POSITION ESTIMATION USING SATELLITE IMAGES

OMAR AMIL HAZZAA

OCTOBER 2014

POSITION ESTIMATION USING SATELLITE IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
ÇANKAYA UNIVERSITY

BY
OMAR AMIL HAZZAA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
MATHEMATICS AND COMPUTER SCIENCE

OCTOBER 2014

Title of the Thesis : **Position Estimation Using Satellite Images.**

Submitted by **Omar Amil HAZZAA**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Billûr K

Prof. Dr. Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Hadi Hakan MARAŞ

Supervisor

Examination Date: 10.10.2014

Examining Committee Members

Assoc. Prof. Dr. Hadi Hakan MARAŞ	(Çankaya Univ.)	
Assist.Prof.Dr. A.Kadir GÖRÜR	(Çankaya Univ.)	
Assoc.Prof.Dr. Fahd JARAD	(THK Univ.)	

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Omar Amil HAZZAA

Signature :

Date : 10.10.2014

ABSTRACT

POSITION ESTIMATION USING SATELLITE IMAGES

HAZZAA, Omar Amil
M.Sc., Department of Mathematics and Computer Science
Supervisor: Assoc. Prof. Dr. Hadi Hakan MARAŞ

October 2014, 111pages

This thesis deals with estimating position and orientation in real time, using visual measurements. A system has been developed with which to solve this problem for unprepared environments, assuming that a map or scene model is available. Compared to ‘camera-only’ examples, the developed system is accurate and robust, and can handle periods of uninformative or no visual data, reducing the need for high frequency visual updates. The system achieves real-time pose estimation via the use of a framework of nonlinear state estimation for which state space models have been developed. System performance was evaluated using an augmented reality application in which the system output was used to superimpose virtual graphics on the live video stream. Furthermore, experiments were performed in which an industrial robot providing ground truth data was used to move the sensor unit. In both cases the system performed well. Calibration of the relative position and orientation of the camera was found to be essential for proper system operation. A new and easy-to-use algorithm with which to estimate these data was developed using a stereo visual approach. Experimental results revealed that the algorithm works well in practice.

Keywords: GPS, Position Estimation, Vision Localization, Stereo Vision.

ÖZ

UYDU GÖRÜNTÜLERİNİ KULLANARAK POZİSYON TAHMİNİ BULMAK

HAZZAA, Omar Amil
Yüksek Lisans, Matematik ve Bilgisayar Anabilim Dalı
Tez Yöneticisi: Doç. Dr.Hadi Hakan MARAŞ

Ekim 2014, 111 sayfa

Bu tezde, görsel ölçümler kullanılarak, gerçek zamanlı konum ve yön tahminden bahsedilmiştir. Bu tezde hazırlıksız ortamlarda bir harita veya bir senaryo modeli mevcut olduğu varsayılarak, bu sorunu çözmek için bir sistem geliştirilmiştir. Sadece camera örnekleri ile kıyaslandığında, geliştirilmiş sistem doğru, sağlam, yüksek frekans ve görsel güncellemelerin ihtiyacını azaltarak, uninformative veya hiçbir görsel veri olmadan işleyebilmektedir. Geliştirilen sistem gerçek zamanlı durum uzayının modellerini geliştirerek doğrusal olmayan durumun çerçeve kullanımı yoluyla tahmini poza ulaşmaktadır. Sistem performansı sistem çıkışının canlı video akışını, sanal grafiğe ekleyerek artırılmış gerçeklik uygulamasını kullanımıyla geliştirilmiştir. Ek olarak, deneyler sensor birimini hareket ettirmek için bir endüstriyel robota gerçek yer verilerini sağlayarak yapılmıştır. Her iki durumda da, sistem iyi bir performans sergilemiştir. Göreli konumu ve yönelimi kameranın kalibrasyon sisteminin düzgün çalışması için gerekli olduğu tespit edilmiştir. Görsel stereo yaklaşımı ile kolay kullanımlı algoritma kullanarak, yeni bir algoritma sunulmuştur. Deneysel sonuçlar ise geliştirilen algoritma uygulama kısmında iyi performans ile çalıştığını gösterilmiştir.

Anahtar Kelimeler: GPS, Konum Tahmini, Vizyon Yerelleştirme, Stereo Görme.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Assoc. Prof. Dr. Hadi Hakan MARAŞ for his supervision, special guidance, suggestions, and encouragement through the development of this thesis.

I would like to express my sincere gratitude to Prof. Dr. Zeki KADOORI who was teach me how to thinking as a programmer with his best to get to this stage

It is a pleasure to express my special thanks to my family for their valuable support.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.. iii
ABSTRACT.. iv
ÖZ…………………………………………………………………………….. v
ACKNOWLEDGEMENTS………………………………………………….... vi
TABLE OF CONTENTS…………………………………………………….. vii
LIST OF FIGURES……………………………………………………….... ix
LIST OF TABLES………………………………………………………… xiii
LIST OF ABBREVIATIONS………………………………………………... xv

CHAPTERS:

1. INTRODUCTION... 1

- **1.1.** Overview... 2
- **1.2.** Motivation... 4
- **1.3.** Objective……………………... 4
- **1.4.** Organization of the Thesis………... 5

2. LITERATURE REVIEW FOCUSSING ON THE DEVELOPMENT OF IMAGE-BASED POSITIONING 6

- **2.1.** Landmark-Based Methods... 6
- **2.2.** Trajectory Integration and Dead Reckoning.................................. 8
- **2.3.** Standard Pattern……………………………………………….. 11
- **2.4** A Priori World Model Knowledge………………………………. 12

3. IMAGE GEOMETRY.. 14

- **3.1.** Transformation.. 14
- **3.2.** Perspective Transformation.. 18
- **3.3.** Camera Model... 19
- **3.4.** Lens Distortion.. 21
- **3.5.** Camera Calibration... 23
- **3.6.** Camera Location... 30

3.7. Camera Orientation 31
3.8. Camera Pose Estimation 32
3.9. Epipolar Geometry 36
3.10. Recovering Image Depth 'Z' 39
3.11. Object Size Estimation 41

4. IMAGE PROCESSING 44
4.1. Color Spaces. 45
4.2. Image Enhancement 58
4.3. Edge Detection 66
4.4. Region Segmentation 70
4.5. Template Matching 75
4.6. Integral Images 83
4.7. Speeded Up Robust Features (SURF) 84

5. THE APPLICATION 90
5.1. Vision System Calibration 92
5.2. Image Pre-Processing 98
5.3. Region Segmentation 100
5.4. Stereo Vision 101
5.5. Comparison 105
5.6. Position Estimation 106

6. Result 108
6.1 Results 108
6.2 Conclusion 110
6.3 Future Work 110

REFERENCES R1
APPENDICES A A1
A. CURRICULUM VITAE A1

LIST OF FIGURES

FIGURES

Figure 1 Visual odometry system ………………………………………. 3

Figure 2 Example of landmark. ………………………………………… 8

Figure 3 Example of trajectory integration and dead reckoning............... 11

Figure 4 Example of standard pattern... 12

Figure 5 Example of a priori world model knowledge.............................. 13

Figure 6 Pinhole camera model …………………………...…….……. 18

Figure 7 Barrel distortion ………………………………………………. 21

Figure 8 Pincushion distortion.. 22

Figure 9 Mustache distortio...…………………………………………… 22

Figure 10 Photogrammetric calibration object …………………………… 23

Figure 11 Determining camera location ……………………….....…… 31

Figure 12 Camera orientation...……......…..…......……......……......…….. 32

Figure 13 Pose algorithm ……….. 35

Figure 14 Epipolar geometry.. 36

Figure 15 Image depth ……………......………......……..….................. 40

Figure 16 Example of size estimation ………………………….……...… 41

Figure 17 Triangle between object and camera …………..…….……...… 42

Figure 18 Triangle of the object inside the camera …………......………. 42

Figure 19 Pythagoras' proof ………..………………..……….........……. 43

Figure 20 The RGB color cube ………..………………………..…......... 45

Figure 21 Gray color space methods ....………………………........ 47

Figure 22 YUV color space …………..…………….……………........... 47

Figure 23 RGB conversion to YUV…...…………….…………….......... 49

Figure 24 YUV conversion to RGB…..………………..……………........ 49

FIGURES

Figure 25 YIQ color space.………….…………………..……………...... 50

Figure 26 RGB conversion to YIQ…..…………………..……………...... 51

Figure 27 YIQ conversion to RGB…..…………………..……………...... 51

Figure 28 HSI color space.……..……..…………………..……..……... 52

Figure 29 HSI color circle.…...………..…………………..……………... 52

Figure 30 Color selection in an RGB image using the HSI color system.... 53

Figure 31 RGB conversion to HSI....…..…………………..…………….. 54

Figure 32 HSI conversion to RGB…..…………………..……………...... 55

Figure 33 HSV color space system…..…………………..…………….... 56

Figure 34 RGB conversion to HSV…..…………………..…………….... 57

Figure 35 HSV conversion to RGB…..…………………..……..……... 57

Figure 36 Block diagram of image enhancement……………………….... 58

Figure 37 Basic intensity transformation functions..................................... 59

Figure 38 Gray scale inversion.........……………………..……………... 59

Figure 39 LOG enhancement image....……………………..…………….. 60

Figure 40 Plots of the power-law function.………………..……..……… 61

Figure 41 Example of power-law function application.…………………... 61

Figure 42 Example of contrast stretching.. 62

Figure 43 Mean filter applied in matrix form.. 63

Figure 44 Mean filter applied to an image.………………..……..……… 64

Figure 45 Median filter applied in matrix form..........…………………….. 64

Figure 46 Median filtering applied to an image... 65

Figure 47 Plots of the Gaussian kernel... 66

Figure 48 Gaussian filtering...................................…………………….. 66

Figure 49 Example of Canny edge detection.. 67

Figure 50 Block diagram of the Sobel edge detection algorithm................ 68

Figure 51 Example of Sobel edge detection......……………....………… 69

FIGURES

Figure 52 Example of Prewitt edge detection......……….........………. 69

Figure 53 Histograms.. 70

Figure 54 Example of a bimodal histogram... 71

Figure 55 Simple thresholds... 71

Figure 56 Peakiness test..........................………………..…..……..…… 72

Figure 57 Pixel connectedness............……………………………….. 73

Figure 58 Block diagram of the seed segmentation algorithm……..…….... 75

Figure 59 Template matching test……………………………….………... 76

Figure 60 Example of squared difference application…………….……..... 77

Figure 61 Example of normalized squared difference application………... 78

Figure 62 Example of cross-correlation application……………………..... 79

Figure 63 Example of normalized cross-correlation application……….…. 80

Figure 64 Example of correlation coefficient application……..……….. 81

Figure 65 Example of NCC application... 82

Figure 66 Example of an integral image.. 83

Figure 67 Summed Area Table................………………..……..…… 83

Figure 68 Example of Hessian matrix application…………………..……… 85

Figure 69 graphical representations of filter side lengths for three different octaves... 86

Figure 70 Non-maximal suppression …………………...……………….. 86

Figure 71 Haar wavelet filters employed to compute the responses in the x and y directions.. 87

Figure 72 Neighborhood of radius 6s around the interest point…..……… 88

Figure 73 Orientation assignment ……………………………………...... 88

Figure 74 Feature descriptor in SURF with 64 dimensions……….…...… 89

Figure 75 Building the descriptor ……..……….…..……….….…...... 89

Figure 76 Descriptor entries of a sub-region representing the nature of the Underlying intensity pattern…………………………………… 89

FIGURES

Figure 77 Project hardware (quadcopter with two cameras) …......……… 90
Figure 78 Block diagram of project software.. 91
Figure 79 Block diagram of single camera calibration.........…..........…...... 92
Figure 80 Calibration pattern (9*6 chess board) ………………………… 93
Figure 81 Block diagram of image rectification and mapping…………. 94
Figure 82 Block diagram of stereo camera calibration ……………........…. 95
Figure 83 Stereo camera calibration……....…………....………….....……. 96
Figure 84 Block diagram of stereo image rectification and mapping…….. 97
Figure 85 Block diagram of image enhancement ……….…………........… 98
Figure 86 Example of image enhancement………........……….………... 98
Figure 87 Block diagram of color segmentation…..……………………… 99
Figure 88 segmentation…………………………………….....………... 99
Figure 89 Block diagram of region segmentation...................................... 100
Figure 90 Region labeling... 100
Figure 91 Block diagram of stereo vision information extraction.............. 101
Figure 92 Extraction and labeling of objects in the left-hand camera image.. 102
Figure 93 Extraction of objects from the right-hand camera image............ 103
Figure 94 Stereo vision and information generation ……....……….......... 104
Figure 95 Block diagram of real-time and database information comparison.. 105
Figure 96 Diagram of position estimation ………..…..………….............… 107
Figure 97 Pose algorithm application ……..…………………..……..…… 107
Figure 98 Comparison of the developed system vs SURF+FLANN…..…. 108
Figure 99 Comparison of sample recognition for the developed system vs SURF+FLANN 109

FIGURES

Figure 100 Comparison of partial-sample recognition for the developed system vs SURF+FLANN.. 110

Figure 101 Comparison of sample description recognition for the devloped system vs SURF+FLANN.. 110

Figure 102 Aerial image of an urban area captured using an infrared camera.. 111

LIST OF TABLES

TABLES

Table 1 Partial Derivatives of Image Coordinates.................................... 34
Table 2 100% RGB Color Bars.. 46
Table 3 YUV Value Bars.. 48
Table 4 YIQ Value Bars... 50
Table 5 HSI Value Bars.. 54
Table 6 Gaussian Kernel When σ=1......……... 65

LIST OF ABBREVIATIONS

GPS	Global Positioning System
VO	Visual Odometry
3D	Three Dimensional System
2D	Two Dimensional System
CAD	Computer-Aided Design
DEM	Digital Earth Model
UAV	Unmanned Aerial Vehicle
NTSC	National Television System Committee
PAL	Phase Alternating Line
SECAM	Sequential Color with Memory

CHAPTER 1

INTRODUCTION

The area of robotics has received great interest from researchers across various fields in recent years. Indeed, the idea of using a robot rather than a human to carry out a specific task is fascinating. Robotics involves a wide variety of technologies, ranging from artificial intelligence algorithms to the physical machines themselves. These components enable the construction of a system whose potentially huge capabilities are greater than those of its basic components. Robotics systems can provide functions previously achievable through the use of traditional machines or through humans working with traditional tools.

Robotics systems can be employed for several different functions, ranging from car assembly to medical surgery [1]. The fundamental principle involved in robot development is the ability to make such systems perform their tasks by themselves, that is, 'autonomously’. Mobile robots are machines that can move autonomously, either in space, on the ground or underwater. Such vehicles are usually unmanned, meaning that no humans are onboard. Mobile machine movement is performed via the use of sensors which sense the environment, with on-board computational resources guiding the robot’s passage.

The main application of such robotic vehicles is their ability to access locations out of reach or which pose a significant risk to any human presence. For example, although human divers may dive to depths of one hundred meters or more, environmental factors such as pressure, light and currents limit further exploration of the vast volume of the Earth’s oceans to mobile machines [2]. Robotic vehicles are also effectively employed for routine functions in environments which are inappropriate for humans, due to factors such as darkness, health risks and noise.

Mobile robot localization is the ability of a robot to understand and sense the properties of its environment in order to reach a specific location more efficiently, reliably and accurately. At the first instance, it may seem a trivial task to navigate a robot when compared to brain surgery or automobile manufacture. However, the latter tasks are so carefully planned and structured that they are essentially high-precision positioning applications for highly specialized tools. In contrast, the problem with robot navigation is associated with a lack of any such high precision, with no databases available as to floor plans and any objects in the environment to be navigated. Furthermore, the environment may be unknown (with obstacles), there may be people moving around, not to mention the presence of deformable objects such as plants and toys. Dealing with such a variable environment poses a plethora of challenges to a mobile system.

1.1 Overview

Position awareness is the result of successful self-localization. Self-localization can thus be defined as answering the question "*where am I?*". Successful localization with high accuracy enables the establishment of the relationship between current position and a framework such as a map, i.e., "*I am in region A on the map*". Position estimation is an important ability for aircraft, supporting navigation and other position-dependent activities. Position estimation is therefore now an essential part of modern life.

A mobile robotic system that navigates huge-scale environments needs to know its location in the real world in order to successfully plan its path and its movements. This requires the establishment of a close relationship between environment and framework. The general approach employed to solve this problem is to provide the system with a detailed description of the environment.

Position estimation systems in almost all aircraft now depend blindly on on-board GPS receivers. However, the GPS signal is often quite weak and can be jammed in cheap and simple ways. Indeed, the United States has identified GPS as a national security threat due to its susceptibility to jamming [3]. Furthermore, GPS service unavailability is frequent due to dense cloud cover, solar flares, and permanent obstructions such as buildings and trees. A more serious threat is represented by hostile action, with the in-flight loss of GPS service potentially leading to the aircraft crashing (especially robotic aircraft such as unmanned aerial vehicles or UAVs). As a result of this GPS fallout problem, the safe use of UAVs in populated areas is still not guaranteed, "except in rare cases such as war zones"[4].

Researchers are currently making a huge effort to solve this problem using many different methods. One possible solution is the development of a localization system based on the use of a visual system. In fact, the common video camera is a highly suitable device with which to attempt to solve the localization problem, considering its ability to sense the surrounding environment. Indeed, virtually all aircraft already employ a multi-camera system as an essential in-flight sensor tool. These cameras are quite light and consume less power, while the images produced contain a huge amount of information. On the other hand, cameras are quite sensitive to light conditions such as sun reflection.

Vision-based localization using an optical sensor system such as a photometric device uses the spatial features found in the visual environment of the camera to determine the system’s location. However, multiple techniques are employed for navigation and positioning through the use of visual information; the main components of each of these techniques are model representations of the environment, location detection algorithms, and sensing models. Further functions include image processing, mapping, motion plan generation, and motion execution. In robot navigation, the task of determining position and orientation via image analysis is known as Visual Odometry (VO), the architecture of which is shown in Figure 1 [5]. The VO system focuses on estimating camera position and displacement, and has a wide range of applications in localization and target tracking.

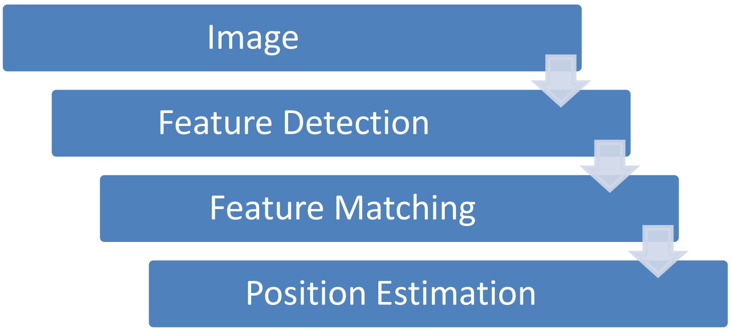


Figure 1 Visual odometry system

1.2 Motivation

Recent years have seen a growing global market for drone aircraft, with demand arising in parallel in both military and civil fields. In the former there is an urgent need to secure borders and to improve reconnaissance and surveillance, as well as to access places unreachable to humans. Military drones are now employed to direct more accurate attacks, as well as to gather information directly from the battlefield. The recent growth in civilian applications is also very noticeable, such as the shopping website Amazon’s aim to deliver goods to consumers via unmanned aircraft [6], and the Government of Dubai delivering parcels to citizens in the same manner [7]. Internet giants Google are also keen to buy factories manufacturing such machines. This indicates that the near future will witness a genuine revolution in the field. The motivation behind the present project was thus to increase the reliability of these aircraft, whose blind reliance on GPS has proven to be a real obstacle to their expanded use inside more densely-populated areas.

