- **SAD windows size**: that controls the size of the windows using for block matching step.

- **Texture threshold**: that helps to remove similar and poor parts of image in order to calculate the disparity only of most variant image parts.
- **Uniqueness ratio**: that is the switch to enable the uniqueness check
- **Pre-filter Size**: that is the size of the filter applied before block matching
- **Number of disparity**: the disparity search range. For each pixel algorithm will find the best disparity from 0 (default minimum disparity) to `ndisparities`. The search range can then be shifted by changing the minimum disparity.
- **Spackle range**: that defines the disparity difference that is considered as speckle

The script implemented allows to tune the Canny and segmentation parameters too.

The obtained disparity values are normalized from 0 to 255 to realize a gray-scale image where the pixel intensity is directly proportional at the disparity value.

The first stereo system implemented has a baseline of 250 mm in order to capture near objects and to obtain a dense disparity map.
The calibration process return the following results:

$$M1 = \begin{bmatrix} 3182,15 & 0 & 309,13 \\ 0 & 3182,15 & 228,83 \\ 0 & 0 & 1 \end{bmatrix} \quad M2 = \begin{bmatrix} 3182,15 & 0 & 311,36 \\ 0 & 3182,15 & 223,08 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} D1 &= \begin{bmatrix} -3,79 & 139,44 & -0,018 & 0,081 & 1792,81 \end{bmatrix} \\ D2 &= \begin{bmatrix} -3,23 & 239,01 & -0,026 & -0,066 & -1092,49 \end{bmatrix} \\ T &= \begin{bmatrix} -24,77 & -21,49 & 7,20 \end{bmatrix} \end{aligned}$$

$$F = \begin{bmatrix} -9,23 \cdot 10^{-8} & -1,80 \cdot 10^{-5} & -2,49 \cdot 10^{-3} \\ 1,83 \cdot 10^{-5} & -6,51 \cdot 10^{-8} & 9,59 \cdot 10^{-2} \\ -2,83 \cdot 10^{-3} & -9,64 \cdot 10^{-2} & -9,99 \cdot 10^{-1} \end{bmatrix}$$

$$E = \begin{bmatrix} -0,036 & -7,201 & -0,209 \\ 7,29 & -0,025 & 12,732 \\ 0,152 & -12,779 & 0,008 \end{bmatrix} \quad R = \begin{bmatrix} 0,99 & 0,004 & -0,007 \\ 0,005 & 0,99 & -0,006 \\ 0,007 & 0,007 & 0,99 \end{bmatrix}$$

With MATLAB calibration toolbox is possible to see the distortion effects introduced by the camera lens:

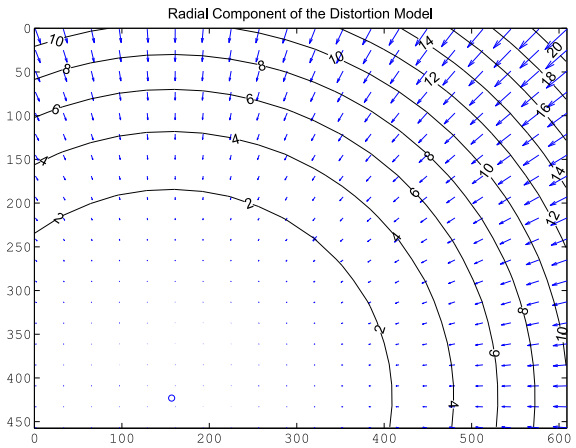


Figure 5.11: *Radial distortion of the left camera*

The implemented algorithms for real scenario are the same of simulated scenario as described in the previous paragraph. As figures 5.15 shows, the calibration and rectification processes have been made properly because we obtained a dense disparity map.

The two benches are at 14 and 20 meters respectively and the depth estimated by the implemented algorithm is about 13.8 meters for the first bench and 19.4 for the second. The uncertainty introduced by pixel quantization is about 2 cm in the first case and 6 cm in the second case. So for closer objects quantization error can be neglected. In this case the error is introduced by the rectification process.

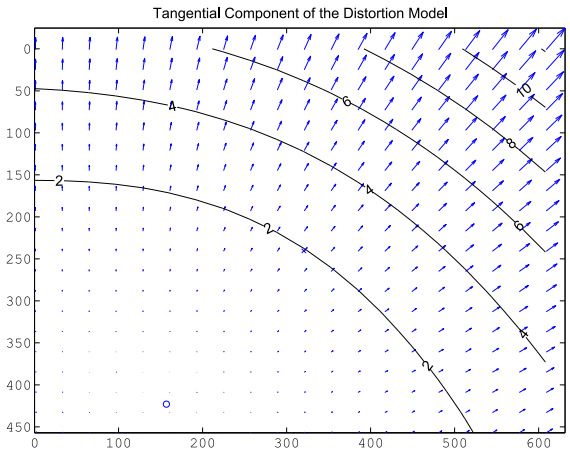


Figure 5.12: *Tangential distortion of the left camera*

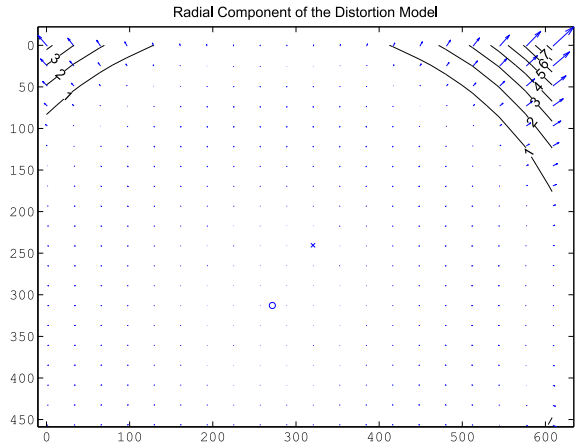


Figure 5.13: *Radial distortion of the right camera*

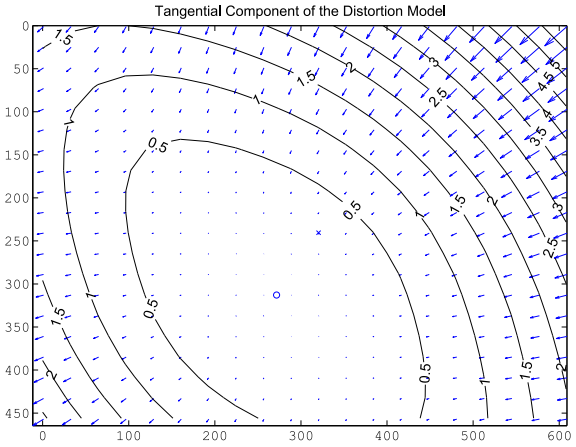
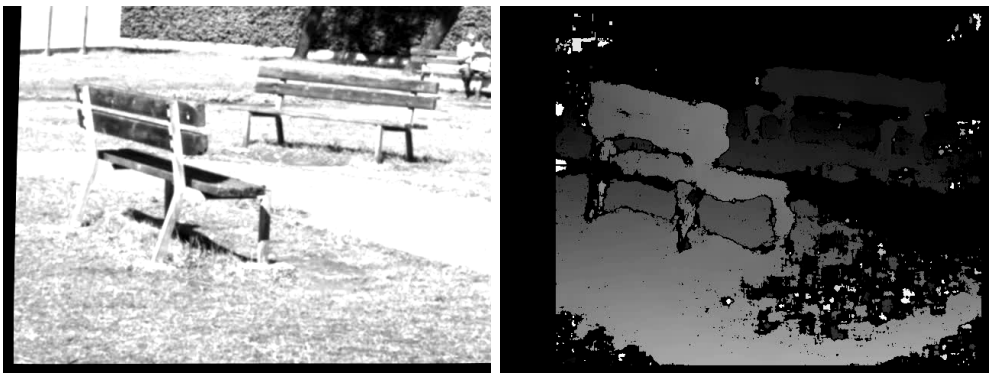


Figure 5.14: *Tangential distortion of right camera*



(a) (b)

Figure 5.15: *Rectified left image and corresponding disparity map*

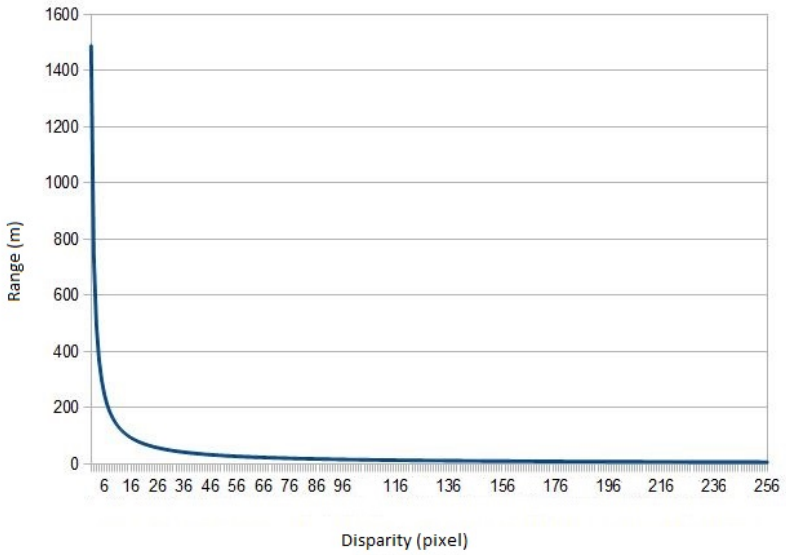


Figure 5.16: *Depth estimation trend with a baseline of 250 mm*

With this configuration the maximum depth achievable is about 1490 meters, but this required only one pixel disparity. The noise in the image and the quantization error introduce high uncertainty as described in figures 5.16 and 5.17. For this reason low disparity values are not considered and disparity values under 5 pixels are neglected.

As we can see from figure 5.17 the uncertainty trend is not linear and became very high at maximum depth achievable. For example at the maximum depth uncertainty is about 290 meters. For this reason disparity values under 5 pixels are neglected.

Figures 5.19 and 5.20 show the system applied on a static scene that describes a little harbour with some ships.

As figure 5.18 shows, the disparity has a dense distribution but presents some variations due to the movement of the sea that disturbs the matching process. Furthermore the disparity map presents a black hole formed by the dark pixels of one ships. The intensity of these pixels doesn't have an high variation and the block match can not matching it properly. This effect