1.3 Objective

The main aim of this project was to develop an application for position estimation that can work under different circumstances at both high speed and with high accuracy. Such a system would work to support navigation in unmanned aircraft in situations of a sudden loss of GPS service for any reason, thereby offering these robots greater reliability.

Basic image-based localization frameworks typically exhibit the following limitations:

1- Scale: Elements in different images have different scales.
2- Orientation: Images are rotated with respect to one another.
3- Illumination: Variation in illumination represents a significant problem for accurate image matching.
4- Occlusion: Objects that are separate in the 3D world might interfere with each other in 2D image planes.
5- Matching: The matching of objects in terms of planar, textured or edge.

6- Clutter: Difficulty in distinguishing between the background and object boundaries.

This thesis represents an attempt to solve these limitations using the following methods:

1- Consistency: Position detection should be insensitive to variation in noise.
2- Accuracy: Position detection should be as accurate as possible.
3- Speed: Position identification will be useless if not completed with sufficient speed.

1.4 Organization of the Thesis

This thesis comprises 6 chapters organized as follows:

Chapter 1: The introduction.

Chapter 2: Discusses the literature reviewing the development of image-based positioning systems.

Chapter 3: Focuses on image geometry.

Chapter 4: Focuses on image processing.

Chapter 5: Application.

Chapter 6: Results.

CHAPTER 2

LITERATURE REVIEW FOCUSSING ON THE DEVELOPMENT OF IMAGE-BASED POSITIONING

Computer vision is now a major part of modern navigation systems, with many image-based strategies proposed since the early 1980s based either on images stored in databases or on physical landmarks. The various techniques employed depend on the nature of the environment in which the robot moves, the limits of the known sphere and the type of sensors with which the robot is supplied. Position estimation techniques can be divided into four general types:

A- Landmark-Based methods.

B- Trajectory integration and dead reckoning.

C- Standard reference patterns.

D- A priori world model knowledge, matching sensor data with this world model for position estimation.

2.1 Landmark-Based Methods

A very popular technique, landmark-based position estimation involves the robot determining its own position by finding landmarks in the sphere. As the extent of these landmarks is known, their general position relative to the robot can be measured, and thus the position and direction of the robot can be triangulated from these measurements with reduced uncertainty. This technique can make use of naturally occurring landmarks such as hilltops, roof edges and the tops of buildings, or infrared beacons placed at known positions in the environment. The essential requirement of the landmark-based approach is that the robots should have the ability to locate and identify landmarks, but this is not an easy function. Position estimation methods based on landmarks vary considerably depending on the type of sensors used (vision sensors or distance sensors) and the class of the technique, i.e., the type of features to be identified (such as streaks, corners or point sources). Furthermore, a huge number of landmarks are often needed, the example of which are shown in Figure 2.

A variety of landmark-based techniques have been proposed for position estimation [8]. McGillem and Rappaport describe autonomous vehicle navigation based on an infrared location system. They employ an active approach to position calculation involving three infrared torches to extract the texture of the sphere, as well as a scanning optical system "capable of measuring the angles between a pair of beacons" [9]. Nasr and Bhanu developed a method of perception reasoning based on landmark recognition, together with an expectation paradigm for robot navigation [10].

Sugihara developed an approach with which to estimate the position of a robot equipped with a monocular camera. The location of this unmanned vehicle is established using images captured by the camera, together with a map of the environment in which the vehicle is to navigate. In the images recorded by the unmanned vehicle, with the visual axis corresponding to the surface, only vertical edges are detected. The map provides points where vertical edges are located.

Sugihara then classifies the problem of position estimation into two types. In the first type, all vertical edges found in the camera images are given, with the robot location established by finding a similarity between the vertical edges shown in the images and those on the known map. In the second problem type, vertical edges are distinguishable from one another, but only the order in which they are visible in the image from left to right is supplied. The limitations are assumed chiefly from a computational complexity standpoint [11].

Hui et al. proposed an approach with which to solve the indoor localization problem via the use of barcode landmarks. These intelligent landmarks are marked with a red color. When the robot obtains an image recorded by a color camera in RGB, it first changes the color system to the HSV system. After searching for the red image, the robot then calculates its position relative to the landmark based on the formula I=(x,y,¥), where x,y are the position co-ordinates and ¥ is the rotation [12]. Similarly, Jang presented a color detection approach based on the chromaticity of special landmarks to solve the indoor position estimation problem [13].

Many of the landmark-based methods considered above suffer from the following disadvantages:

1- Assume the provision of data regarding landmarks located inside the machine sphere domain.
2- Require an elementary position in order to start landmark testing.
3- Depend on the robot's vision and ability to identify landmarks from frames and to identify shape properties such as attitude with respect to the robot's current position.
4- Require a database of landmarks found on the flight path which are then searched for in the frames.

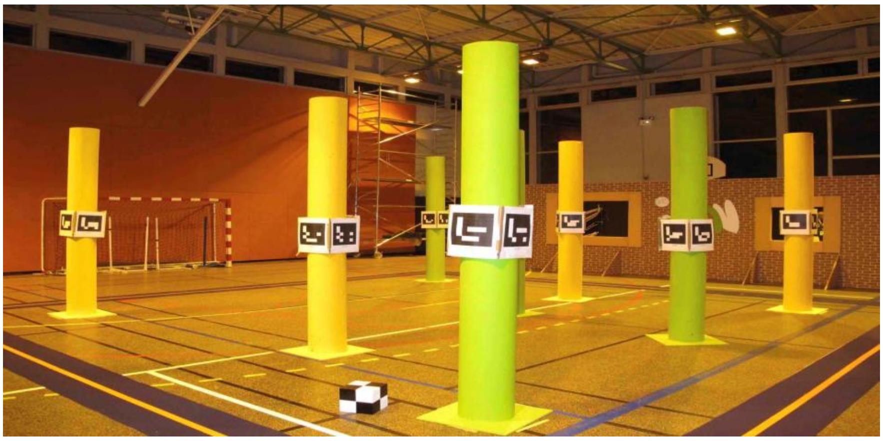


Figure 2 Example of landmark

2.2 Trajectory Integration and Dead Reckoning

Another type of method employed to estimate robot localization and pose is trajectory integration and dead reckoning. In this technique the unmanned vehicle constantly stores information regarding its present location and pose; as it changes its position it modifies the location by dead reckoning, using the camera to sense its environment relative to the new location. Features are revealed by the camera when monitoring specific positions of the unmanned vehicle, with data then employed to build a 3D model of the world. When the unmanned vehicle changes its position, new features are detected and reiterated to correspond with the old features. A certain degree of unmanned vehicle activity is also recognized by its positioning systems. The on-board positioning system of an unmanned vehicle uses the information obtained from the estimation of these activities to predict the position of new features in the 3D model of the world. This process is used to help the system reduce the search space and to build up a relationship between the features and the 3D model of the world in the view. The parameters of unmanned vehicle motion between the new and old locations are obtained precisely because the relationships were established by cameras. This solution enables the position of the unmanned vehicle to be obtained with a high degree of accuracy. This process proceeds as a loop and thus the 3D world model is updated continuously. Figure 3 displays example of this type of approach.

Moravec was the first researcher to develop an autonomous system for unmanned vehicle navigation based on the use of binocular vision. He also defined properties with which to reduce features in the obtained frame from coarse to fine, with a correlation strategy used to establish a correspondence between features in different frames selected by the operator [14]. Matthies and Shafer discussed the employment of the 3-D Gaussian distribution, finding this approach to be more appropriate for use with binocular vision to reduce triangulation errors associated with limited sensor resolution [15]. In their approach they used a method to determine the 3-D Gaussian error distribution parameters of mean and covariance for binocular frames. They also developed a technique with which to constantly update unmanned vehicle location, taking into consideration the Gaussian error distribution of the features, as well as moving parameters and their error co-variance. Kalman filtering was employed in an efficient frequent process to find the current state of the dynamic system from a concatenation of testing and noisy measurements [16]. Kalman filtering is a probabilistic approach typically used for forward inference in linear systems with Gaussian noise.

Faugeras and Ayache also tried to solve the limitations associated with the autonomous navigation of an unmanned vehicle, specifically employing a more mathematical and strict approach. They used three vision sensors to build a stereo system, with target features then found using a 3-D model of the environment to detect line segments. This model was created by combining all optical information obtained at many different locations. The final result of this approach was the creation of a number of 3D line segments, which were linked to the coordinate frames and related by accurate motion [17]. Errors in the 3D model and line segment measurements were tackled via the use of a Kalman filtering technique. Similar to that employed by Faugeras and Ayache, Crowley developed a line segment-based technique to create a model of the environment, with Kalman filtering again used to reduce variance error. However, whereas the former authors used a vision sensor to obtain their data, the latter used a circular ring of 24 ultra-sound sensors [18]. Chatila and Laumond created a model of the environment and a position referencing system via the use of multi-laser range finder sensors to measure depth, and optical shaft encoders on the drive wheel axis to obtain trajectory integration for their unmanned vehicle "HILARE" [19].

In the above techniques, the position estimation strategy depends on the representation used for the sensory observations and the environment. Miller suggested a method with which to solve the indoor position estimation problem for an unmanned vehicle involving the construction of a model for the indoor surface of the real world. Ranging devices such as sonar and laser range finders enabled the system to move around obstacles and walls [20]. However, this approach does not work in open space because wall and obstacle representation requires a huge amount of information for the unmanned vehicle due to its projection on the floor, and thus the latter’s motion is limited. The general key to this approach is the fact that it is easy to make a map of the environment by linking regions with frame coordinates. Features represented by obstacles and walls are employed to build line segments, with end point locations set for the region using the frame coordinate system. Region edges are identified with labels in order to allow the system to refer to adjoining regions. All regions are divided into four types (0--F, l--F, 2—F, and 3--F), with the environment floor for the unmanned vehicle including 3 degrees, two representing translation (x,y) and the other orientation. A further type "JF" is employed to reduce j degrees via the use of sonar to obtain range information. As well as the map model, Miller also presented an approach for location estimation involving a guide search paradigm. Because of the limited space of this type of environment, the unmanned vehicle can determine the size of positional information in an estimated fashion. If the unmanned vehicle is in the first type of region (0--F), the only information available would be in the form of extrapolations from its last known position, obtained using the unmanned vehicle’s ability to perform dead reckoning. When the unmanned vehicle must move to region 1--F or higher, positional information can be detected by reading the state of several sensors and by applying a heuristic search algorithm over the table of matches between walls and map borders. This approach requires the use of one-time matching between features and the map border, with a simple geometric calculation then applied to process the location and orientation of the unmanned vehicle.

Elfes employed a grid-based representation technique known as "Certainty Grids" to build a workspace map for an unmanned vehicle. This approach involved creating a rectangular grid model of the environment floor, with each grid containing spatial information recognized by the unmanned vehicle via the use of laser sonar [21]. The author additionally developed a fast algorithm for two maps of the aforementioned area to determine vehicle displacement, angle, and the advantage of the match. This data is then used to identify the position and administration of the adaptable unmanned vehicle.

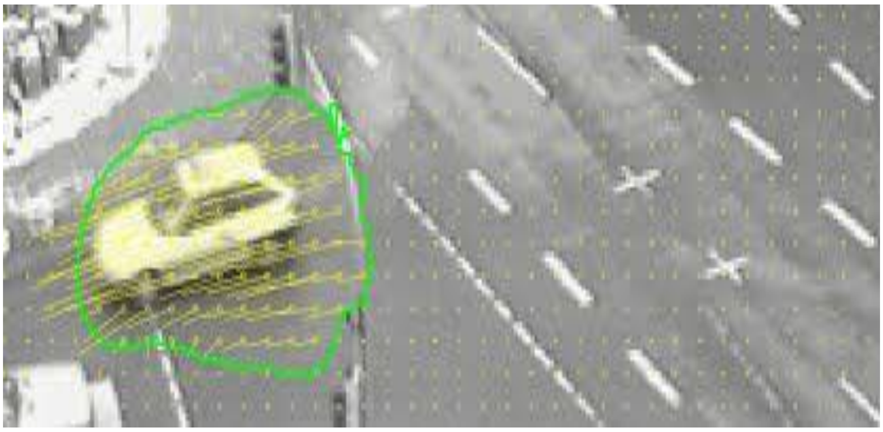


Figure 3 Example of trajectory integration and dead reckoning

2.3 Standard Pattern

Another approach aimed at the accurate estimation of an unmanned vehicle’s movement direction and position involves the placement of accepted or standard patterns in the environment. Figure 4 displays an example in the form of lamps placed on an airport runway. Once the unmanned vehicle identifies these patterns, its location can be estimated based on the accepted area of the model and its geometry. Such models are designed to produce a huge amount of geometric information when the pattern is subject to the perspective projection of the vision sensors. Due to factors such as noise, ambiguous objects may be found. The designed approach will avoid ambiguous interpretations, although a minimum level of camera a priori knowledge is desirable. These methods are decidedly advantageous for those applications which require a degree of accuracy with respect to unmanned vehicle position location. Simple path recognition systems can be employed to locate the unmanned vehicle by recognizing the mark (i.e., the standard pattern). Researchers have employed a variety of signs, patterns and geometry techniques for location estimation [22, 23].

Figure 4 Example of standard pattern

2.4 A Priori World Model Knowledge

When sufficient data is available regarding the environment in which the unmanned vehicle is to navigate, researchers have employed unique approaches to solve the location problem. In such scenarios the environment can be divided into two types: indoor and outdoor. In Figure 5, information for a DEM. A variety of location estimation techniques can be employed. Essentially, the unmanned vehicle senses the environment by using its on-board vision system, detecting the correspondence between what it sees and the provided model of the environment (CAD or DEM). Navigation in the world is then performed by estimating pose and location based on this correspondence. One problem with this type of method is that sensor readings and environment model data can be in various formats. For example, for an unmanned vehicle provided with a CAD model of an apartment and an RGB on-board camera, as the CAD model uses three-dimensional data and the camera two-dimensional images, the two types of information are difficult to match. This type of problem has attracted the attention of many researchers, including Kak et al. [24]. In their work the latter authors present the "PSEIKI" system, which uses logic in a hierarchical framework to interpret images obtained by vision sensors. The authors also discuss how "PSEIKI" can be employed for the self-location of an unmanned vehicle, with PSEIKI output data used in combination with a navigational system in the unmanned vehicle “PETER” [24]. Vehicle position-encoders store estimates of its location and heading at each point. Errors are derived from many sources such as wheel slippage. However, the encoders, vision sensors and CAD model of the environment together generally provide an accurate estimation of vehicle location and direction. The general idea behind using encoders which generate CAD model data is to approximate vehicle location and to estimate that actually obtained by the on-board vision system. Matches are then built between features in the two images (expected and actual), and thus the position of the unmanned vehicle can be estimated with reduced uncertainty.

Other researchers have employed a similar approach in developing a position estimation system. For example, Tsubouchi and Yuta constructed a navigation system for their unmanned vehicle "YAMABICO" by using a color camera as a vision sensor and a map for the working environment, taking into account real-time requirements [25]. Their presented system includes three functions, the first of which involves the generation of images by the on-board color camera. These images are then processed to extract perspective information based on the map model and coordinate transformation. The final function is to create a correspondence between the perspective data and the map model. A color camera was considered essential, as color images are invariant under shadow or light.

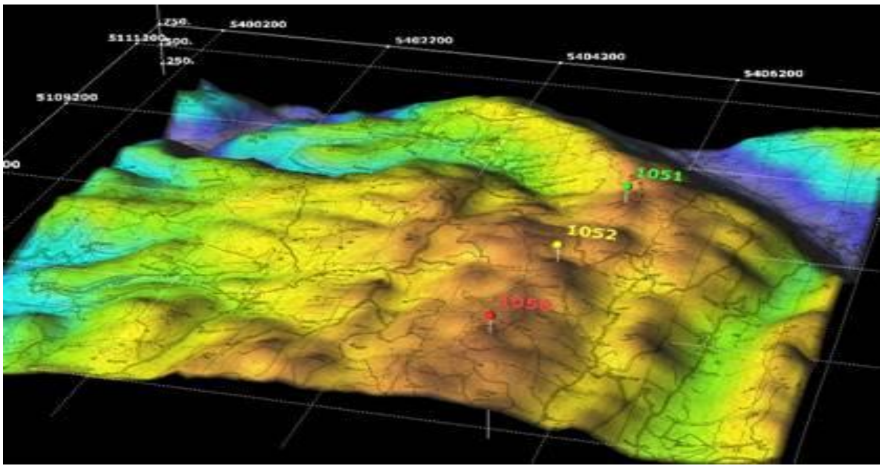


Figure 5 Example of a priori world model knowledge

Chapter 3

IMAGE GEOMETRY

Many types of imaging devices are available, from simple telescopes through to more complex devices such as video cameras, radio telescopes, and even the human eye. Invented in the sixteenth century, the first camera or "camera obscura" contained no lens, instead employing a pinhole to let in light and focus the rays onto the camera wall. In contrast, the modern cameras which replaced the pinhole model are equipped with (sometimes many) sophisticated lenses.

Image geometry can be defined as the spatial relationship between objects in the real world and objects in an image plane. The main application of image geometry is to correct distortion in digital images taken by a camera. Images taken by cheap cameras are often of low quality and suffer from significant distortion effects. These images must be corrected in order to be used in any survey. The world we live in is three dimensional, which means that any point in the "real world" space can be specified by three coordinates (X, Y, and Z). In contrast, an image is a two-dimensional plane, with points on this image accordingly represented by a two-dimensional coordinate system (X and Y). One of the main aims of computer vision is to recover the lost 'Z' axis. The present chapter focuses on the camera matrix and camera calibration, and how these are involved in recovering the 'Z' axis.

3.1 Transformation

Transformations are used to solve many problems associated with computer vision. These transformations are also used in computer graphics. The following section discusses the transformations of translation, scaling and rotation.

3.1.1 Translation

Let us consider the point of an object in the real world ' ***P*** ' with the coordinates ($X1$, $Y1$, $Z1$). Assume point ' ***P*** ' is subject to translation by [DX DY DZ] respectively in the 3D space. As a result of this process, a new point ' ***P´*** ' is produced with three new coordinates ($X2$, $Y2$, $Z2$) in the 3D space, as given by the following equations:

$$X2 = X1 + DX \tag{3.1}$$

$$Y2 = Y1 + DY \tag{3.2}$$

$$Z2 = Z1 + DZ \tag{3.3}$$

These equations can also be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & DX \\ 0 & 1 & 0 & DY \\ 0 & 0 & 1 & DZ \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \mathrm{T} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.5}$$

$$\text{where } \mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & DX \\ 0 & 1 & 0 & DY \\ 0 & 0 & 1 & DZ \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

'T' is called the "Translation matrix", for which the inverse translation matrix is given by:

$$\mathrm{T}' = \begin{bmatrix} 1 & 0 & 0 & -DX \\ 0 & 1 & 0 & -DY \\ 0 & 0 & 1 & -DZ \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can verify that TT´ = T´T = I, where I is the identity matrix.

3.1.2 Scaling

Let us consider the point of an object in the real world ' ***P*** ' with coordinates ($X1$, $Y1$, $Z1$). Assume that point ' ***P*** ' is scaled by [SX SY SZ] respectively in the 3D space.

As a result of this process, a new point ' ***P*´** ' is produced with new coordinates (*X*2, *Y*2, *Z*2) in the 3D space, as given by the following equations:

$$X2 = X1 * SX \tag{3.6}$$

$$Y2 = Y1 * SY \tag{3.7}$$

$$Z2 = Z1 * SZ \tag{3.8}$$

These equations can also be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \mathrm{S} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.10}$$

$$\text{Where S} = \begin{bmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

'S' is known as the "scaling matrix", the inverse of which is given by:

$$\mathrm{S}' = \begin{bmatrix} 1/SX & 0 & 0 & 0 \\ 0 & 1/SY & 0 & 0 \\ 0 & 0 & 1/SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.1.3 Rotation

Let us consider the point of an object in the real world ' ***P*** ' with coordinates (*X*1, *Y*1, *Z*1). Denote the line between the origin point ' ***O*** ' (0, 0, 0) and point ' ***P*** ' as ' ***R*** ', and assume the angle between ' ***R*** ' and the *X* axis is ***θ***. This process will produce a triangle from which we can obtain *X*1 and *Y*1 using the following expressions:

$$\mathrm{X1} = \mathrm{R} * \cos\theta \tag{3.11}$$

$$Y1 = R * sin\theta \tag{3.12}$$

Now consider rotating point ' ***P*** ' about the Z axis by angle *ø*. This now means that the angle between ' ***R*** ' and the *X* axis is (θ + *ø*), and we can write the formulae for *X2* and *Y2* as follows:

$$\mathrm{X2} = \mathrm{R} * \cos(\text{ø} + \theta) \tag{3.13}$$

$$Y2 = R * \sin(\text{ø} + \theta) \tag{3.14}$$

We can also simplify these formulae to the following:

$$X2 = R * (cos\,\text{ø} * cos\theta) - R * (sin\text{ø} * sin\theta) \tag{3.15}$$

$$Y2 = R * (sin\,\text{ø} * sin\theta) + R * (cos\text{ø} * cos\theta) \tag{3.16}$$

From equations 3.13 and 3.14 we can then write the formulae as follows:

$$X2 = X1 * cos\theta - Y1 * sin\theta \tag{3.17}$$

$$Y2 = X1 * sin\theta + Y1 * cos\theta \tag{3.18}$$

The above equations can be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \mathrm{A} * \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{3.20}$$

Where 'A' is:

$$\text{Either} \quad \mathrm{R}(\theta, \mathrm{Z}) = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Or} \quad \mathrm{R}(\theta, \mathrm{X}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Or} \qquad R(\theta, Y) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2 Perspective Transformation

Perspective transformation involves transforming a point from the real world 3D coordinate system to the 2D image coordinate system. The distance between the lens and the image plane is known as the focal length, here denoted ' f ' and the origin point as ' O ' in Figure 6.

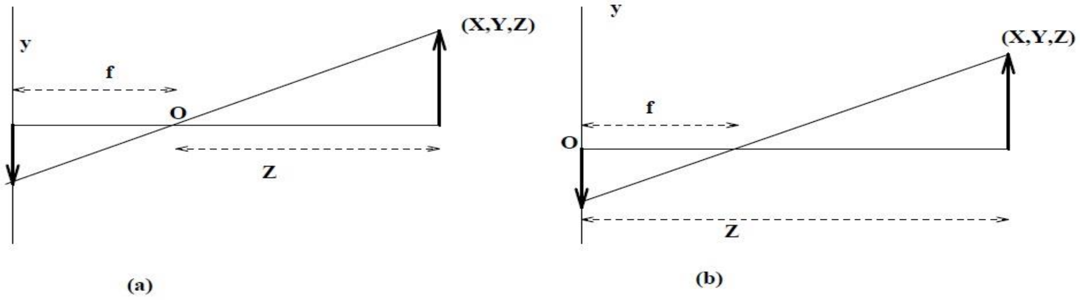


Figure 6 Pinhole camera model: a) origin at the lens, b) origin at the image plane.

Using two equivalent triangles we can write the following:

$$\frac{-y}{Y} = \frac{F}{Z} \tag{3.21}$$

The point in the image plane is actually upside down and for this reason 'y' is negative.

Thus we can obtain 'x' and 'y' via the following equations:

$$\text{x} = -\frac{F * X}{Z} \tag{3.22}$$

$$\text{y} = -\frac{F * Y}{Z} \tag{3.23}$$

The above equations represent the perspective transform with the origin ' O ' at the lens. If the origin ' O ' is moved to the image, the perspective transform is defined by the following equations:

$$\mathrm{x} = \frac{F * X}{F - Z} \tag{3.24}$$

$$\mathrm{y} = \frac{F * Y}{F - Z} \tag{3.25}$$

Due to the nature of the last two equations it is impossible to derive a perspective matrix, and thus another type of transformation known as "homogeneous transformation" is required to convert the Cartesian coordinate system data (X, Y, Z) into that for a homogeneous coordinate system (CX, CY, CZ, C). Transformation of each axis is carried out by multiplying a constant 'C'; the reverse transformation from the homogeneous coordinate system to the Cartesian coordinate system thus involves dividing all axis vectors (CX, CY, CZ) by 'C'.

From the above we can define the perspective matrix as follows:

$$\mathrm{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

and the inverse perspective matrix as:

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}$$

The perspective transform which relates the homogeneous world coordinates to the homogeneous image coordinates is defined thus:

$$\begin{bmatrix} CX \\ CY \\ CZ \\ -\frac{CZ}{f} + C \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} * \begin{bmatrix} CX \\ CY \\ CZ \\ C \end{bmatrix}$$

3.3 Camera Model

From the above section we know that the perspective transform relates the world coordinates with the image coordinates when the camera is located at the origin of the world coordinates. For this reason, in real life we need to translate and rotate the camera to bring the object of interest to the field of view. This means that the camera model contains several transformations besides the perspective transform.

Assuming that the camera is at the origin of the real world coordinates, we first need to translate by the G matrix and then rotate by θ counterclockwise about the Z axis (matrix R (z,-θ)). A further rotation of ø counterclockwise about the X axis (matrix R(x,-ø)) is then required, followed by translation by (r1 r2 r3) (matrix of C).

From the above we can relate the homogeneous coordinates of the real world (Wh) with the homogeneous coordinates of the camera image (Ch) as follows:

$$Ch = P * C * R(-ø, X) * R(\theta, Z) * G * Wh \tag{3.26}$$

where

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & -r1 \\ 0 & 1 & 0 & -r2 \\ 0 & 0 & 1 & -r3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & -X1 \\ 0 & 1 & 0 & -Y1 \\ 0 & 0 & 1 & -Z1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R\ (-\theta, Z) \ = \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R\ (-ø, X) \ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cosø & sinø & 0 \\ 0 & -sinø & cosø & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now we can calculate the location of the point in the image plane using the Cartesian coordinate system as follows:

$$\mathrm{x} = F\frac{(X - X0) * cos\theta + (Y - Y0) * sin\theta - r1}{-(X - X0) * sin\theta * sin\phi + (Y - Y0) * cos\theta * sin\phi - (Z - Z0) * cos\phi + r3 + F} \tag{3.27}$$

$$\mathrm{y} = F\frac{-(X - X0) * sin\theta * cos\phi + (Y - Y0) * cos\theta * cos\phi + (Z - Z0) * sin\phi - r2}{-(X - X0) * sin\theta * sin\phi + (Y - Y0) * cos\theta * sin\phi - (Z - Z0) * cos\phi + r3 + F} \tag{3.28}$$

3.4 Lens Distortion

In reality, cameras, especially those used for filmmaking, are much more complicated than simple pinhole cameras because they employ multi-component lenses in front of their apertures that gather and focus incoming light. As a result, the incoming light from the point of an object in the real three-dimensional world to the image plane will likely deviate slightly due to lens distortion. Lens distortion can be divided into three types as follows:

1- **Barrel distortion**: The camera lens causes image magnification, with the straight lines in the image bulging outwards as shown in Figure 7. This effect is significant in zoom lenses, where distortion increases when moving away from the center "principle point" and reaches a peak in the image borders. However, this effect can be exploited in a positive way by using "fisheye lenses" in the plane to map specific objects in the environment.

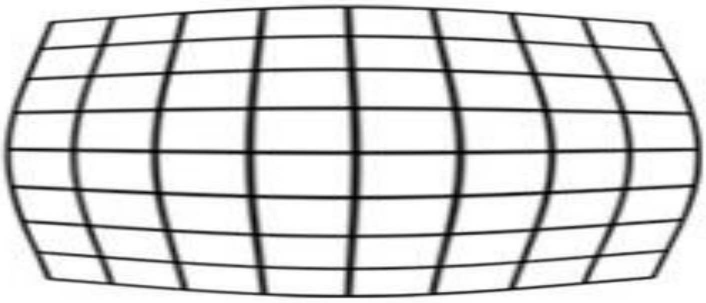


Figure 7 Barrel distortion

2- **Pincushion distortion**: Here the camera lens causes image magnification in which the image is bowed inwards, as shown in Figure 8. This effect also occurs in zoom lenses, with distortion increasing when moving to the center "principle point" and the greatest damage observed at the image border.

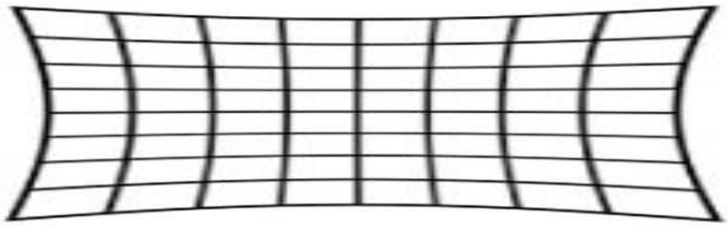


Figure 8 Pincushion distortion

3- **Mustache distortion**: A combination of barrel and pincushion distortion, mustache distortion is the least common of the three distortion types. Barrel-like distortion appears near the center and lessens gradually with distance from the principle point, while pincushion distortion is observed near the border and decreases gradually with distance, as shown in Figure 9.

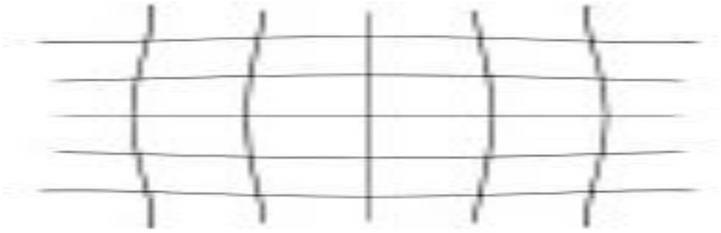


Figure 9 Mustache distortion

We can correct lens distortion by using the following equation:

$$\begin{bmatrix} xdist \\ ydist \end{bmatrix} = (1 + K1\,(x^{-2} + y^{-2}) + K2\,(x^{-2} + y^{-2})^2) * \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.29}$$

where 'k'1 and 'K2' are coefficients that control the amount of distortion. From the above equation we can rewrite a full equation as follows:

$$\begin{bmatrix} xdist \\ ydist \end{bmatrix} = \begin{bmatrix} \mathrm{x'dist/dx} \\ y'dist/dy \end{bmatrix} + \begin{bmatrix} x0 \\ y0 \end{bmatrix} \tag{3.30}$$

$$= \begin{bmatrix} \mathrm{x0} \\ y0 \end{bmatrix} + (1 + K1\,(x^{-2} + y^{-2}) + K2\,(x^{-2} + y^{-2})^2) * \begin{bmatrix} x0 \\ y0 \end{bmatrix}$$

Now we will outline a simple scenario for the estimation of the lens distortion coefficients. Knowledge of the camera’s internal parameters enables the acquisition of the ideal points (x, y) which correspond to the distortion points (xdist, ydist). Each point generates two equations from which we obtain the K coefficients.

$$\begin{bmatrix} (x - x0)(x^{-2} + y^{-2}) & (x - x0)(x^{-2} + y^{-2})^2 \\ (y - y0)(x^{-2} + y^{-2}) & (y - y0)(x^{-2} + y^{-2}) \end{bmatrix} * \begin{bmatrix} K1 \\ K2 \end{bmatrix} = \begin{bmatrix} xdist - x \\ ydist - y \end{bmatrix} \tag{3.31}$$

3.5 Camera Calibration

In the previous section we discussed the camera model which relates the world coordinates with the image coordinates. It was assumed that camera focal length, angles, and translation displacements are known. Camera calibration is carried out in order to determine the camera model parameters which enable the camera to be used as a measuring device. Camera calibration can thus be defined as the process of extracting all camera parameters, including focal length, principle point, lens distortion, and pixel size.

Calibration is an essential component of the three-dimensional vision system and is necessary to obtain the hardware information from a two-dimensional image. A variety of calibration methods have been employed in the literature, two of which are described here:

1- Photogrammetric calibration.
2- Plane-based internal parameter estimation.

Photogrammetric calibration is performed by observing a calibration object whose geometry in the real world is known with very good precision. In this scenario height calibration can also be carried out efficiently. This type of calibration typically uses an object consisting of two or three orthogonal planes, as shown in Figure 10. However, the method also requires expensive apparatus to calculate the geometry of objects in the real world such as laser scanners and ultrasound range finders.

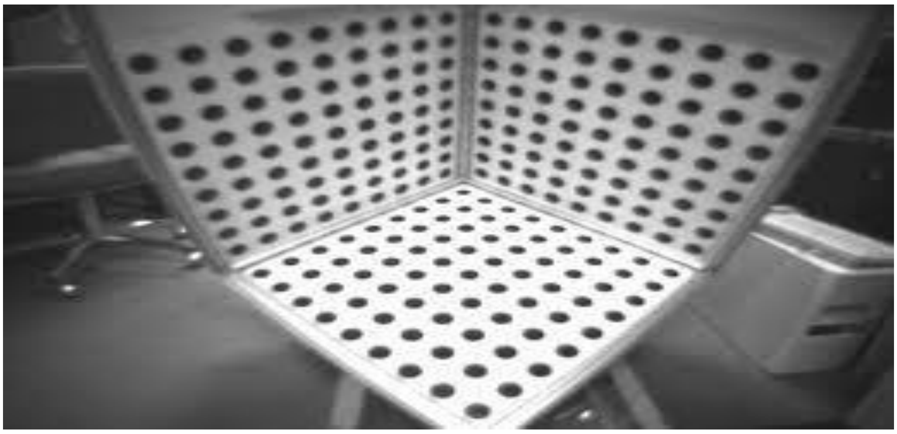


Figure 10 Photogrammetric calibration object

As shown in the camera model, the relationship between the homogeneous coordinate system of a point in the real world and the homogeneous coordinate system of the point in the image is obtained via equation 3.26. Here let us assume the following equation:

$$A = P * C * R(-\o, X) * R(\theta, Z) * G \tag{3.32}$$

This means we can rewrite equation 3.26 using equation 3.32 as follows:

$$Ch = A * Wh \tag{3.33}$$

'A' is a 4*4 matrix expressed by the following:

$$\text{A}=\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix}$$

Equation 3.26 can be written in matrix form as follows:

$$\begin{bmatrix} Ch1 \\ Ch2 \\ Ch3 \\ Ch4 \end{bmatrix}=\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix}*\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.34}$$

The aim of camera calibration is to determine the 'A' matrix using the known 3D points for which there are corresponding image coordinates. From the above, 'A' is a 4*4 matrix with 16 unknown elements. In order to obtain the 'A' matrix we must try to simplify as much as possible. 'Ch3' is not meaningful in the image coordinate system because the 'Z' axis is absent. As a result the third row in the 'A' matrix can be removed, with the A matrix then rewritten as follows:

$$\text{A}=\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a41 & a42 & a43 & a44 \end{bmatrix}$$

Equation 3.34 can then be rewritten:

$$\begin{bmatrix} Ch1 \\ Ch2 \\ Ch4 \end{bmatrix}=\begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a41 & a42 & a43 & a44 \end{bmatrix}*\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.35}$$

Now we have an 'A' matrix with 12 unknown elements. We can find [Ch1, Ch2 and Ch4] from the previous equation via the following expressions:

$$Ch1 = X * a11 + Y * a12 + Z * a13 + a14 \tag{3.36}$$

$$Ch2 = X * a21 + Y * a22 + Z * a23 + a24 \tag{3.37}$$

$$Ch4 = X * a41 + Y * a42 + Z * a43 + a44 \tag{3.38}$$

To convert data from the homogeneous coordinate system to the Cartesian coordinate system we must divide all elements by the last element. In this case we obtain 'x' and 'y' by respectively dividing 'Ch1' and 'Ch2' by 'Ch4' as follows:

$$\mathrm{x} = \frac{Ch1}{Ch4} \tag{3.39}$$

$$\mathrm{y} = \frac{Ch2}{Ch4} \tag{3.40}$$

We can simplify these equations to:

$$x * Ch4 = Ch1 \tag{3.41}$$

$$y * Ch4 = Ch2 \tag{3.42}$$

and then to:

$$x * Ch4 - Ch1 = 0 \tag{3.43}$$

$$y * Ch4 - Ch2 = 0 \tag{3.44}$$

Equations 3.43 and 3.44 can then be solved:

$$T = x * (X * a41 + Y * a42 + Z * 43 + a44\ )$$

$$G\ = X * a11 + Y * a22 + Z * 13 + a14$$

$$T - G\ = 0 \tag{3.45}$$

$$J = y * (X * a41 + Y * a42 + Z * 43 + a44\ )$$

$$B\ = X * a21 + Y * a22 + Z * 23 + a24$$

$$J - B = 0 \tag{3.46}$$

We now have five known points X, Y and Z of the object in the real world for which x and y are the corresponding image coordinates, as well as 12 unknown elements a11,...,a44. This means that if we collect n points from observing the object we will obtain 2*n equations, which can be used to solve the 12 unknowns.