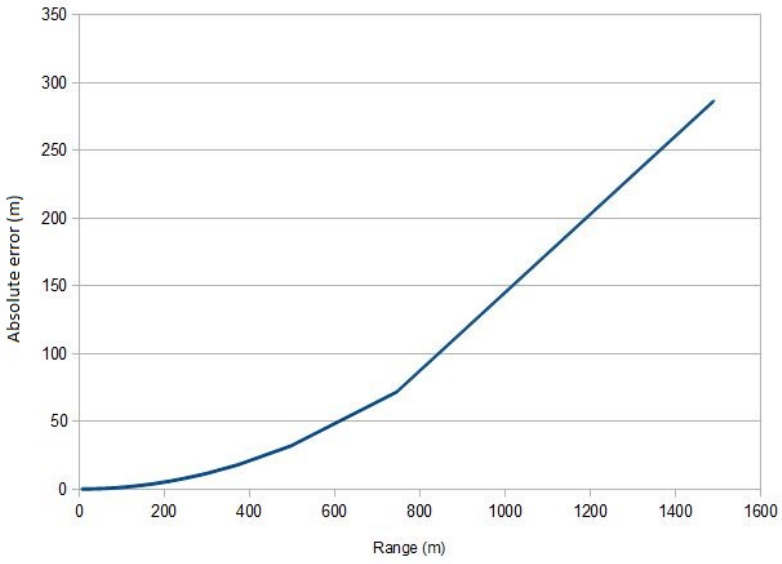


Figure 5.17: *Trend of depth estimation uncertainty with a baseline of 250 mm*

causes a non defined disparity values that is also presented in the sea at far distance. Setting a low SAD size of 15 pixels and using 32 numbers of disparity it is also possible to find the little windsurf in the background. As we can see from disparity map it is so difficult to discriminate the ships from the sea due to the resolution of the system that is lower respect the distance between ships and sea.

Comparing the frames obtained from SteViE and SteViS algorithms, we can see that SteViS algorithm has a better definition respect SteViE that sometimes recognized only some parts of the ships and presents very high variation due to the movement of the sea. SteViS reduces these bad effects because the core of this algorithm is the intensity of the entire objects and not only of the edges as shows by Fig. 5.21.

As we can see from frame 5.20(h) SteViS algorithm recognizes a little windsurf in the background that SteViE algorithm doesn't recognise. The other windsurfs are not recognized because they are closer to the skyline and confused with it.

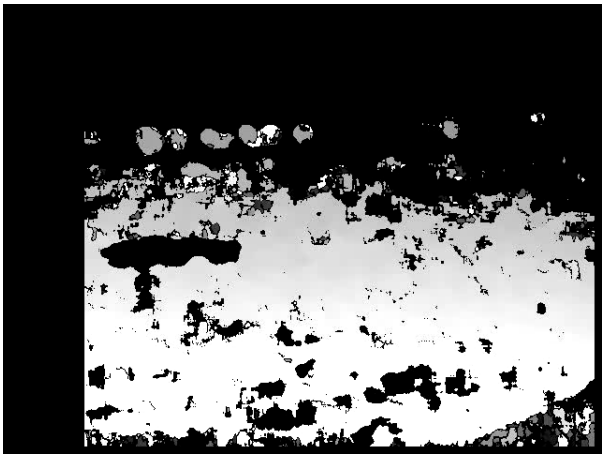


Figure 5.18: *Disparity of harbour with some ships*

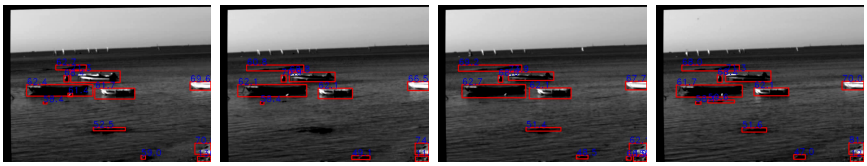
The SteViS algorithm presents better results that allow to recognize more ships and have a better stability in the image.

The same system is applied at a different scenario that contains moving objects (aircraft) and low complexity background. In this case the goal of the system is to recognize only the landing aircraft.

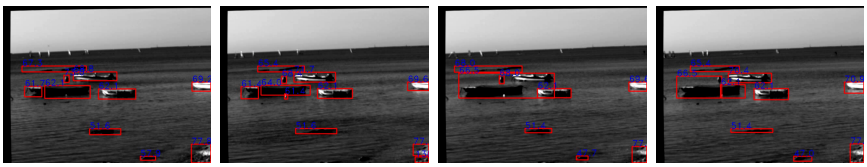
Figures 5.22 and 5.23 show the aircraft recognition. As we can see SteViE 5.22 presents more false recognition respect SteViS, but both algorithms recognized the landing aircraft. In fact either the two algorithms find the aircraft when it is impossible to see at glance.

Since our goal is to find the aircraft in the scene, the objects with a length over the 80% of image size are neglected. As we can see from figures 5.22 and 5.23 the skyline, with the house and trees, is not recognized as an object. In pictures 5.22(c) and 5.23(c) the two algorithms detect a flock of birds.

In this case the obtained disparity map doesn't allow to estimate the distance from the aircraft because the baseline is too short. For this reason the baseline has been increased at 700 mm (the maximum length achievable by this system). Obviously a new calibration has required. From the new calibration process the following results are obtained:

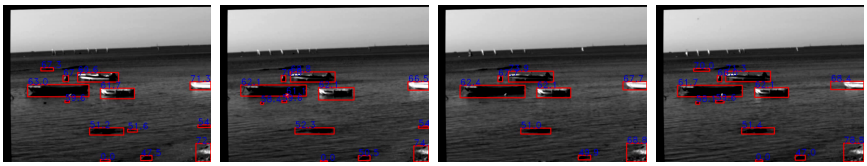


(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150

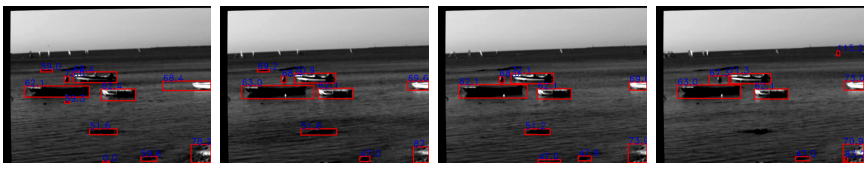


(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

Figure 5.19: *SteViE algorithm applied in real scenario*



(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150



(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

Figure 5.20: *SteViS algorithm applied in real scenario*

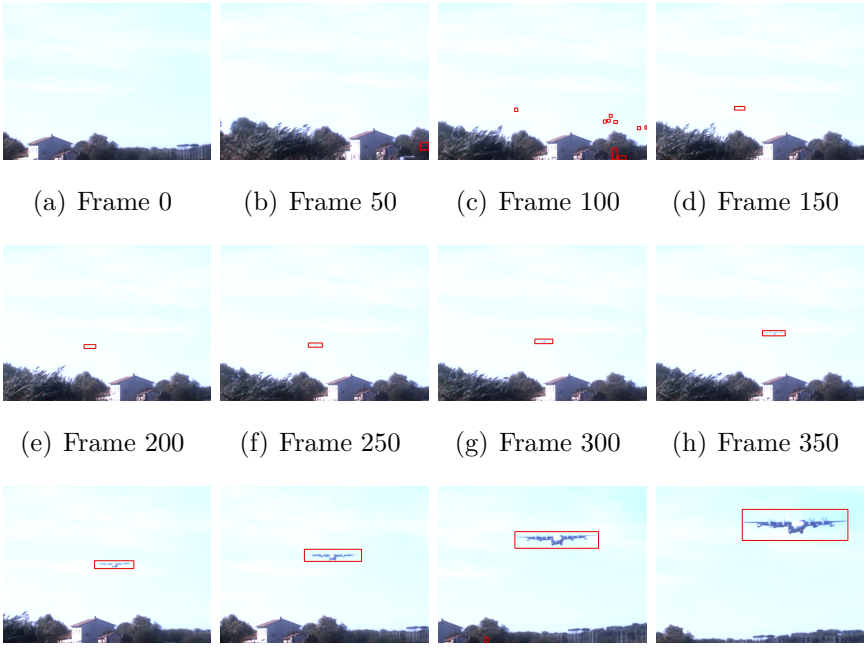
Figure 5.21: *Saliency of harbour scenario*

(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150

(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

(i) Frame 400 (j) Frame 450 (k) Frame 500 (l) Frame 550

Figure 5.22: *Frames sequence of SteViE algorithm applied on aircraft landing*



(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150

(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

(i) Frame 400 (j) Frame 450 (k) Frame 500 (l) Frame 550

Figure 5.23: *Frames sequence of SteViS algorithm applied on aircraft landing*

$$M1 = \begin{bmatrix} 6,02 \cdot 10^3 & 0 & 2,07 \cdot 10^2 \\ 0 & 6,02 \cdot 10^3 & 1,41 \cdot 10^2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 6,07 \cdot 10^3 & 0 & 1,17 \cdot 10^2 \\ 0 & 6,07 \cdot 10^3 & 3,44 \cdot 10^2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} D1 &= \begin{bmatrix} 7,42 & -2,56 \cdot 10^2 & 0 & 0 & 3,55 \cdot 10^3 \end{bmatrix} \\ D2 &= \begin{bmatrix} 0,39 & 5,61 \cdot 10^{-1} & 0 & 0 & -1,22 \cdot 10^3 \end{bmatrix} \\ T &= \begin{bmatrix} -6,95 \cdot 10^1 & 3,63 & -3,83 \cdot 10^1 \end{bmatrix} \end{aligned}$$

$$F = \begin{bmatrix} 1,97 \cdot 10^{-7} & -6,22 \cdot 10^{-6} & -1,48 \cdot 10^{-3} \\ 3,21 \cdot 10^{-6} & -1,35 \cdot 10^{-7} & -7,16 \cdot 10^{-2} \\ -1,60 \cdot 10^{-3} & 7,38 \cdot 10^{-2} & -9,99 \cdot 10^{-1} \end{bmatrix}$$

$$\begin{aligned} E &= \begin{bmatrix} -1,22 & 3,84 \cdot 10^1 & 2,52 \\ -1,98 \cdot 10^1 & 8,33 \cdot 10^{-1} & 7,50 \cdot 10^1 \\ 2,69 \cdot 10^{-1} & 6,75 \cdot 10^1 & 2,68 \end{bmatrix} \\ R &= \begin{bmatrix} 9,65 \cdot 10^3 & 4,75 \cdot 10^{-2} & -2,56 \cdot 10^{-1} \\ -5,60 \cdot 10^{-2} & 9,98 \cdot 10^{-1} & -2,58 \cdot 10^{-2} \\ 2,54 \cdot 10^{-1} & 3,93 \cdot 10^{-2} & 9,66 \cdot 10^-1 \end{bmatrix} \end{aligned}$$

In this configuration the maximum depth achievable is about 4173 meters, but we consider disparity over 3 pixels and under 1000 meters.

Figures 5.27 shows a landing aircraft at Pisa airport. As we can see in the first frame the system fails in the depth estimation because the aircraft is farther respect the maximum depth achievable. The system starts to estimate the aircraft depth from about 800 m.

Testing the system we obtain an error over 20 meters on an estimate distance

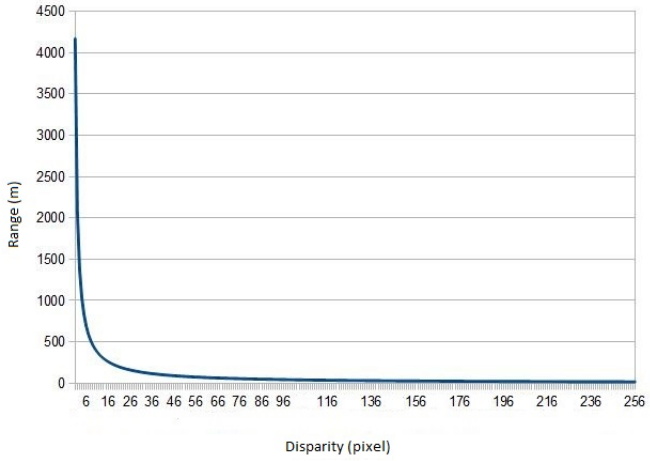


Figure 5.24: *Depth estimation trend with a baseline of 700 mm*

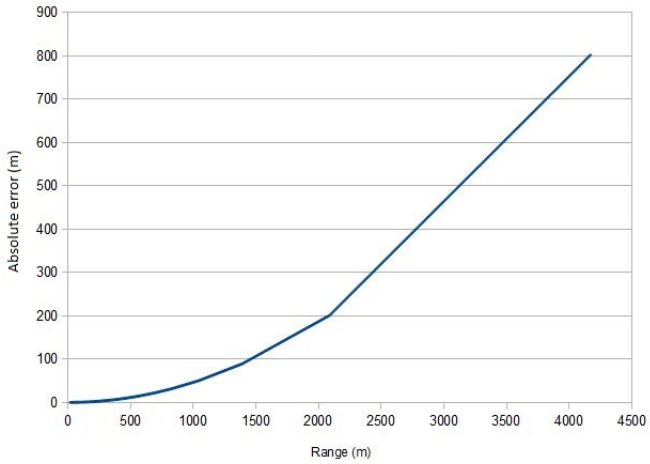


Figure 5.25: *Depth uncertainty estimation trend with a baseline of 700 mm*

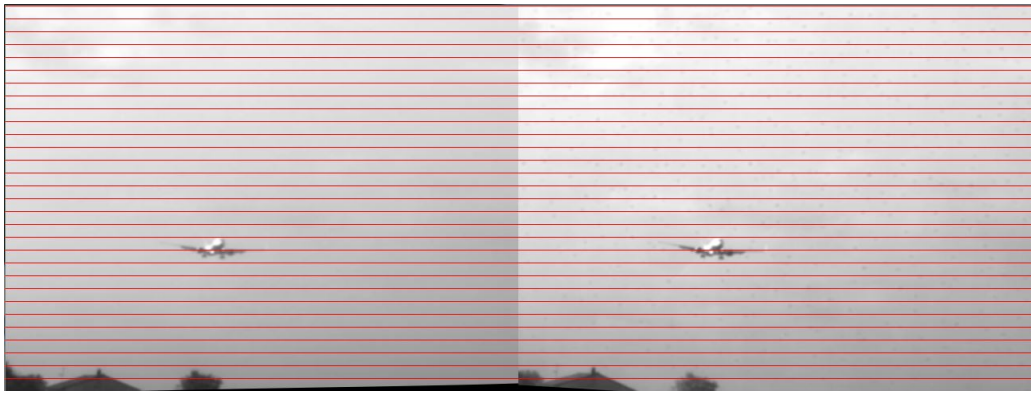


Figure 5.26: *Pair of rectified images*

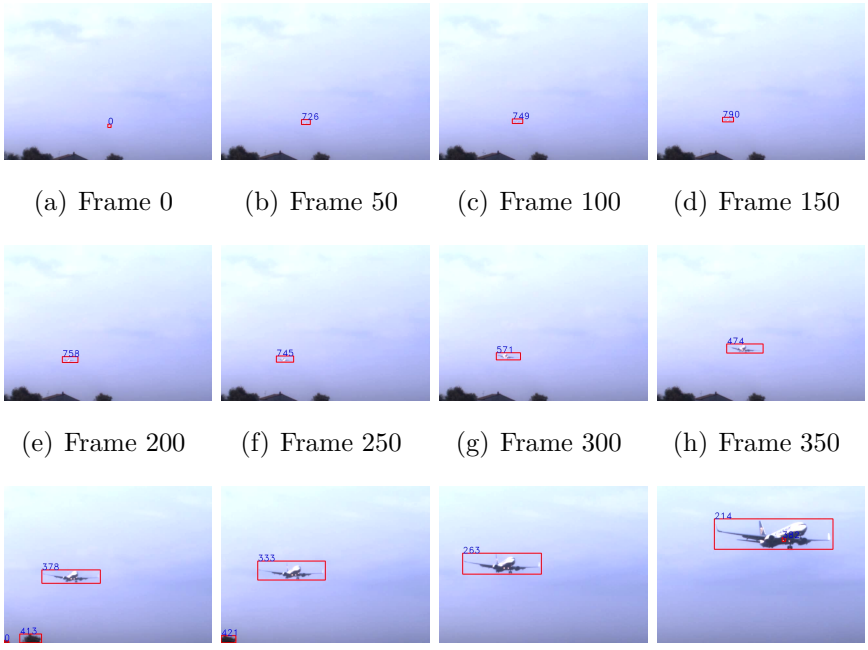
of 300 meters.
In order to remove outliers and have a smoothing depth variation has been introduced a first order fading filter. The filter estimate is a linear combination of the old estimate and a gain that multiplies the residual. Difference among filter is the order of the dynamics that generates the new estimate assuming a polynomial model of the process.

$$\begin{cases} \hat{x}_n = \hat{x}_{n-1} + \alpha[\tilde{x}_n - \hat{x}_{n-1}] \\ \alpha = 1 - \beta \end{cases} \tag{5.1}$$

where $\hat{x}_n$ is the estimated depth value at step n, $\hat{x}_{n-1}$ is the depth estimation value at previous step and $\tilde{x}_n$ is the measured depth value at actual step. Changing $\beta \in [0, 1]$ is possible to change the memory of the filter.