$$\begin{bmatrix}
X1*a11+Y1*a12+Z1*a13+a14-x1*X1*a41-x1*Y1*a42-x1*Z1*a43-x1*a44=0\\
X1*a21+Y1*a22+Z1*a23+a24-y1*X1*a41-y1*Y1*a42-y1*Z1*a43-y1*a44=0\\
X2*a11+Y2*a12+Z2*a13+a14-x2*X2*a41-x2*Y2*a42-x2*Z2*a43-x2*a44=0\\
X2*a21+Y2*a22+Z2*a23+a24-y2*X2*a41-y2*Y2*a42-y2*Z2*a43-y2*a44=0\\
.\\
.\\
.\\
Xn*a11+Yn*a12+Zn*a13+a14-xn*Xn*a41-xn*Yn*a42-xn*Zn*a43-xn*a44=0\\
Xn*a21+Yn*a22+Zn*a23+a24-yn*Xn*a41-yn*Yn*a42-yn*Zn*a43-yn*a44=0
\end{bmatrix}$$

The previous matrix can be rewritten as follows:

$$\begin{bmatrix}
X1 & Y1 & Z1 & 1 & 0 & 0 & 0 & 0 & -x1*X1 & -x1*Y1 & -x1*Z1 & -x1\\
X2 & Y2 & Z2 & 1 & 0 & 0 & 0 & 0 & -x2*X2 & -x2*Y2 & -x2*Z2 & -x2\\
. & . & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & . & .\\
Xn & Yn & Zn & 1 & 0 & 0 & 0 & 0 & -xn*Xn & -xn*Yn & -xn*Zn & -xn\\
0 & 0 & 0 & 0 & X1 & Y1 & Z1 & 1 & -y1*X1 & -y1*Y1 & -y1*Z1 & -y1\\
0 & 0 & 0 & 0 & X2 & Y2 & Z2 & 1 & -y2*X2 & -y2*Y2 & -y2*Z2 & -y2\\
. & . & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & . & .\\
0 & 0 & 0 & 0 & Xn & Yn & Zn & 1 & -yn*Xn & -yn*Yn & -yn*Zn & -yn
\end{bmatrix}
*
\begin{bmatrix} a11\\ a12\\ a13\\ a14\\ a21\\ a22\\ a23\\ a24\\ a41\\ a42\\ a43\\ a44 \end{bmatrix}
=
\begin{bmatrix} 0\\ 0\\ .\\ .\\ .\\ 0\\ 0\\ 0\\ .\\ .\\ 0 \end{bmatrix}
\qquad (3.47)$$

In this homogeneous system, which has multiple solutions, we can assume that a44=1. We can therefore rewrite equation 3.48 as either:

$$\begin{bmatrix}
X1 & Y1 & Z1 & 1 & 0 & 0 & 0 & 0 & -x1*X1 & -x1*Y1 & -x1*Z1\\
X2 & Y2 & Z2 & 1 & 0 & 0 & 0 & 0 & -x2*X2 & -x2*Y2 & -x2*Z2\\
. & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & .\\
Xn & Yn & Zn & 1 & 0 & 0 & 0 & 0 & -xn*Xn & -xn*Yn & -xn*Zn\\
0 & 0 & 0 & 0 & X1 & Y1 & Z1 & 1 & -y1*X1 & -y1*Y1 & -y1*Z1\\
0 & 0 & 0 & 0 & X2 & Y2 & Z2 & 1 & -y2*X2 & -y2*Y2 & -y2*Z2\\
. & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & .\\
. & . & . & . & . & . & . & . & . & . & .\\
0 & 0 & 0 & 0 & Xn & Yn & Zn & 1 & -yn*Xn & -yn*Yn & -yn*Zn
\end{bmatrix}
*
\begin{bmatrix} a11\\ a12\\ a13\\ a14\\ a21\\ a22\\ a23\\ a24\\ a41\\ a42\\ a43 \end{bmatrix}
=
\begin{bmatrix} x1\\ x2\\ .\\ .\\ .\\ xn\\ y1\\ y2\\ .\\ .\\ yn \end{bmatrix}
\qquad (3.48)$$

$$D*F=R \qquad (3.49)$$

$$\text{When D} = \begin{bmatrix} X1 & Y1 & Z1 & 1 & 0 & 0 & 0 & 0 & -x1*X1 & -x1*Y1 & -x1*Z1 \\ X2 & Y2 & Z2 & 1 & 0 & 0 & 0 & 0 & -x2*X2 & -x2*Y2 & -x2*Z2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Xn & Yn & Zn & 1 & 0 & 0 & 0 & 0 & -xn*Xn & -xn*Yn & -xn*Zn \\ 0 & 0 & 0 & 0 & X1 & Y1 & Z1 & 1 & -y1*X1 & -y1*Y1 & -y1*Z1 \\ 0 & 0 & 0 & 0 & X2 & Y2 & Z2 & 1 & -y2*X2 & -y2*Y2 & -y2*Z2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & Xn & Yn & Zn & 1 & -yn*Xn & -yn*Yn & -yn*Zn \end{bmatrix}$$

$$\mathrm{F} = [a11\ a12\ a13\ a14\ a21\ a22\ a23\ a24\ a41\ a42\ a43]^T$$

$$\mathrm{R} = [x1\ x2\ \ldots xn\ y1\ y2\ \ldots yn]^T$$

In the above system we know the values of D and thus it is easy to determine the values of F using matrix algebra as follows:

$$D^T * \mathrm{D} * \mathrm{F} = D^T\ * R \tag{3.50}$$

$$\mathrm{F} = (D^T D)^{-1}\ * D^T R \tag{3.51}$$

A second method of camera calibration, known as *Plane-based internal parameter estimation*, is more commonly employed to obtain internal camera parameters and requires no precise environmental measurements. As an example we will use several images of a planar surface with different orientations. A checkerboard of black and white squares is employed as a calibration pattern by being moved in front of a fixed camera in several directions.

As outlined above, equations 3.21 and 3.22 are used to identify the points of an object in the real world environment on the image plane. We can thus identify the points on the image plane in the image array via the following equations:

$$\mathrm{x}' = \frac{x}{\alpha x} + x0 \tag{3.52}$$

$$\mathrm{y}' = \frac{y}{\alpha y} + y0 \tag{3.53}$$

$$\alpha x = \frac{f}{dx} \tag{3.54}$$

$$\alpha y = \frac{f}{dy} \tag{3.55}$$

where 'dx' is the pixel width and 'dy' is the pixel height in physical units, and point (x0, y0) is the location of the origin.

Equations 3.52 and 3.53 can be rewritten in matrix form as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A * \begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix} \tag{3.56}$$

$$A = \begin{bmatrix} \alpha x & 0 & x0 \\ 0 & \alpha y & y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Where

$$\begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{3.57}$$

$$R = r1 * r2 * r3 \tag{3.58}$$

and 't' is the translation vector.

Let us denote the camera matrix as 'P'.

$$P = A[R|t] \tag{3.59}$$

This produces the following final equation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = P * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.60}$$

The previous equation is then used to translate points from the 3D real world coordinate system to the 2D image coordinate system.

Assuming that the Z axis is equal to zero, this produces:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = P * \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A[r1\ r2\ r3\ |t] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$r3 = 0$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A[r1\ r2\ t] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.61}$$

$$H = A[r1\ r2\ t] \tag{3.62}$$

'H' is a 3*3 matrix, expressed as follows:

$$H = \begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & h33 \end{bmatrix}$$

We can also rewrite this matrix:

$$H = \begin{bmatrix} | & | & | \\ h1 & h2 & h3 \\ | & | & | \end{bmatrix}$$

This means that:

$$H = [h1\ h2\ h3] \tag{3.63}$$

$$A^{-1}H = [\mathrm{r1\ r2\ t}] \tag{3.64}$$

$$[h1\ h2\ h3]A^{-1} = [r1\ r2\ t]$$

$$h1^T\ (A^{-T}A^{-1})\ h2 = 0 \tag{3.65}$$

$$h1^T\ (A^{-T}A^{-1})\ h1 = h2^T\ (A^{-T}A^{-1})\ h2 \tag{3.66}$$

We can then denote $(A^{-T}A^{-1})$ as B, where B is a 3*3 matrix written as follows:

$$B = (A^{-T}A^{-1}) = \begin{bmatrix} B11 & B12 & B13 \\ B21 & B22 & B23 \\ B31 & B32 & B33 \end{bmatrix}$$

Since we know the form of the camera calibration matrix in equation 3.61, we can verify that:

$$\mathrm{B} = \begin{bmatrix} \frac{1}{\alpha x^2} & 0 & -\frac{x0}{\alpha x^2} \\ 0 & \frac{1}{\alpha y^2} & -\frac{y0}{\alpha y^2} \\ -\frac{x0}{\alpha x^2} & -\frac{y0}{\alpha y^2} & \frac{x0^2}{\alpha x^2} + \frac{y0^2}{\alpha y^2} + 1 \end{bmatrix}$$

Note that B is symmetric and defined by a 6D vector:

$$b = [B11 \quad B12 \quad B22 \quad B13 \quad B23 \quad B33]T$$

$$hi^T \mathrm{B}\, hj = vij^T\, b \tag{3.67}$$

Where v equals:

$$vij = [hi1hj1 \quad hi1hj2 + hi2hj1 \quad hi2hj2 \quad hi3hj1 + hi1hj3 \quad hi3hj2 + hi2hj3 \quad hi3hj3]T$$

$$\begin{bmatrix} v12^T \\ (\mathrm{v}11 - v22)^T \end{bmatrix} * b = 0 \tag{3.68}$$

$$v * b = 0 \tag{3.69}$$

$$x0 = -\frac{B13}{\mathrm{B}11} \tag{3.70}$$

$$y0 = -\frac{B23}{\mathrm{B}22} \tag{3.71}$$

$$\alpha y \; = (\frac{B11B22B33 - B22B13^2 - B11B23^2}{B11B22^2})\frac{1}{2} \tag{3.72}$$

$$\alpha x \; = \alpha y(\frac{B23}{\mathrm{B}22})\; T \tag{3.73}$$

3.6 Camera Location

After we obtain the camera matrix, it is simple to determine the camera location by taking any two points in the real world’s three dimensions. Let us consider two points 'P1' and 'P2' whose coordinates are (X1, Y1, Z1) and (X2, Y2, Z2), respectively. Both points have a corresponding point in the image coordinate system: point 'I' (cf1, cf2, cf3, cf4) and point 'J' (cs1, cs2, cs3, cs4). This means that:

$$I\ = \mathrm{A} * \mathrm{P1} \tag{3.74}$$

$$\mathrm{J} = \mathrm{A} * \mathrm{P2} \tag{3.75}$$

As the 'A' matrix is now known, we can obtain 'P1' and 'P2' using the following equations:

$$P'1 = \frac{I}{\mathrm{A}} \tag{3.76}$$

$$P'2 = \frac{J}{\mathrm{A}} \tag{3.77}$$

As the image coordinate system has no Z axis, cf3 and cs3 are both equal to zero. From the above we can deduce that the two points P´1 and P´2 will lie on the line connecting the original point in the real world and the center of projection L, but because we are using two points it is now easy determine the camera location by drawing two lines, the first from P1 through P´1 and I to L, and the second from P2 through P´2 and J to L (Figure 11). The intersect between these two lines is the camera location.

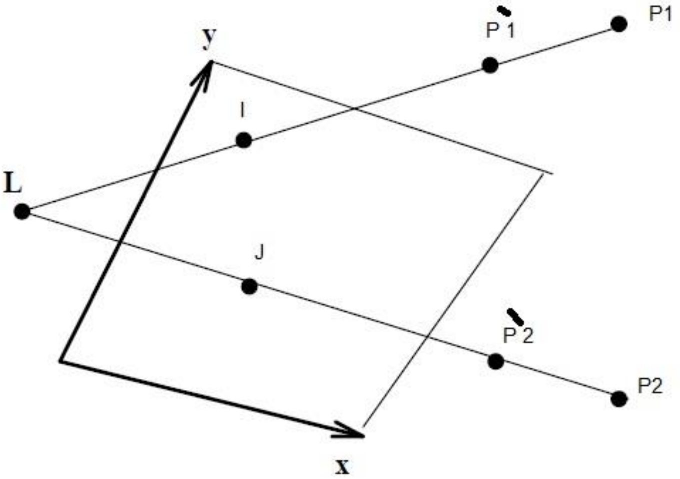


Figure 11 Determining camera location

3.7 Camera Orientation

The camera orientation is the same as that of the image plane. When the object moves closer to the lens, its image moves farther away from the image center along the Y axis. This means that when the object is at the lens, the image will be formed at infinity. The only way the image of a finite world point can be at infinity is if the four homogenous elements in the image coordinate system are equal to zero, as shown in

Figure 12 From the above we can understand the fourth row in the camera matrix which determines the camera orientation:

$$a41X + \mathrm{a42Y} + \mathrm{a43Z} + \mathrm{a44} = 0 \tag{3.78}$$

This is the equation of a plane passing through lens L parallel to the image plane.
It is now easy to recover the camera parameter from the camera matrix using the following:

$$\theta = arctan\frac{a42}{-a43} \tag{3.79}$$

$$\varnothing = arcsin\frac{-a41}{\sqrt{a41^2 + a42^2 + a43^2}} \tag{3.80}$$

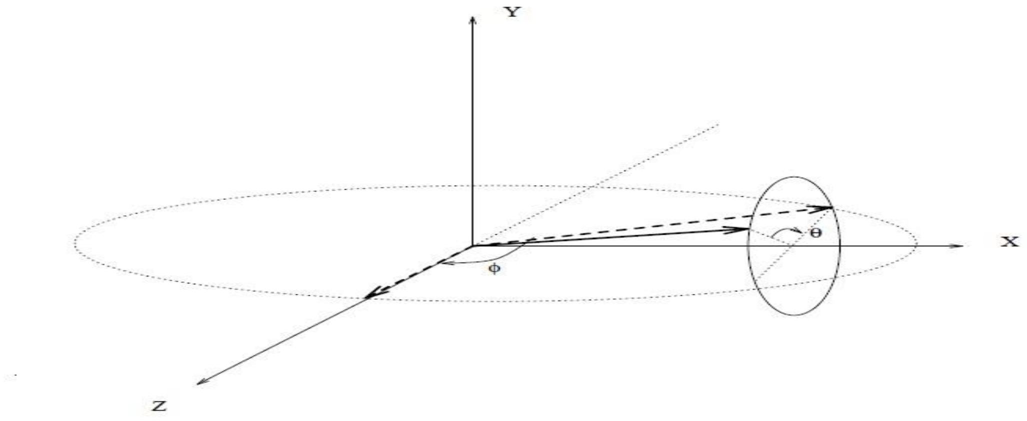


Figure 12 Camera orientation

3.8 Camera Pose Estimation

The aim of pose estimation is to determine the orientation and position of the camera which would result in the projection of a given set of three-dimensional points into a given set of image points. The most common applications of pose estimation are object recognition and making a 3D model from a 2D model. Each point of an object in the real world "3D space" has to project onto the “2D” space of the camera image coordinate system, using three rotations and three translations relative to the same base coordinate system.

Let us consider a point 'W' in the real three-dimensional world with coordinates (X1,Y1,Z1) and a second point 'w´' with coordinates (x´, y´), which is the projection of 'W' onto the two-dimensional image space. To obtain 'w´' the principal point 'W' must first be rotated by (øX, øY, øZ) to produce a new real-world point 'R' with coordinates (X, Y, Z). 'R' is then translated by (Tx, Ty, Tz), with point 'w´' finally produced from 'R' via perspective translation using a lens-centered coordinate system:

$$(\mathrm{x}', \mathrm{y}') = \left(\frac{F(X+Tx)}{\mathrm{Z}+\mathrm{Tz}}, \frac{F(Y+Ty)}{\mathrm{Z}+\mathrm{Tz}}\right) \tag{3.80}$$

In order to simplify the above equation, (Tx, Ty, Tz) can be replaced by (Dx, Dy, Dz), where 'Dx' and 'Dy' are the displacement in the first two dimensions and 'Dz' is the displacement in the third dimension. As a result we can rewrite the above equation as follows:

$$(\mathrm{x}', \mathrm{y}') = \left(\frac{FX}{\mathrm{Z}+\mathrm{Dz}} + Dx, \frac{FY}{\mathrm{Z}+\mathrm{Dz}} + Dy\right) \tag{3.81}$$

$$(\mathrm{x}', \mathrm{y}') = (FXc + Dx, \mathrm{FYc} + Dy) \tag{3.82}$$

Where

$$\mathrm{c} = \frac{1}{\mathrm{Z}+\mathrm{Dz}} \tag{3.83}$$

Pose estimation can be formulated as an optimization problem which minimizes the error between the model and image coordinate system [26]. The error existing between the elements of model and image can be determined by tiny changes in the six unknown elements (Dx, Dy, Dz, øX, øY, øZ), partially derived with respect to the image coordinates. The equation for error 'R' in the 'x' image coordinate, expressed as the sum of the products of its partial derivative multiplied by the error correction values, can be written as follows:

$$R = \frac{\omega x'}{\omega DX} \blacktriangle DX + \frac{\omega x'}{\omega DY} \blacktriangle DY + \frac{\omega x'}{\omega DZ} \blacktriangle DZ + \frac{\omega x'}{\omega \text{ø}X} \blacktriangle \text{ø}X + \frac{\omega x'}{\omega \text{ø}Y} \blacktriangle \text{ø}Y + \frac{\omega x'}{\omega \text{ø}Z} \blacktriangle \text{ø}Z \tag{3.84}$$

If (X, Y, Z) is rotated about the 'Y' axis by 'øY', the new coordinate of this point is given by the following:

$$X2 = \mathrm{X1}\cos\ øY + Z\ sin\ øY \qquad (3.85)$$

$$Y2 = Y1 \qquad (3.86)$$

$$\mathrm{Z2} = -\mathrm{X1}\sin\ øY + Z1\cos\ øY \qquad (3.87)$$

Which means that:

$$\frac{\omega Z}{\omega øY} = \mathrm{X1}\cos\ øY + Z\ sin\ øY = -X2 \qquad (3.88)$$

$$\frac{\omega Y}{\omega øY} = 0 \qquad (3.89)$$

$$\frac{\omega X}{\omega øY} = -\mathrm{X1}\sin\ øY + Z1\cos\ øY = Z2 \qquad (3.90)$$

For 'k' image points we can write '2*k' equations with six unknowns, with this linear system of equations expressed as follows:

$$\mathrm{A}\ \blacktriangle\ = R \qquad (3.91)$$

where '▲' is the vector of the six unknown parameters, as shown in Table 1, A is the derivative matrix, and R is the vector of the error. Calculation of '▲' can be achieved via the following:

$$\blacktriangle\ = (A^{-T}A^{-1})^{-1}\ A^{-T}\ R \qquad (3.92)$$

	x'	y'
D_X	1	0
D_Y	0	1
D_Z	$-fc^2X$	$-fc^2Y$
ϕ_X	$-fc^2XY$	$-fc(Z + cY^2)$
ϕ_Y	$fc(Z + cX^2)$	fc^2XY
ϕ_Z	$-fcY$	fcX

Table 1 Partial Derivatives of Image Coordinates

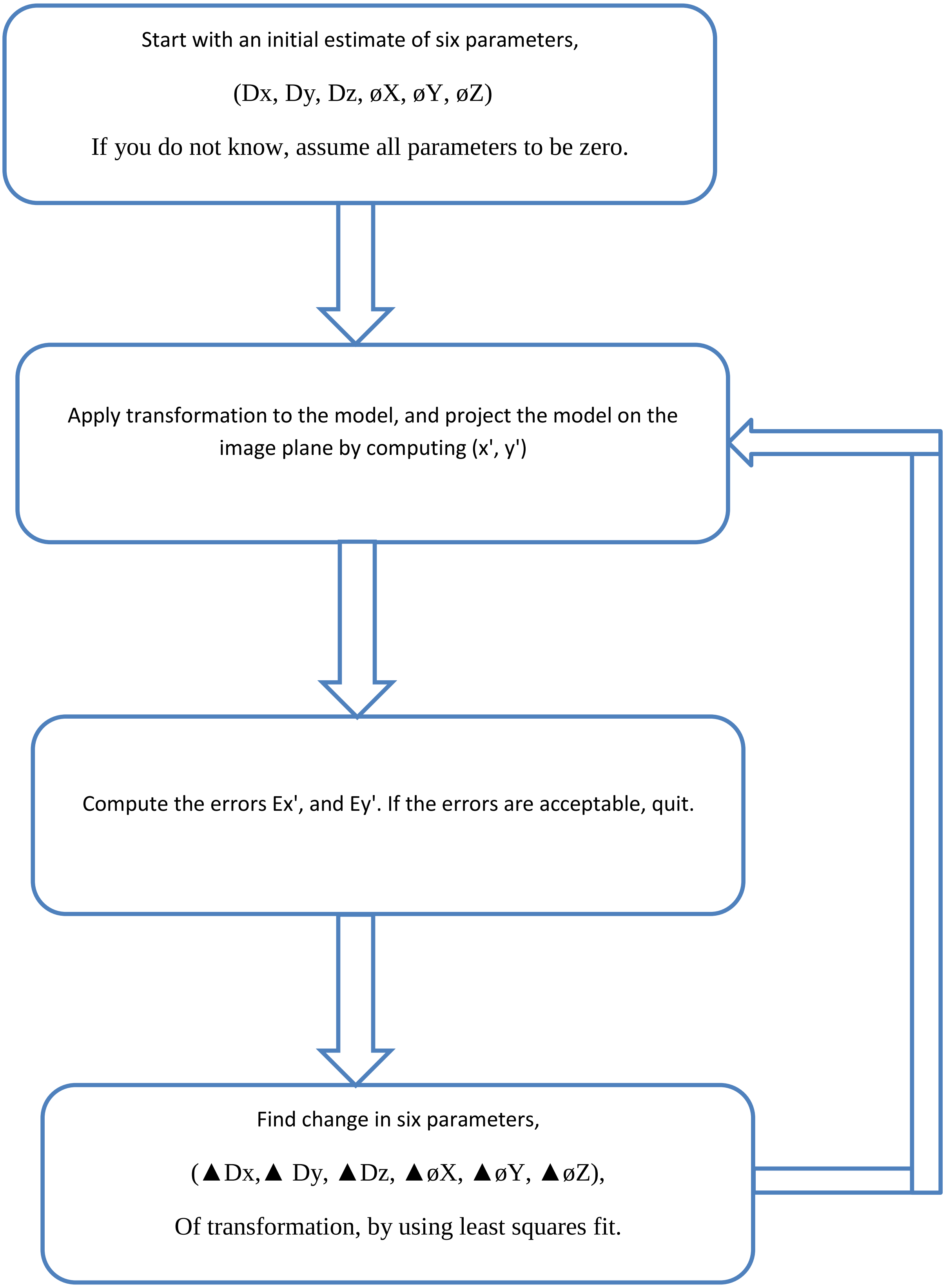

Start with an initial estimate of six parameters,
(Dx, Dy, Dz, øX, øY, øZ)
If you do not know, assume all parameters to be zero.
Apply transformation to the model, and project the model on the image plane by computing (x', y')
Compute the errors Ex', and Ey'. If the errors are acceptable, quit.
Find change in six parameters,
(▲Dx,▲ Dy, ▲Dz, ▲øX, ▲øY, ▲øZ),
Of transformation, by using least squares fit.

Figure 13 Pose algorithm

3.9 Epipolar Geometry

As discussed earlier in the chapter, when any point in the real three-dimensional world is imaged by a camera it will lose its 'Z' axis value. If we therefore consider a point P in the real world and draw a straight line intersecting P and the optical center of the camera, each point on this line will project to the same position in the image, resulting in a loss of image depth. In order to solve this problem we will now consider a more advanced case of point translation with which to estimate image depth or the z axis, known as epipolar geometry. This involves translating the same point onto two image planes simultaneously using two cameras placed at different locations. The motion vector of this point to the cameras will determine its 'z' axis value, and as a result we must find the corresponding point on the image, as shown in Figure 14. In this case we don't need to inspect the point across the entire image because in fact we have only one degree of freedom for its possible location in the second image.

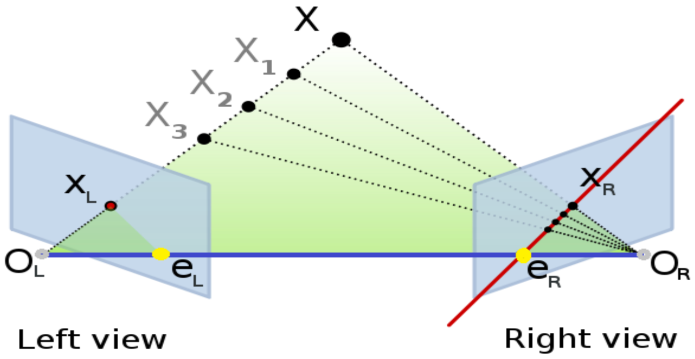


Figure 14 Epipolar geometry

Based on the above figure the following terms should first be defined:

1- Base-line: the straight line between the two optical centers of the right- and left-hand cameras (blue line).
2- Epipolar plane: a triangle containing the base-line as its base, and its three corners being the optical centers of the cameras (gray dots) and a point in the real world (X).
3- Epipole: a point of intersect between the base-line and the image plane (yellow dots), produced by the projection of each camera’s center of projection onto the opposing camera’s image plane.

4- Epipolar line: the line created by the intersection between the epipolar plane and the image plane; this line must pass through the epipole.

Any point on the epipolar line of one image plane should therefore have a corresponding point somewhere on the epipolar line of the other image plane.

3.9.1 Essential Matrix

This matrix is a 3x3 matrix that "encodes" the epipolar geometry of the two views, as well as the other properties described below, that deal with corresponding points in stereo cameras. In order to obtain an essential matrix, we will first denote 'P' as a point in the real world, 'Pl' as the projection of point 'P' onto the left-hand camera image plane, 'Pr' as the projection of point 'P' onto the right-hand camera image plane, and 'B' as the base-line. This means that:

$$\mathrm{Pr} = \mathrm{Pl} - \mathrm{B} \tag{3.93}$$

From the above, (Pl, Pr & B) are coplanar as follows:

$$\mathrm{Pr}. B * Pl = 0 \tag{3.94}$$

The two vectors "Pl & Pr" can now be defined based on their respective optical centers, in this case "Pl" with "Ol" and "Pr" with "Or". This is followed by a translation on "B" and the application of a rotation "R" matrix, resulting in:

$$Pr' = R\,Pr = R\,(\,Pl - B\,) \tag{3.95}$$

$$Pr = R^{-1}\,Pr' = R^{T}\,\mathrm{Pr}' \tag{3.96}$$

and substituting gives:

$$(R^{T} Pr')B * Pl = 0 \tag{3.97}$$

As base-line 'B' is the translation of the point between the two cameras, translation vector 'T' is equal to:

$$T = \begin{bmatrix} 0 & -Tz & Ty \\ Tz & . & -Tx \\ -Ty & Tx & 0 \end{bmatrix}$$

This means that:

$$(R^{T} Pr'\,)^{T} T * Pl = 0 \tag{3.98}$$

$$Pr'^{T}(R\ T)\ Pl = 0 \tag{3.99}$$

Denoting 'E' for (R T), we can rewrite the equation as follows:

$$Pr'^{T}E\ Pl = 0 \tag{3.100}$$

As the above equation expresses the relationship between the same point in the two cameras, the equation for the epipolar line can be written as follows:

$$\mathrm{Pl} = \frac{Fl * P}{Zl} \tag{3.101}$$

$$\mathrm{Pr} = \frac{Fr * P}{Zr} \tag{3.102}$$

Eliminating 'Pl' & 'Pr´' and dropping the prime we find:

$$Pr^{T}E\ Pl = 0 \tag{3.103}$$

As 'Zl', 'Zr ' and ' fl', 'fr ' can be canceled from this matrix equation, it follows that:

$$Pr^{T}E\ = I1^{T} \tag{3.104}$$

$$I2^{T}\ = \mathrm{E\ Pl} \tag{3.105}$$

The two equations above lead us to the following relations:

$$Pr^{T}I2^{T}\ = 0 \tag{3.106}$$

$$Pl^{T}I1^{T}\ = 0 \tag{3.107}$$

This means that:

$$\mathrm{E\ Pl}\ = \mathrm{I2} \tag{3.108}$$

$$E^{T}Pr\ = \mathrm{I1} \tag{3.109}$$

We can now obtain the epipolar lines corresponding to 'pl' and 'pr', respectively, and ultimately find the epipoles from the above formulation. In fact, the epipole lies on every epipolar line within the same image. Thus, 'er' satisfies (can be substituted for 'pr' in) the above equation, and hence:

$$er^{T}I2 = 0 \tag{3.110}$$

$$er^{T}E\ Pl = 0\ \ \text{(for all 'Pl')} \tag{3.111}$$

3.9.2 Fundamental Matrix

We implicitly assumed in the final part of the essential matrix calculation that the cameras are calibrated. The present section thus examines the scenario for uncalibrated images. Let us denote the intrinsic matrix for the left camera as 'A', the intrinsic matrix for the right camera as 'B', the point in the left image plane as 'I', the point in the right image plane as 'J', and the two points in the real world as 'P1' and 'P2'. This means that:

$$\mathrm{I} = \mathrm{A} * P1 \tag{3.112}$$

$$\mathrm{J} = \mathrm{B} * P2 \tag{3.113}$$

As we actually need to obtain the reverse direction, the inverse equations are required:

$$\mathrm{P1} = A^{-1} * I \tag{3.114}$$

$$\mathrm{P2} = B^{-1} * J \tag{3.115}$$

Substituting 'p1' and 'p2' in the essential matrix equation:

$$(j * B^{-1})^T * E * (j * A^{-1}) = 0 \tag{3.116}$$

$$J^T * (B^{-1})^T * E * A^{-1} * I = 0 \tag{3.117}$$

$$J^T * \mathrm{F} * I = 0 \tag{3.118}$$

$$F = (B^{-1})^T * E * A^{-1} \tag{3.119}$$

'F' is defined as the “fundamental matrix”. Because it contains all the information that would be needed to calibrate the cameras, it contains more free parameters than the essential matrix.

3.10 Recovering Image Depth 'Z'

Almost all photography involves converting a point from three dimensions to two dimensions during image capture. However, camera modifications can be made to recover the third dimension "z" Figure 15 via the use of active methods or extra devices such as ultra-sound sonar or laser range finders. As discussed earlier, image depth can be recovered if we use two optically aligned cameras in "stereo vision" and apply epipolar geometry.

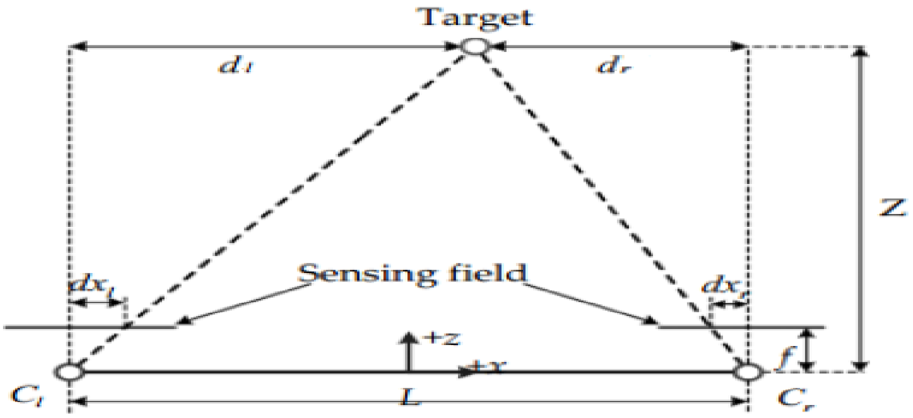


Figure 15 Image depth

In the above figure the point in the real world is denoted as "Target". This point is projected onto the two-camera stereo system, with distances of 'dXl' pixels from the optical center of the left-hand camera and 'dXr' pixels from the optical center of the right-hand camera. 'L' is the base-line, 'Z' is image depth, 'dl' is the distance from the optical center of the left-hand camera to the point in the real world, 'dr' is the distance from the optical center of right-hand camera to the point in the real world, and 'f ' is the focal length. Using triangulation we obtain:

$$\frac{dl}{-dXl} = \frac{Z}{f} \tag{3.120}$$

$$dl = \frac{-dXl * Z}{f} \tag{3.121}$$

$$\frac{dr}{dXr} = \frac{Z}{f} \tag{3.122}$$

$$dr = \frac{dXr * Z}{f} \tag{3.123}$$

$$\mathrm{L} = \mathrm{dl} + dr \tag{3.124}$$

$$L = \frac{-dXl * Z}{f} + \frac{dXr * Z}{f} \tag{3.125}$$

$$L = \frac{-dXl * Z + dXr * Z}{f} \tag{3.126}$$

$$L = \frac{Z(dXr - dXl)}{f} \tag{3.127}$$

$$Z = \frac{L * f}{(dXr - dXl)} \tag{3.128}$$

$$Z = \frac{L * f}{(dXr - dXl) * PS} \tag{3.129}$$

where 'PS' is pixel size in centimeters.

3.11 Object Size Estimation

Object size estimation is essential to a variety of applications. Similarly, knowledge of the real size of an object captured by a camera is very important for the accurate classification this object ("an ant is smaller than an elephant"). Object size estimation can be carried out by determining the distance between two points in the real world using a camera. Let us denote the first point as 'A' and the second point as 'B', as shown in Figure 16.

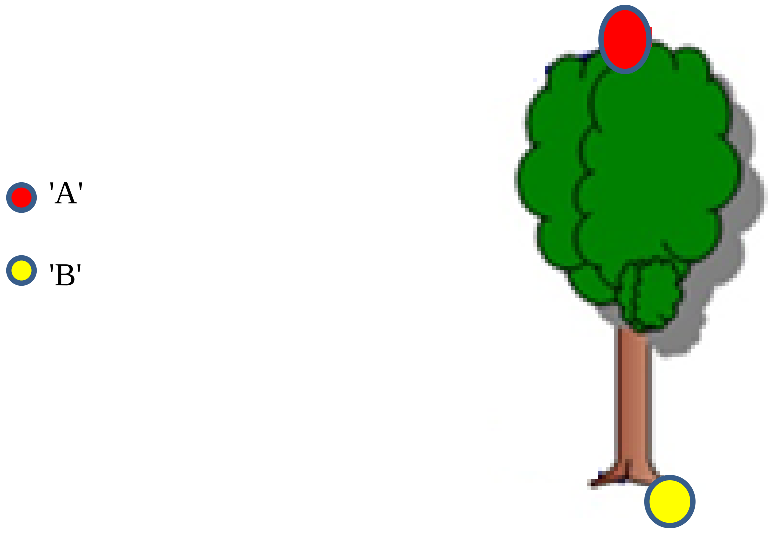


Figure 16 Example of size estimation

If we know the distance between the camera and point 'A', the distance between the camera and point 'B', and the angle of intersect of the two lines from 'A' and 'B' to the optical center of the camera 'ø', the following triangle is produced in figure 17:

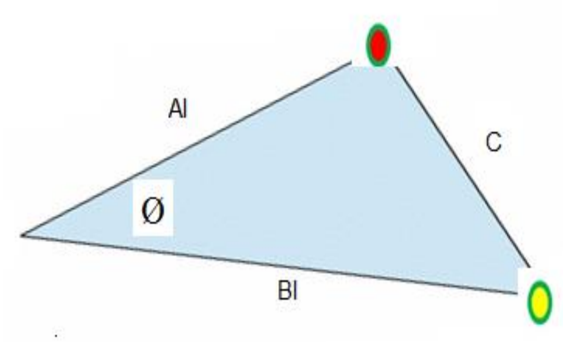


Figure 17 Triangle between object and camera

We can calculate 'C' using the following equation:

$$\mathrm{C} = \sqrt{Al^2 + Bl^2 - Al * Bl * cos\emptyset} \tag{3.130}$$

Equation 3.129 describes how to obtain 'Al' and 'Bl'. The angle between 'Al' and 'Bl' can be determined using the following figure:

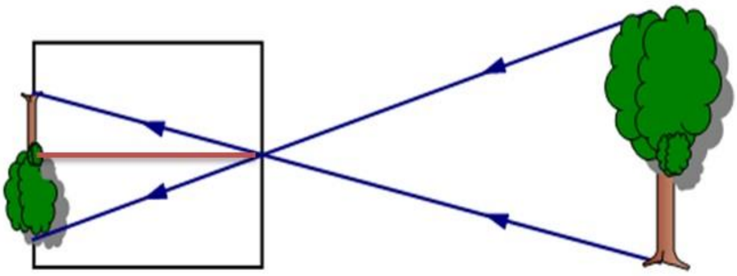


Figure 18 Triangle of the object inside the camera

The two focal length lines intersect to make two right-angled triangles in the image plane. In these triangles the base and rib are known, with Pythagoras’ proof (Figure 19) then applied to obtain the chord.

$$a^2 + b^2 = c^2 \tag{3.131}$$

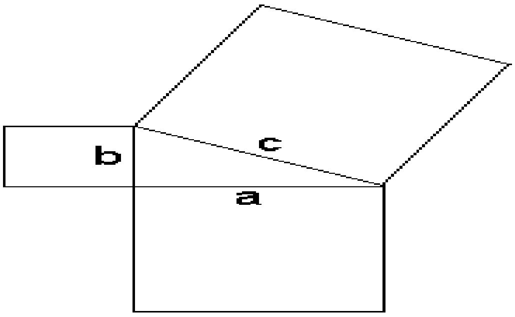


Figure 19 Pythagoras’ proof

After applying this twice to find "Al' " and "Bl' ", and as the base is already known, the following equation can then be used to obtain the cos of ø:

$$\cos ø = \frac{Al'^2 + Bl'^2 - C'^2}{Al' * Bl'} \qquad (3.132)$$

With this knowledge we can then calculate the length and width of the object.

Chapter 4

IMAGE PROCESSING

A color image can be defined as a 2D light intensity function "$f(x, y)$", where *(x* and *y)* are spatial coordinates and the value of f in (x, y) is similar to the brightness of the scene at that point. For a multispectral image, $f(x, y)$ is a vector, each component of which indicates the brightness of the scene at point (x, y) in the corresponding spectral band.