Due to the low complexity of the scene there aren't any differences form the two algorithms. The different between SteViE and SteViS is only the computational time that is about 100 ms for SteViE and 600 ms for SteViS. So in this case is preferable to use SteViE algorithm.

After that, the image coordinates of the landing aircraft are re-projected from 2D image plane to 3D camera space using the relation described in Sec. 2.3. From the knowledge of the pose of the the stereo rig, from consecutive frames respect the real world coordinates system, it is possible to georeference the aircraft in the world.



(a) Frame 0 (b) Frame 50 (c) Frame 100 (d) Frame 150

(e) Frame 200 (f) Frame 250 (g) Frame 300 (h) Frame 350

(i) Frame 400 (j) Frame 450 (k) Frame 500 (l) Frame 550

Figure 5.27: *Frames sequence of SteViS algorithm applied on aircraft landing*

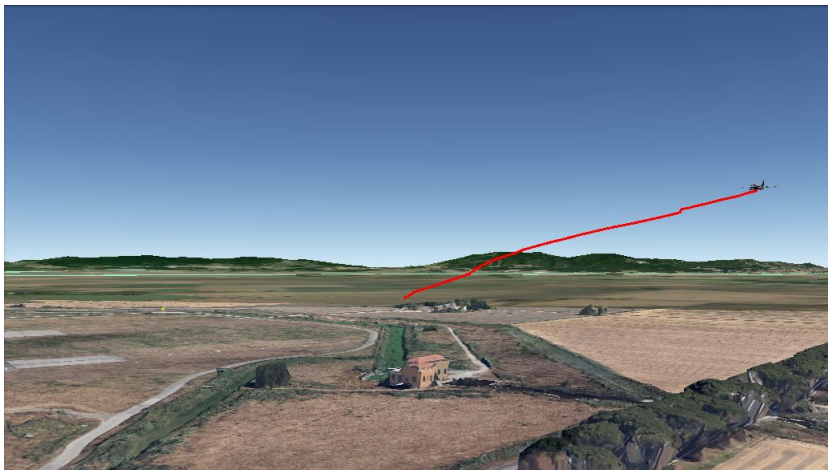


Figure 5.28: *Trajectory of landing aircraft on Google Earth*

The aircraft coordinates are transformed into latitude, longitude and altitude values in order to track the trajectory in the real world as figure 5.28 shows.

All the tests described, both in simulated and real environment, are executed on 64-bit Intel Core Duo CPU T6570 at 2.10 GHz with 4 Gb RAM system.

CHAPTER 6

Conclusions and future works

This work described two different algorithms for obstacle detection and range estimation that can be applied for unmanned aircraft vehicles. To tackle this, stereo vision and image processing theories have been studied. These two different concepts have combined to realize two algorithms, called SteViE (STEreo VIsion Edges) and SteViS (STEreo VIsion Saliency) respectively, that try to recognize objects in the environment and to estimate the distance between the observer and the obstacles.

At the beginning, stereo vision theory was presented to describe the fundamental steps that help to modify the images acquired and extrapolate the required information. In order to respect the real time constraint, the calibration process and both the two algorithms are implemented in `C++` code with some dedicated libraries.

The two algorithms were applied in simulated and real scenarios. As explained in Chapter 5 both the two algorithms presented good results. If we have some information about the environment where we are working, is possible to choose which algorithm is better to use in order to obtain the best results.

The core of the first algorithm is the Canny edges detector, used to find the edges of the obstacles in the scene.
As explained, SteViE algorithm is faster with respect to SteViS and should be used when we have a fast-varying scenario in order to capture as much

information as possible. The weakness of the system is the error objects recognition in case of low-intensity variation or in presence of objects close to each other that are recognized as a single bigger object.

SteViS algorithm circumvents all the problems of SteViE algorithm but its computational time is bigger with respect to the other algorithm.
The core of SteViS algorithm is the computation of the saliency map, that highlights the most important parts of the image, in agreement with a particular definition of saliency. This process presents a high computational burden, but provides better results. In fact, with a segmentation of saliency map we can isolate the most salient parts of the image, that with high probability correspond to the objects we want to detect in the captured figure. The SteViS algorithm is a clear improvement of other existed algorithms because it does not use color segmentation on the raw captured image and thus can be applied to detect a number of different objects whose color is not known a priori. For this reason it can be applied in a real scenario.

Canny edges detector finds the intensity variation only of the obstacle's corner pixels whereas saliency studies the intensity of all detected obstacles pixels.

In order to have the fastest possible system, the graph based image segmentation was utilized on saliency map. Segmentation can be applied at this stage because the saliency map produces a gray-level image where darker pixels correspond to uninformative elements and brighter pixels correspond to saliency elements.

For this reason SteViS algorithm should be used in more complex scenarios where only slow variations occur, in order to lose as less information as possible.

The range estimation step is the same for both the two algorithms. The obtained results are so sensitive to the calibration and rectification processes, that is important to make them in a careful way to obtain accurate range

estimates. As previous mentioned, the error in the range estimation can be quite high because there are uncertainties introduced by pixels quantization, matching and rectification processes. In our case, more error is also given by the differences of the utilized cameras. In fact the two cameras should be as similar as possible to have the best results. For all the previous reasons the range estimate error became very high in the case of far objects.

During this work we have identified some critical elements in the range estimation problem, that are list below:

- **Camera Problems**: In this work two different cameras have been used and dedicated manage software was used to obtain two identical pictures. This was done using the cameras own drivers.

- **Loss of detected objects**: To remove the possibility of losing the detected objects in the picture, when an occlusion occurs, a Kalman filter has been implemented to estimate the position of the objects in image coordinates.

- **Improve calibration results**: In order to obtain a better calibration result, in the edges extraction process from chessboard pattern, a sub-pixel edges extraction has been done. In this way the positions of the squares corners were captured in a much more accurate way.

- **Noise in binary image**: To remove the noise in the disparity map and in the binary images, obtained by canny detection and saliency map, complex morphological operators were implemented.

- **Error matching in Hungarian algorithm**: If two objects are close to each other, their cost function can be similar and it is possible that the algorithm does not match the objects in the correct way. To help the algorithm to remove this problem, the range value was introduced and a matching was done only if its range variation, between consecutive frames, was below a given threshold.

In order to remove the weak points of the implemented system, we have identified some future works of interest. One future step is to use two identical cameras. In this way, a better and more accurate result will be obtained, with less distortions in the rectified images.
In order to have a faster system, it can be useful to use a dedicated hardware architecture, that can help to remove loss of information. In this way Semi Global Block matching and graph based image segmentation can be used respecting real time constraint, obtaining a better result in the same time of the system implemented here.
To mitigate the error in range estimation it can be possible to implement a varying baseline system using, for example, more than two cameras. In this way we can use the farthest cameras to estimate the range of far objects and the closest cameras to estimate the position of near objects. In this way we can select the appropriate cameras according to the task that UAVs should perform.
Another important future work will be to combined the stereo vision information with the IMU (Inertial Measurements Unit) and GPS (Global position System) information, installed in the UAV. In this way it is possible to know the position and the attitude of the UAV with high precision and detect the obstacles in a real geographic system coordinates. In this way it is also possible to support the GPS signal, with the information produced by the stereo vision system, thus mitigating problems arising from losing GPS signal. This navigation technique is also known as Visual Odometry, i.e. the process of determining the position and attitude of an unmanned system by analysing the corresponding camera images. Such a merging problem between IMU and visual information, has already been preliminary studied in [23].

APPENDIX A

Camera Datasheet

Parameters	IDS UI-1490SE-C-HQ	EO-1312M
Sensor Technology	CMOS	CMOS
Manufacture	Aptina	Aptina
Resolution ($h \times v$)	3840 × 2748	1280 × 1024
Color Depth (sensor)	12 bit (color)	8 bit (monochrome)
Pixel Class	10MP	1.3MP
Shutter	Rolling Shutter	Rolling Shutter
fps in FreeRun Mode	3.2	25
Binning Modes	Color	Monochrome
Subsampling Modes	Color	Monochrome
Sensor Model	MT9J003STC	MT9M001
Pixel Size	1.67 ηm	5.2 ηm
Optical Size	4.589 mm × 6.413 mm	6.6 mm × 5.3 mm
Focal Length	8 mm	25 mm

Table A.1: *Principal parameters of camera used. For more information* http://en.ids-imaging.com