A digital image is made up of individual picture elements known as pixels. Typically, pixels are organized in an ordered rectangular array. The size of an image is thus determined by the dimensions of this pixel array, with the image width the number of columns and the image height the number of rows in the array. A pixel array can therefore be defined as a matrix of 'M' columns and 'N' rows.

To refer to a specific pixel within the image matrix, we define its coordinates at 'x' and 'y'. The coordinate system of image matrices defines 'x' as increasing from left to right and 'y' as increasing from top to bottom. Compared to normal mathematical convention, the origin is in the top left corner and the 'y' coordinate is flipped. Why is the coordinate system flipped vertically? Originally, digital images were defined in terms of the electron beam scanning pattern of televisions, which scanned from left to right and top to bottom. Other than this historical reason, there is no purpose served by this inversion of the y coordinate. Image size is not to be confused with the size of the real-world representation of an image. Image size specifically describes the number of pixels within a digital image. The real-world representation of a digital image requires one additional factor known as resolution. Resolution is the spatial scale of the image pixels.

$$f(x,y)=\begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & f(1,4) & . & . & . & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & f(2,4) & . & . & . & f(2,N) \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ f(M,1) & f(M,2) & f(M,3) & f(M,4) & . & . & . & f(M,N) \end{bmatrix} \qquad (4.1)$$

4.1 Color Spaces

The color space is a mathematical classification of a group of colors. The color space most commonly used in graphical and another applications is "RGB", where 'R' refers to the color red, 'G' to green, and 'B' to blue. Other types of color space include YUV, YIQ, and YCbCr. Each of these color spaces has a special function, with YCbCr, for example, used in video systems. However, all types of color space have three basic concepts: "saturation", "hue", and "brightness". A variety of different models have since been developed, such as HSI and HSV, aimed at making color systems easy to understand, and are employed in processing and programming. The "RGB" color space is considered essential because vision sensors typically capture images using this system. As a result, all other systems can be directly related to the "RGB" color space.

4.1.1 RGB Color Space

As discussed above, the RGB system is produced by mixing the colors R, G, and B. Every other color is therefore a mixture of these three base colors in different proportions, as shown in Table 2. This space is typically represented using the Cartesian coordinate system as illustrated below figure:

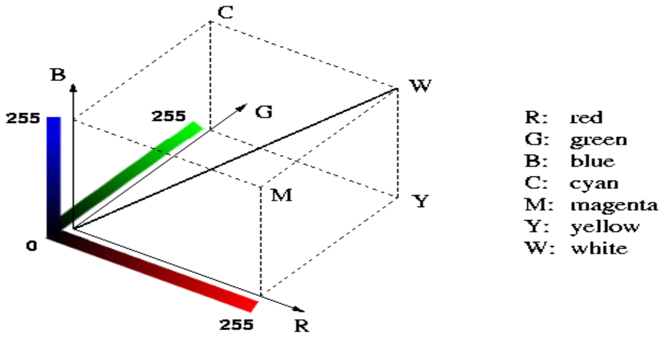
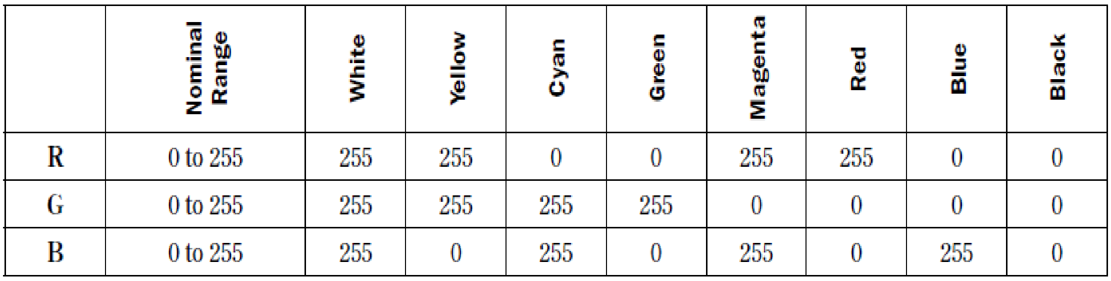


Figure 20 The RGB color cube



	Nominal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
R	0 to 255	255	255	0	0	255	255	0	0
G	0 to 255	255	255	255	255	0	0	0	0
B	0 to 255	255	0	255	0	255	0	255	0

Table 2 100% RGB Color Bars

Although this system has a wide range of applications and almost all vision sensors capture images in "RGB", it is not appropriate for image processing due to the fact that each color is derived from a mixture of three other colors. As each color has 226 levels, the color space comprises more than 16 million colors, leading to a lack of precision in the results.

4.1.1.1 Gray Color Space

In this space, colors are represented only by the intensity of the original color (in the RGB space), meaning that this color space contains only 256 colors ranging from black to white. The grays comprising this range are more suitable for use in image processing. Three methods are available with which to convert an image from RGB to Gray:

1- *Lightness method*: based on the average of the most and the least prominent colors figure 21-a.

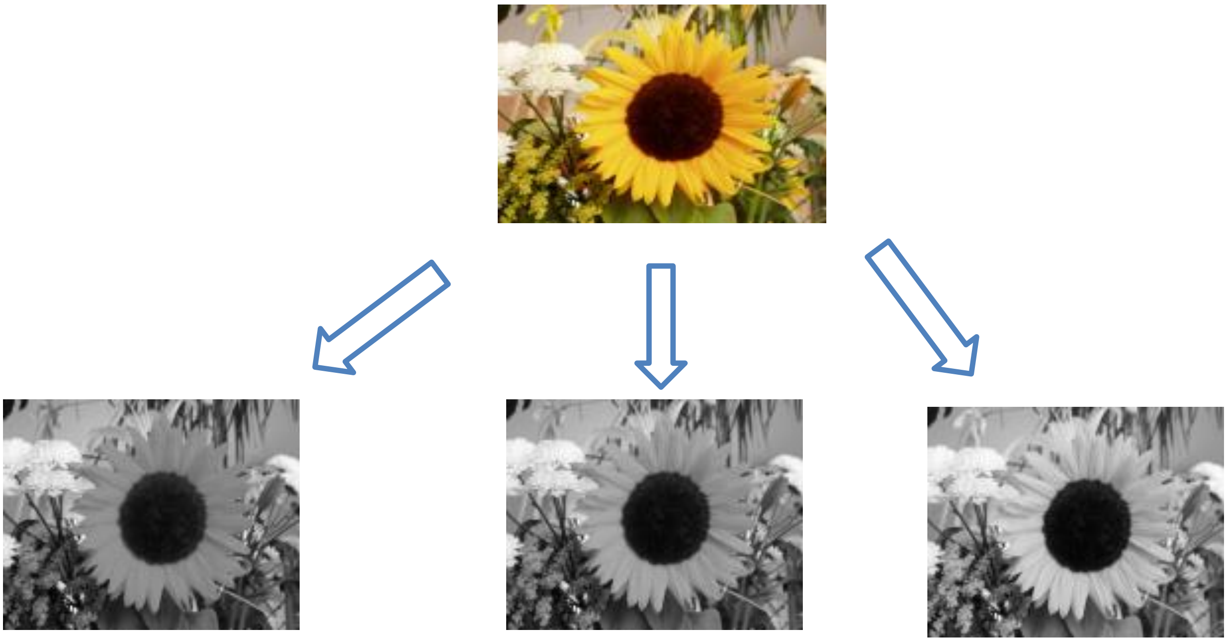
$$Gray = Max\,(R,G,B) + Min\,(R,G,B) \;/\; 2 \qquad (4.2)$$

2- *Average method*: the three-color average figure 21-b.

$$Gray = R + G + B \;/\; 3 \qquad (4.3)$$

3- *Luminosity method*: this method is more complex than the previous two because it depends on the sensitivity of humans to different colors figure 21-c. As humans are more sensitive to green, followed by red then blue, a weight can be applied as follows to generate the color space:

$$Gray = 0.21 * R + 0.72 * G + 0.07 * B \qquad (4.4)$$



a b c

Figure 21 Gray color space methods: a) lightness method, b) average method, c) luminosity method

4.1.2 YUV Color Space

This space is used in three types of video system (SECAM, NTSC and PAL). The 'Y' in YUV represents a type of gray space comprising black and white color information, with the other components derived by subtracting the 'Y' value from the red and blue components of the original RGB space as illustrated below figure.

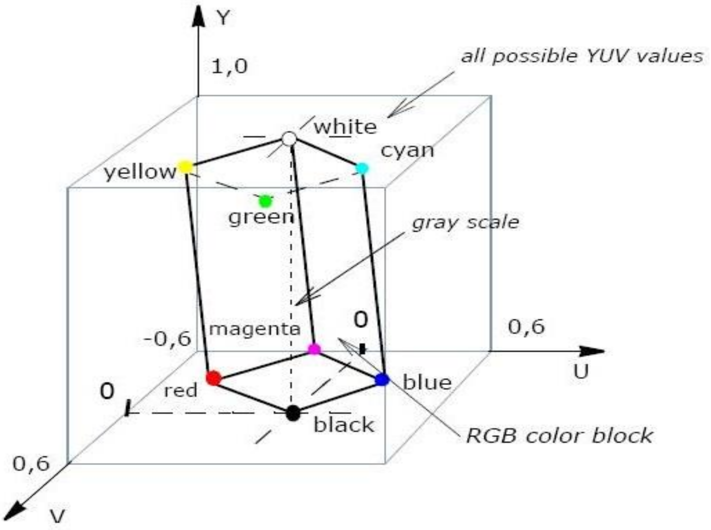


Figure 22 YUV color space

Colors are translated from the RGB space to the YUV (figure 23) space as follows:

1- As outlined earlier, the RGB space comprises three layers, with the color range of each color varying from [0-255]. For translation to the YUV space, we must change this range to [0-1] for each layer using the following equations:

$$R' = R \;/\; 255 \tag{4.5}$$

$$G' = G \;/\; 255 \tag{4.6}$$

$$B' = B \;/\; 255 \tag{4.7}$$

These equations can be rewritten in matrix form as follows:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} \frac{1}{255} & 0 & 0 \\ 0 & \frac{1}{255} & 0 \\ 0 & 0 & \frac{1}{255} \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.8}$$

2- The RGB colors can then be applied to the YUV space as:

$$Y = 0.299 * R' + 0.587 * G' + 0.114 * B' \tag{4.9}$$

$$U = -0.147 * R' - 0.289 * G' + 0.436 * B' \tag{4.10}$$

$$V = 0.615 * R' - 0.515 * G' - 0.100 * B' \tag{4.11}$$

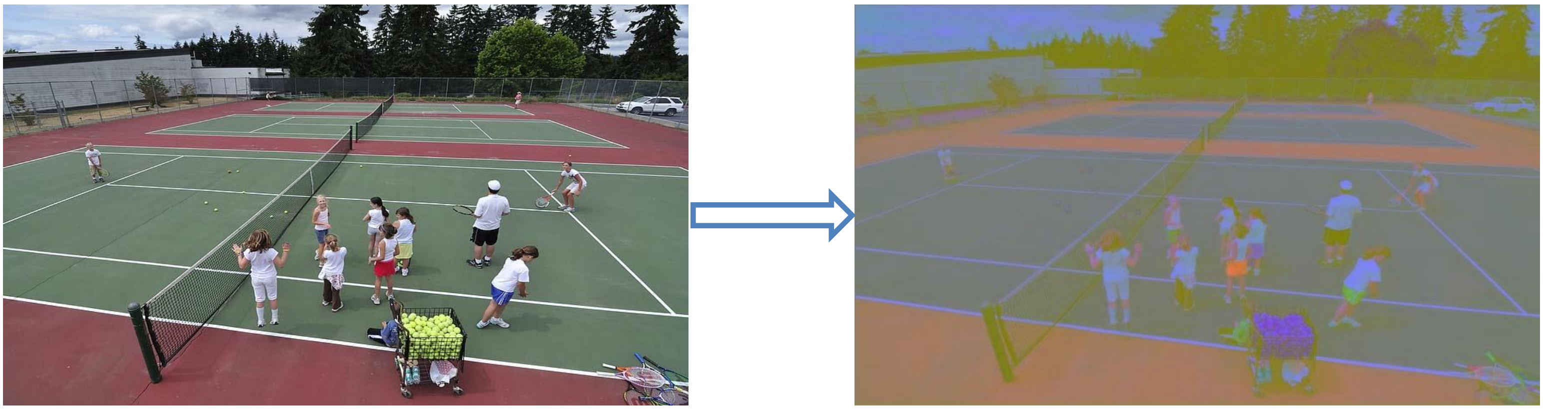
The above equations can be rewritten in matrix form as follows:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.439 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} * \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.12}$$

This means that the color range of the YUV space differs from the RGB color range space by the values in the following Table:

	FROM	TO
Y	0	255
U	0	± 112
V	0	± 157

Table 3 YUV Value Bars



a b

Figure 23 RGB conversion to YUV: a) image in the RGB color space, b) image in the YUV color space

We can also transfer colors from the YUV space system to the RGB space system (figure 24) using the following equations:

$$R' = Y + 1.140 * V \tag{4.13}$$

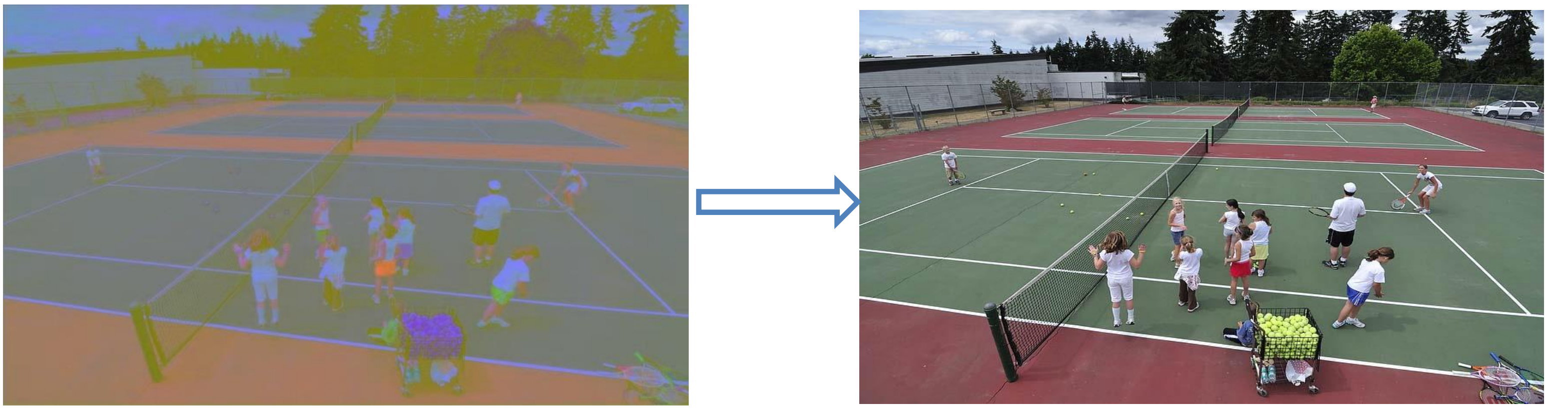
$$G' = Y - 0.395 * U - 0.581 * V \tag{4.14}$$

$$B' = Y + 2.032 * U \tag{4.15}$$

$$R = R' * 255 \tag{4.16}$$

$$G = G' * 255 \tag{4.17}$$

$$B = B' * 255 \tag{4.18}$$



a b

Figure 24 YUV conversion to RGB: a) image in the YUV color space, b) image in the RGB color space

4.1.3 YIQ Color Space

The YIQ color space is used in video systems such as the NTSC TV system. Derived from the YUV color space, in this case the 'Y' represents a gray space and (I & Q) carry the chrominance information as illustrated below figure:

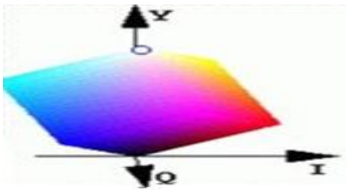


Figure 25 YIQ color space

The translation of colors from the RGB space to the YIQ space (figure 26) involves the following steps:

1- As the RGB space comprises three layers, with each color ranging from [0-255], this range must be converted into [0-1] for each layer using equations [4.4, 4.5, and 4.6].

2- RGB colors can then be applied to the YIQ space via the following equations:

$$Y = 0.299 * R' + 0.587 * G' + 0.114 * B' \tag{4.19}$$

$$I = 0.596 * R' - 0.275 * G' - 0.321 * B' \tag{4.20}$$

$$Q = 0.212 * R' - 0.523 * G' + 0.311 * B' \tag{4.21}$$

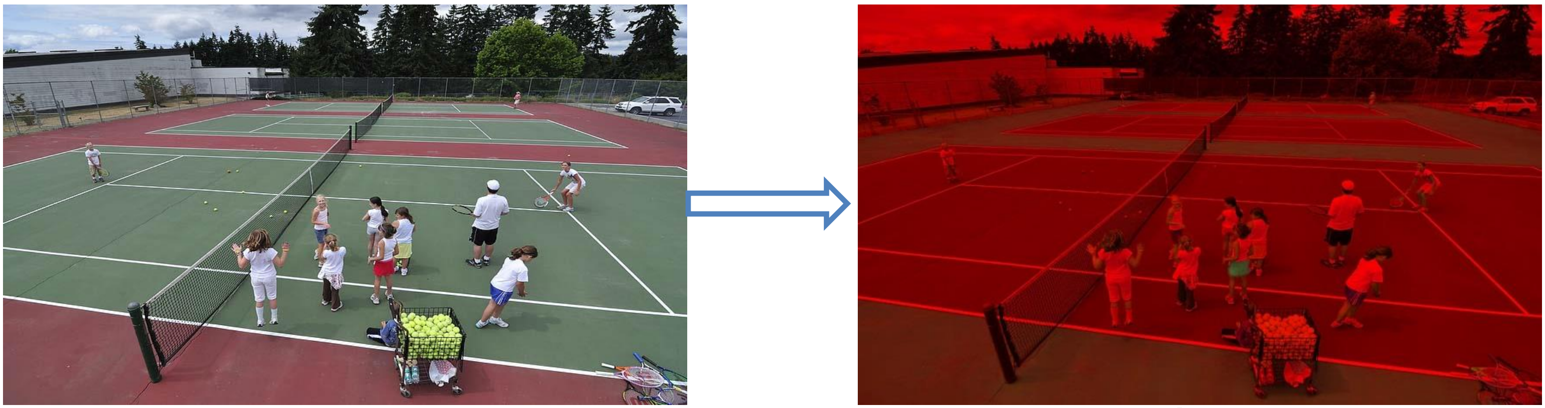
These equations can be rewritten in matrix form as follows:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} * \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.22}$$

This means that the color range of the YIQ space differs from the RGB color range space by the values in the following Table:

	FROM	TO
Y	0	255
I	0	± 152
Q	0	± 134

Table 4 YIQ Value Bars



a b

Figure 26 RGB conversion to YIQ: a) image in the RGB space, b) image in the YIQ color space