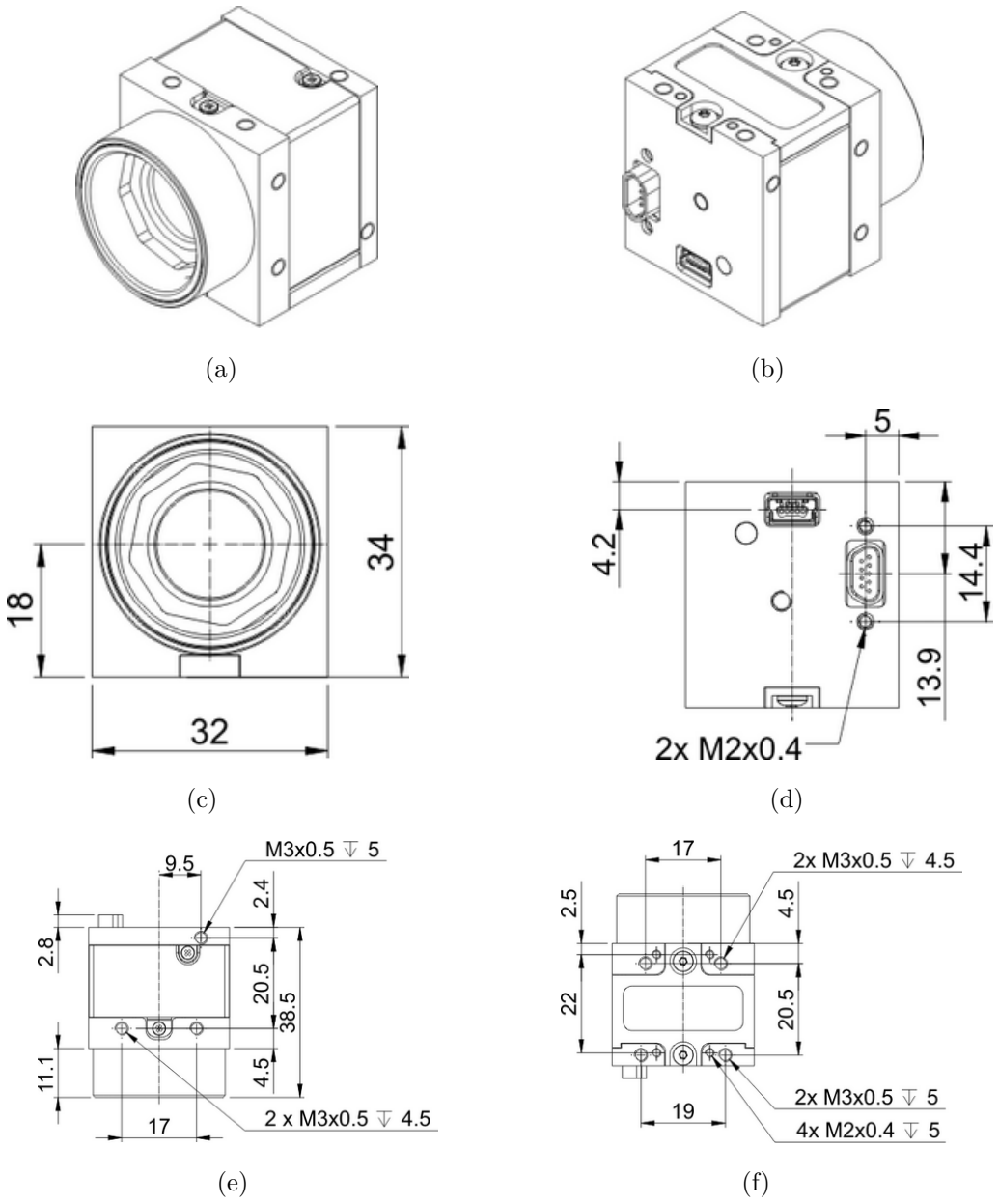


Figure A.1: *Mechanical specifications of used cameras. [a-b] 3D view, [c] rear view, [d] top view, [e] bottom view*

Bibliography

[1] R. E. Zelenka and L. D. Almsted. Design and flight test of 35-gigahertz radar for terrain and obstacle avoidance. *Journal of Aircraft*, 1997.

[2] Midair Collisions. W.h.harman tcas: A system for preventing.

[3] Richard Szeliski. *Computer Vision: Algorithms and Applications (Texts in Computer Science)*. Springer, 2011 edition, November 2010.

[4] Dr. Gary Rost Bradski and Adrian Kaehler. *Learning opencv, 1st edition*. O'Reilly Media, Inc., first edition, 2008.

[5] J. Y. Bouguet. Camera calibration toolbox for Matlab, 2008.

[6] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *in ICCV*, pages 666–673, 1999.

[7] C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. Technical Report MSR-TR-99-21, Microsoft Research, June 2001.

[8] Andrea Fusiello, Vito Roberto, and Emanuele Trucco. Efficient stereo with multiple windowing. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 0, pages 858–863, Los Alamitos, CA, USA, 1997. IEEE Comput. Soc.

[9] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE Interna-*

tional Conference on, volume 2, pages 1150–1157 vol.2, Los Alamitos, CA, USA, August 1999. IEEE.

[10] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[11] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140. IEEE, 2001.

[12] Christian Banz, Sebastian Hesselbarth, Holger Flatt, Holger Blume, and Peter Pirsch. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In Fadi J. Kurdahi and Jarmo Takala, editors, *ICSAMOS*, pages 93–101. IEEE, 2010.

[13] Heiko Hirschm. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, June 2007.

[14] Nikhil R. Pal and Sankar K. Pal. A review on image segmentation techniques. *Pattern Recognition*, pages 1277–1294, 1993.

[15] S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France.*, September 1979.

[16] Yair Weiss. Segmentation using eigenvectors: A unifying view. In *In ICCV*, 1999.

[17] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:2004, 2004.

[18] J. Canny. A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, November 1986.

[19] Frank Moosmann, Diane Larlus, and Frederic Jurie. Learning saliency maps for object categorization. 2006.

[20] Simone Frintrop. *VOCUS: A Visual Attention System for Object Detection and Goal-directed search*. PhD thesis.

[21] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[22] M. Peris S. Martull and K. Fukui. Realistic cg stereo image dataset with ground truth disparity maps. In *ICPR2012 workshop TrakMark2012*.

[23] L.Semeraro L.Sestini, R.Incaini. Implementation of measurements augmented unscented kalman filter with epipolar constraints. Technical report, Final project of "Sistemi di guida e navigazione" University of Pisa, 2013.

[24] Jeffrey Byrne, Martin Cosgrove, and Raman Mehra. Stereo based obstacle detection for an unmanned air vehicle, 2006.

[25] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[26] On Call, Y Beard, Clark Taylor, and Blake Barber. Obstacle avoidance for unmanned air vehicles using image feature tracking. In *AIAA Guidance, Navigation, and Control Conference*, 2006.

[27] Richard Wallace, Ping-Wen Ong, Ben Bederson, and Eric Schwartz. Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90, 1994.

[28] Raman Mehra J. Byrne, Martin Cosgrove. Real time stereo based obstacle detection for uav threat avoidance.

[29] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. 1989.

[30] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2004.

[31] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[32] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact Maximum A Posteriori Estimation for Binary Images.

[33] Yuichi Ohta and Takeo Kanade. Stereo by intra- and inter-scanline search using dynamic programming, 1985.

[34] S.A.M. Buljan and Pontificia Universidad Católica de Chile. Escuela de Ingeniería. *Human detection using uncontrolled moving cameras and novel features derived from a visual saliency mechanism*. Pontificia Universidad Católica de Chile, 2008.

[35] Jiebo Luo and David J. Crandall. Color object detection using spatial-color joint probability functions. *IEEE Transactions on Image Processing*, 15(6):1443–1453.

[36] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.