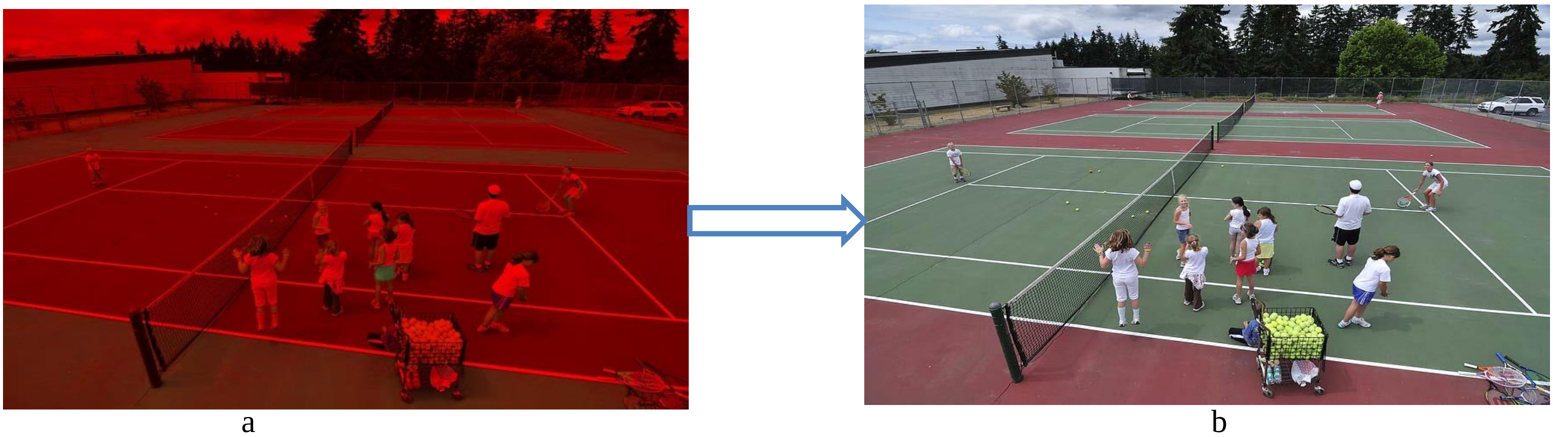
Colors can also be translated from the YIQ space system to the RGB space system (figure 27) using the following equations:

$$R' = Y + 0.956 * I + 0.621 * Q \qquad (4.23)$$

$$G' = Y - 0.2721 * I - 0.647 * Q \qquad (4.24)$$

$$B' = Y - 1.107 * I + 1.704 * Q \qquad 4.25)$$

Equations [4.15, 4.16 and 4.17] can then be used to obtain [R, G and B].



a b

Figure 27 YIQ conversion to RGB: a) image in the YIQ color space, b) image in the RGB color space

4.1.4 HSI Color Space

Attempting to produce a more intuitive representation of colors, the HSI color space has three components: 'H'ue, 'S'aturation and 'I'ntensity. The system is derived from a double cone shape with one axis running down its center, representing 'I'. A range of gray colors run the length of this axis, with black and white at either end. The figure below illustrates how colors are represented in the HSI color space.

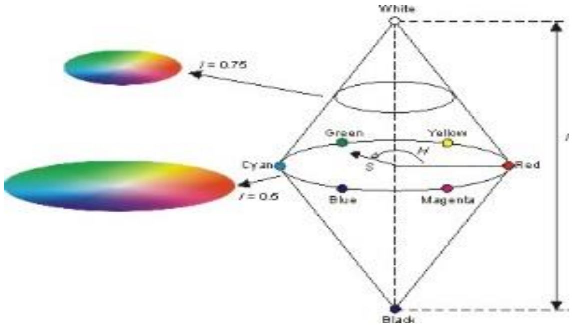


Figure 28 HSI color space

If the cone is viewed from above, it becomes a circle. Different colors, or hues, are defined as having specific positions around this circle. Hues are determined by their angular location on this wheel, with red at 0 degrees and the others progressing as illustrated below figure:

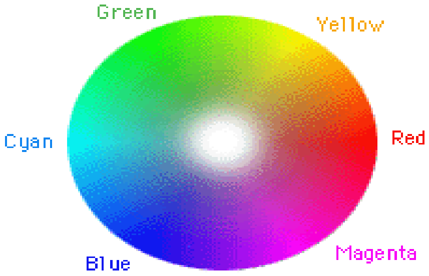


Figure 29 HSI color circle

Saturation is measured in terms of vertical distance from the intensity axis. Colors closer to the center of intensity are lighter, while those far from the center are distinct in emersion. The HSI system is considered more appropriate for use in image processing than the RGB system. Detecting and changing the yellow color in an image would be an impossible task using the RGB system, because this color has a lot of values. In contrast, the HSI system includes a range of yellow colors. Such a task could proceed as follows figure:

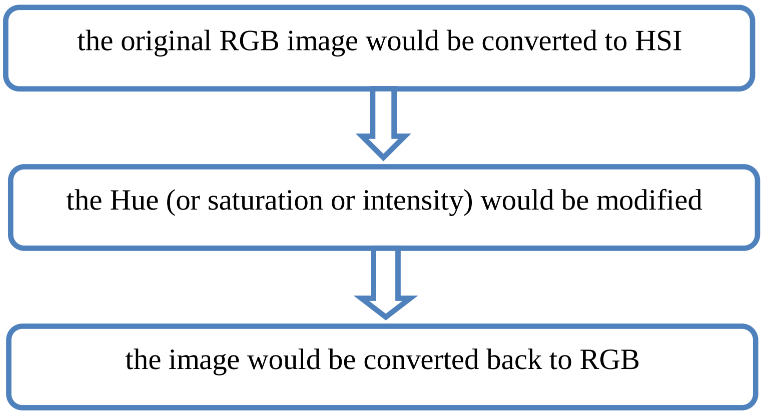

the original RGB image would be converted to HSI
the Hue (or saturation or intensity) would be modified
the image would be converted back to RGB

Figure 30 Color selection in an RGB image using the HSI color system

The following steps are involved in transferring colors from the RGB space to the HSI (figure 31) space:

1- Normalization of RGB color values:

$$r = \frac{R}{R+G+B} \tag{4.26}$$

$$g = \frac{G}{R+G+B} \tag{4.27}$$

$$b = \frac{B}{R+G+B} \tag{4.28}$$

2- 'H', 'S' and 'I' are normalized as follows:

$$s = 1 - \min(r, g, b) \quad s \in [0,1] \tag{4.29}$$

$$i = (R+G+B)/(3*255) \quad i \in [0,1] \tag{4.30}$$

If $(b \leq g)$:

$$h = \cos^{-1}\left\{ \frac{0.5 \cdot [(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{1/2}} \right\} \qquad h \in [0, \pi] \tag{4.31}$$

Else:

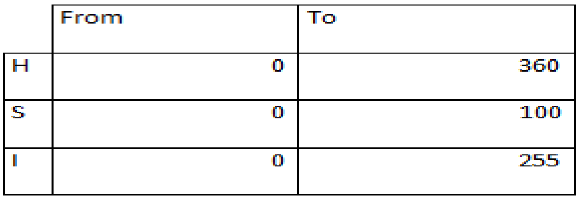
$$h = 2\pi - \cos^{-1}\left\{ \frac{0.5 \cdot [(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{1/2}} \right\} \qquad h \in [\pi, 2\pi] \tag{4.32}$$

Now we must convert [h s i] to [H S I] using the following equations:

$$H = h * \frac{180}{\pi} \tag{4.33}$$

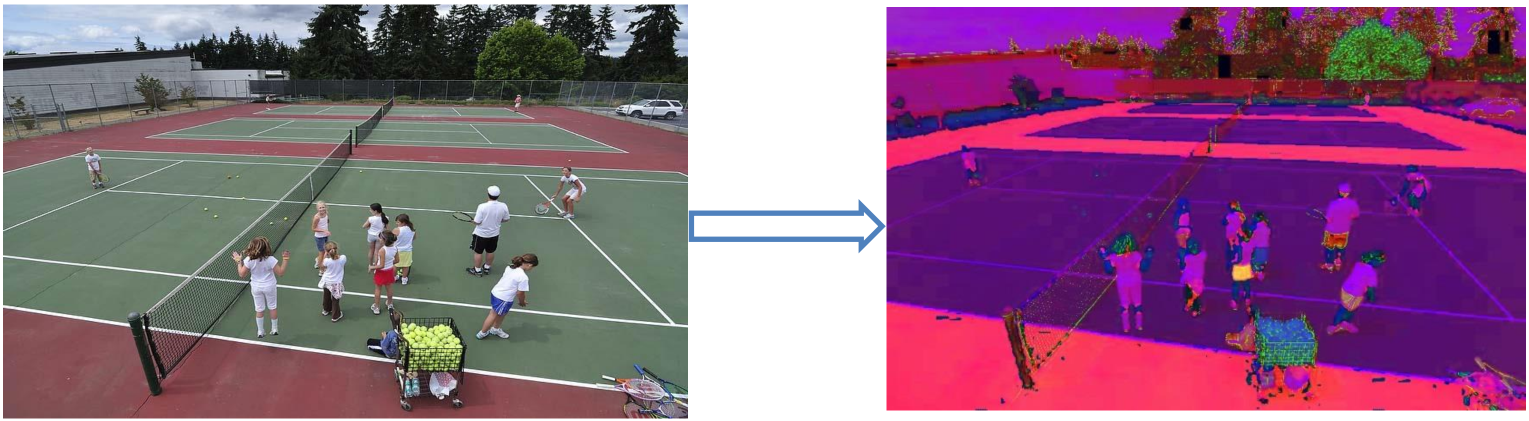
$$S = s * 100 \tag{4.34}$$

$$I = i * 255 \tag{4.35}$$



	From	To
H	0	360
S	0	100
I	0	255

Table 5 HSI Value Bars



a

b

Figure 31 RGB conversion to HSI: a) image in the RGB color space, b) image in the HSI color space

We can also transfer colors from the HSI space system to the RGB space system (figure 32) using the following equations:

$$h = \frac{H * \pi}{180} \tag{4.36}$$

$$s = \frac{S}{100} \tag{4.37}$$

$$i = \frac{I}{100} \tag{4.38}$$

$$x = i * (1 - s) \tag{4.39}$$

$$y = i * [1 + \frac{S * \cos(h)}{\cos\left(\frac{\pi}{3} - h\right)}] \tag{4.40}$$

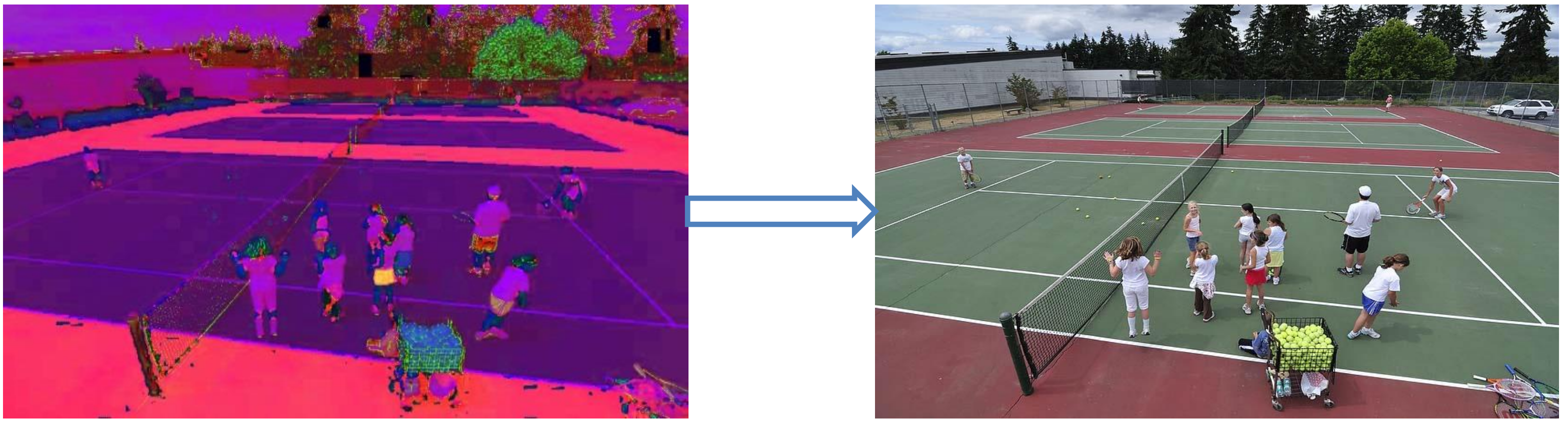
$$z = 3 * i - (x + y) \tag{4.41}$$

$$\text{where } h < \frac{2\pi}{3} \rightarrow r = y, g = z\ and\ b = x$$

$$\text{where } \frac{2\pi}{3} \leq h < \frac{4\pi}{3} \rightarrow r = x, g = y\ and\ b = z$$

$$\text{where } \frac{4\pi}{3} \leq h < 2\pi \rightarrow r = z, g = x\ and\ b = y$$

Then by using equations [4.16, 4.14 and 4.18] we obtain [R, G and B].



a b

Figure 32 HSI conversion to RGB: a) image in the HSI color space, b) image in the RGB color space

4.1.5 HSV Color Space

Based on human color perception, the HSV color system represents colors in terms of three components: hue, saturation, and value. Whereas the RGB color space generates colors by mixing the three essential colors red, green and blue, in the HSV space colors are generated from the factors of brightness, vibrancy, and color. The H, S and V components (figure 33) can be defined as follows:

1- Hue is represented by a circle containing the full color range, where the zero degree point is the original color. For example, if red is at zero degrees, the two hundred and forty degree point represents blue.
2- Saturation is a ratio of color purity ranging from 0 to 100. Values closer to zero are lighter and those closer to 100 more saturated and bold.
3- Value essentially represents the lightness value of the original color in the RGB system.

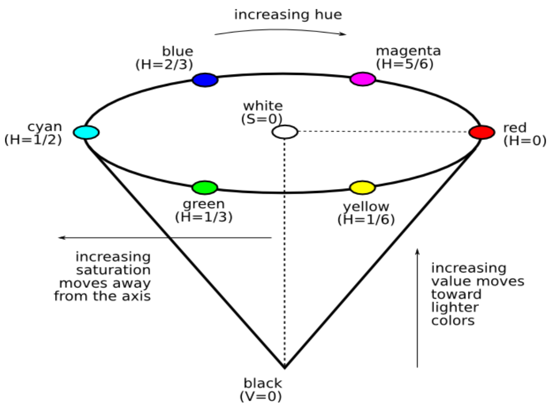


Figure 33 HSV color space system

Colors can be translated from the RGB space system to the HSV space system (figure 34) using the following steps:

1- Normalization of RGB color values via equations [4.26, 4.27 and 4.28].

2- Acquisition of 'max' and 'min' values, as well as the difference between them ‘delta’.

$$Cmax = Max(r, g, b) \tag{4.42}$$

$$Cmin = Min(r, g, b) \tag{4.43}$$

$$delta = Cmax - Cmin \tag{4.44}$$

3- Calculation of **H**ue:

$$T = \frac{g-b}{Cmax-Cmin} \quad \text{if (r == max)} \tag{4.45}$$

$$T = \frac{b-t}{Cmax-Cmin} + 2 \quad \text{if (g == max)} \tag{4.46}$$

$$T = \frac{r-g}{Cmax-Cmin} + 4 \quad \text{if (b == max)} \tag{4.47}$$

$$H = T * 60 \tag{4.48}$$

4- Calculation of **S**aturation:

If (Cmax != 0):

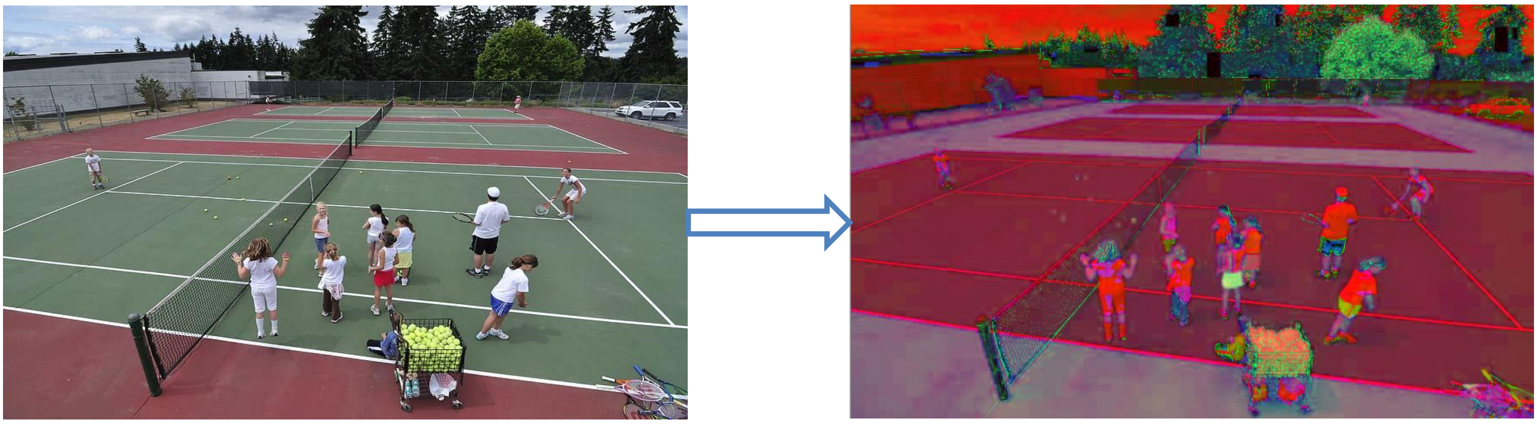
$$S = \frac{delta}{Cmax} \tag{4.49}$$

Else:

$$S = 0 \tag{4.50}$$

5- Calculation of **V**alue:

$$V = Cmax \tag{4.51}$$



a b

Figure 34 RGB conversion to HSV: a) image in the RGB color space, b) image in the HSV color space

We can also transfer colors from the HSV space system to the RGB space system (figure 35) using the following equations:

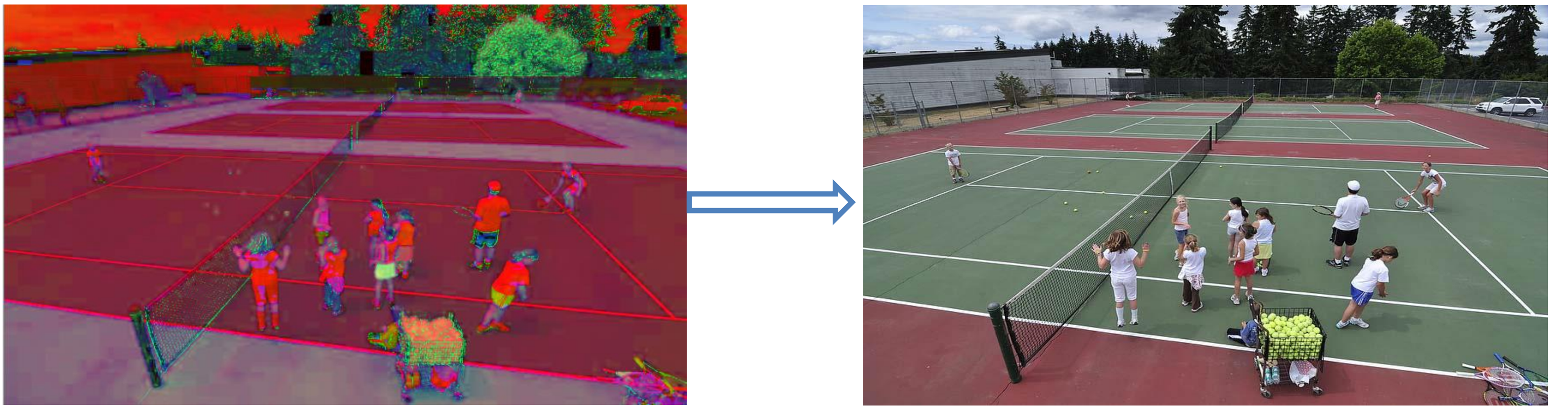
$$C = C * S \tag{4.52}$$

$$x = C * (1 - |\left(\frac{H}{60}\right) mod\ 2 - 1|) \tag{4.53}$$

$$m = V - C \tag{4.54}$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \leq H < 60^\circ \\ (X, C, 0) & , 60^\circ \leq H < 120^\circ \\ (0, C, X) & , 120^\circ \leq H < 180^\circ \\ (0, X, C) & , 180^\circ \leq H < 240^\circ \\ (X, 0, C) & , 240^\circ \leq H < 300^\circ \\ (C, 0, X) & , 300^\circ \leq H < 360^\circ \end{cases}$$

$$(R, G, B) = (R' + m, G' + m, B' + m) \tag{4.55}$$



a b

Figure 35 HSV conversion to RGB: a) image in the HSV color space, b) image in the RGB color space

4.2 Image Enhancement

The main goal of image enhancement is to improve the original image in order to obtain a more suitable result for the intended application. This approach can be applied under two essential categories: the frequency domain and the spatial domain. Whereas the latter includes the direct manipulation of image pixel values, such values cannot be accessed directly in the frequency domain. As a result, the Fourier transform is employed to convert pixel values to the frequency domain for frequency modification, with the inverse Fourier transform then applied to obtain the final image. During image enhancement we can obtain light images from dark images and vice versa, as well as change the image contrast. This popular function is used to reduce image noise, which has a direct effect on image processing applications such as edge detection and region detection. The method also assists in the perception of visual images by changing various properties as illustrated below figure:

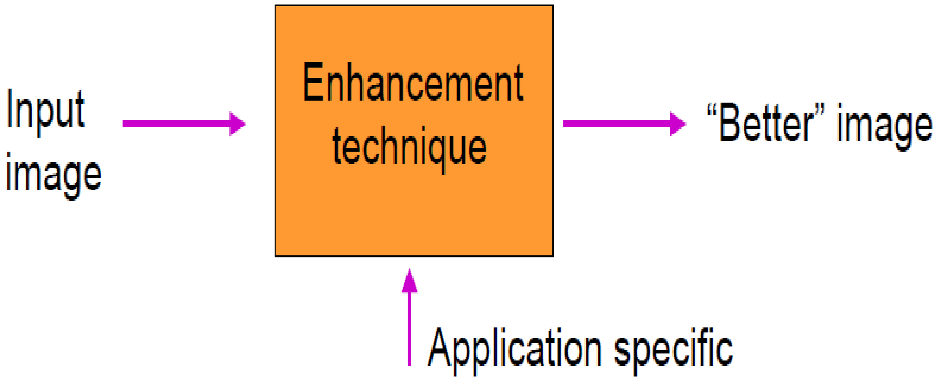


Figure 36 Block diagram of image enhancement

As outlined previously, image enhancement using a spatial domain approach is carried out directly, and thus can be expressed as follows:

$$g\ (x,y) = T\quad[\quad f(x,y)\quad] \tag{4.56}$$

where 'f' refers to the original image (or set of images), ‘g’ refers to the image after enhancement, and ‘T’ refers to the function applied to image ‘f’.

4.2.1 Gray Image Transformation

A variety of gray transformation categories are available, as shown in Figure 37. Here 'r' refers to the pixel value before processing and's' refers to the pixel value after processing.

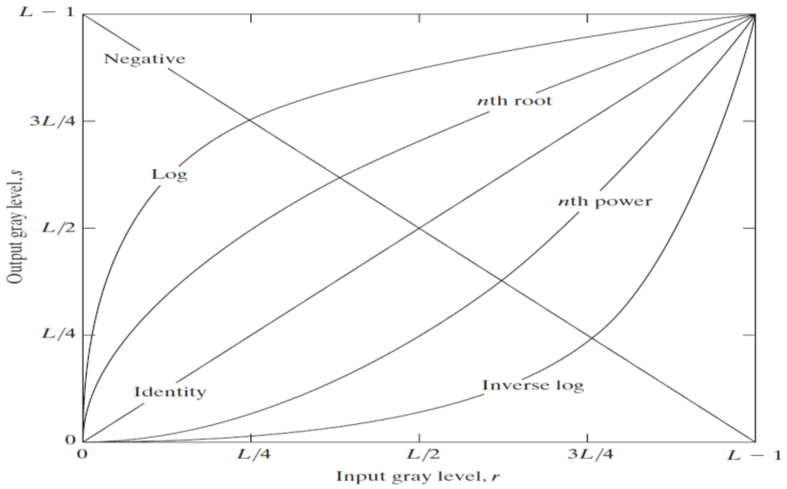


Figure 37 Basic intensity transformation functions

In the above figure 'r' refers to the original image, ‘s’ refers to the image after applying the enhancement functions, and 'L' is the range of pixel values in gray level "L = 256".

4.2.1.1 Gray Image Negatives

This transformation is applied to the gray space that has a color range from [0 to L-1], where 'L' equals 255 (as illustrated in figure 38), and is obtained via the following expression:

$$s = L - 1 - r \tag{4.57}$$

This process effectively works by reversing the value of pixel "intensity". As a result, dark regions, especially black, are dominant in size. This method is commonly employed in air force planes.

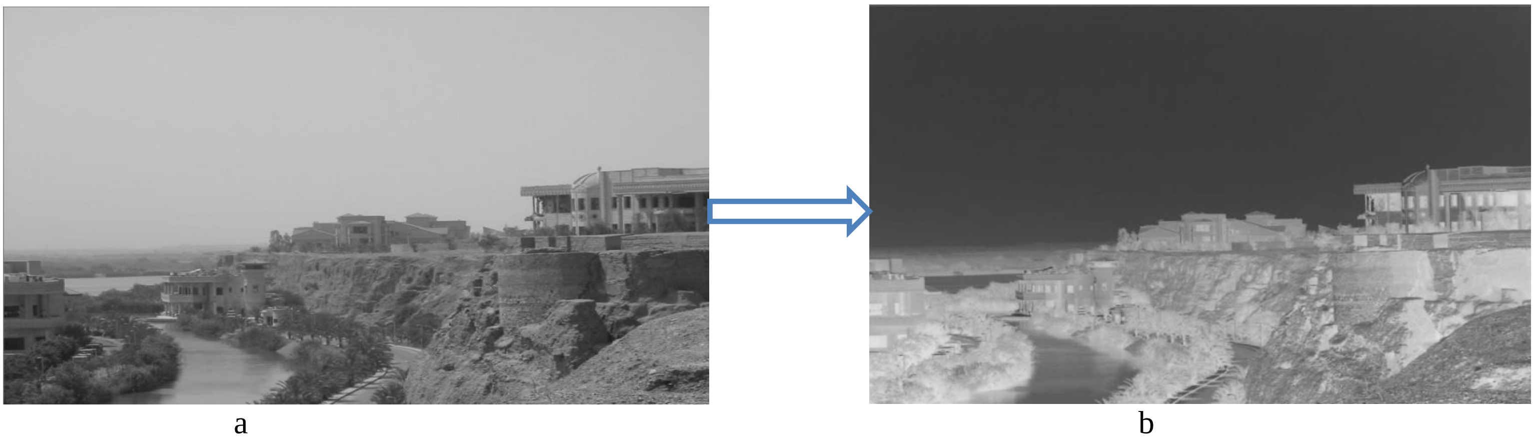


Figure 38 Gray scale inversion: a) image in gray scale, b) the inverted image

4.2.1.2 Log Transformation

In this method a narrow range of low-level gray space intensities in the original image are transformed into a broad range in the final image, and vice versa, i.e., a wide range of high-intensity values in the original image are transformed into a narrow range in the final image. This transformation thus increases the pixel values of dark regions and compresses the pixel values of light regions as illustrated below figure. The inverse-log transform will do the opposite. This transformation can be applied via the following expression:

$$s = c * log(1 + r) \tag{4.58}$$

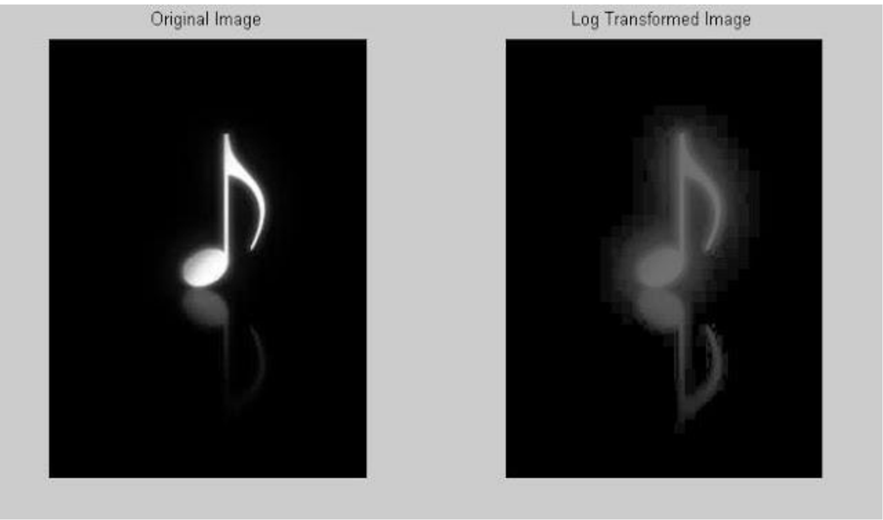


Figure 39 LOG enhancement image, c=20

4.2.1.3 Power-Law Transformation

This transformation, as shown in Figure 40, is produced using the following expression:

$$S = cr^{y} \tag{4.59}$$

This transformation function is also known as gamma correction. For various values of γ, different levels of enhancement can be obtained. As shown in Figure 40, the value of 'γ' has a significant effect on enhancement. In Figure 41, variation in ' γ' produces the observed changes in pixel light intensity in the displayed images. The main difference between power-law transformation and log transformation is that the curves of the former can be obtained only by changing the value of 'γ'.

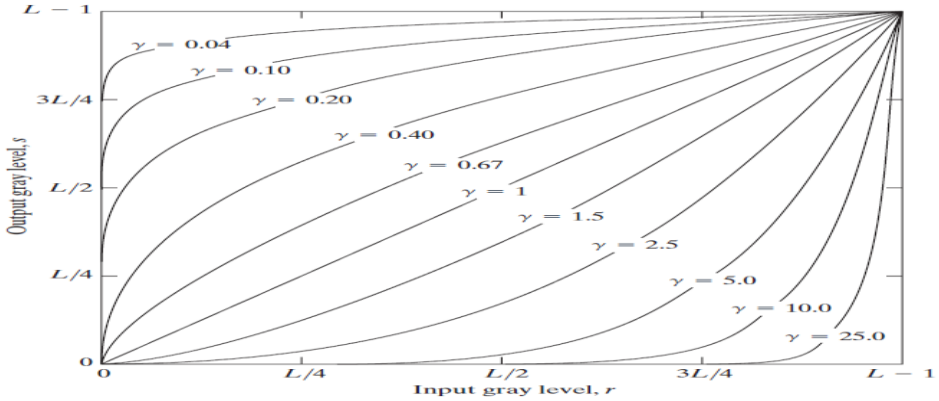


Figure 40 Plots of the power-law function when c=1 and for values of y between [0.04 – 25.0]

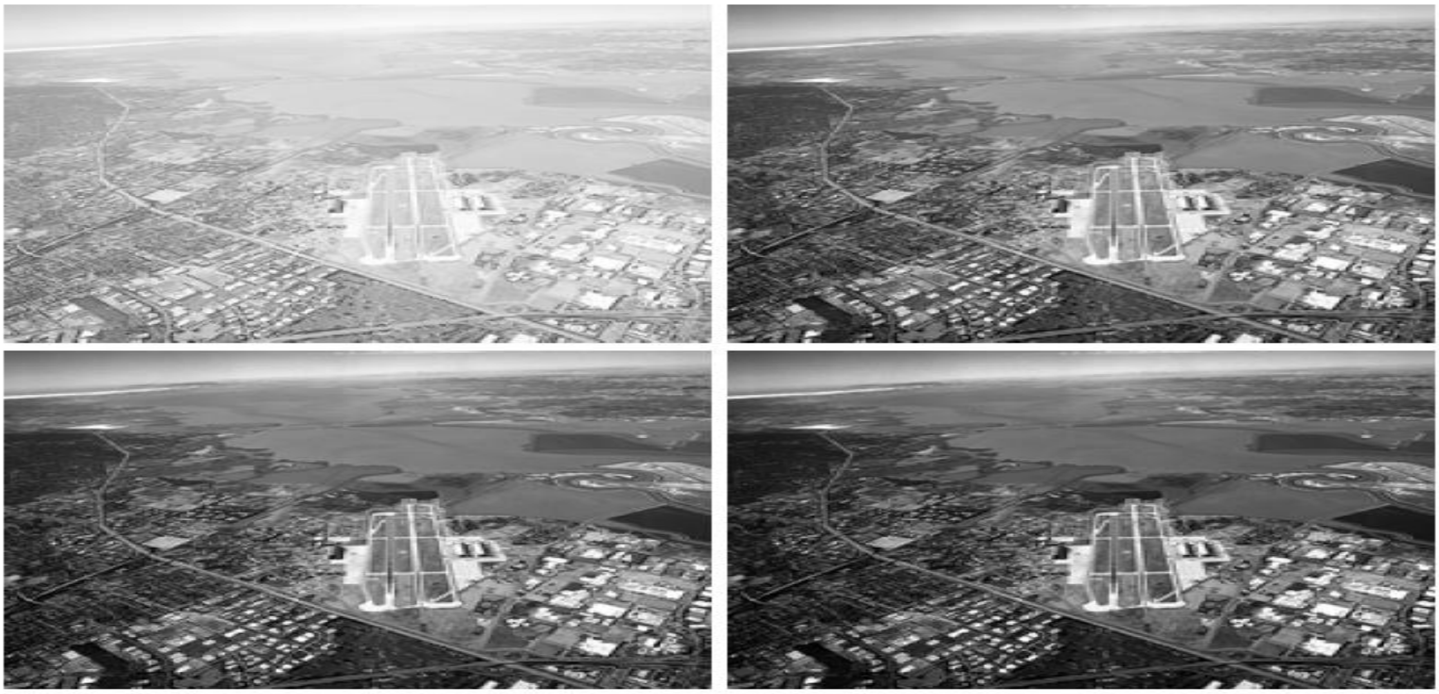


Figure 41 Example of power-law function application: a) original aerial image, b) result when y=3.0, c) result when y=4.0, d) result when y=5.0, for all c=1

4.2.1.4 Contrast Stretching

In this transformation the image is improved by stretching the area of pixel intensity values. This process assists in understanding the content of an image, especially regarding objects inside dark or highly luminous regions. Contrast stretching is carried out using the following expression:

$$s = (r - c)\left[\frac{b - a}{d - c}\right] + a \qquad (4.60)$$

In the above equation, 'a' and 'b' represent the assigned limits of pixel intensity values in the final image (in the gray space system the value of 'a' is usually equal to zero and the value of 'b' is equal to two hundred and fifty five).

The letters 'c' and'd' respectively represent the lower and upper limit of pixel intensity values in the original image, and can be obtained via a histogram. If the value of 'a' is equal to the value of 'c', and the value of 'b' is equal to the value of 'd', the final image will be the same as the original image. The main obstacle facing this transformation method is that outlier values can reduce its effectiveness. For example, consider an image with the dimensions 512*512, which thus contains a total of 262144 pixels. Let us assume we have one pixel with the value zero and one pixel with the value two hundred and fifty five, with the remaining pixel values between one hundred and one hundred and fifty; in such a scenario, contrast stretching cannot be applied and we should instead divide the image into blocks, with each individual block representing an image as illustrated below figure.

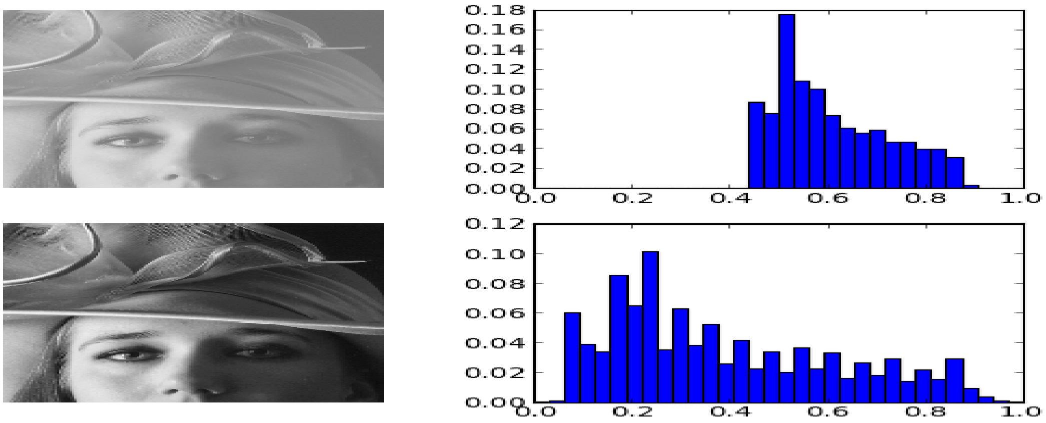


Figure 42 Example of contrast stretching

4.2.2 Spatial Domain Filtering

Spatial filtering involves the use of a pixel and its neighbors in order to select a new value for that pixel. The simplest type of spatial filtering is known as linear filtering. In this method a weight is attached to the pixels adjacent to the pixel of interest, with these weights then used to blend the pixels together to provide a new value for the pixel of interest. Linear filtering can be employed to smooth, blur, sharpen, or find the edges of an image.

4.2.2.1 Mean Filtering

Mean filtering is very important at the pre-processing stage because it involves smoothing local variations and blurring the image (figure 44), which in turn reduces image noise. In this case the filter is a window with size (m,n), where 'm' and 'n' are odd integers (figure 43). This window is used to corrupt the original image in order to calculate the average value in the area covered by the window. The mean filter equation is expressed as follows:

$$\mathrm{R(x,y)} = \frac{1}{N\sum_{k=1}^{m}\sum_{l=1}^{n} W(k,l)O(x-k,y-l)} \tag{4.61}$$

O: is the original image.

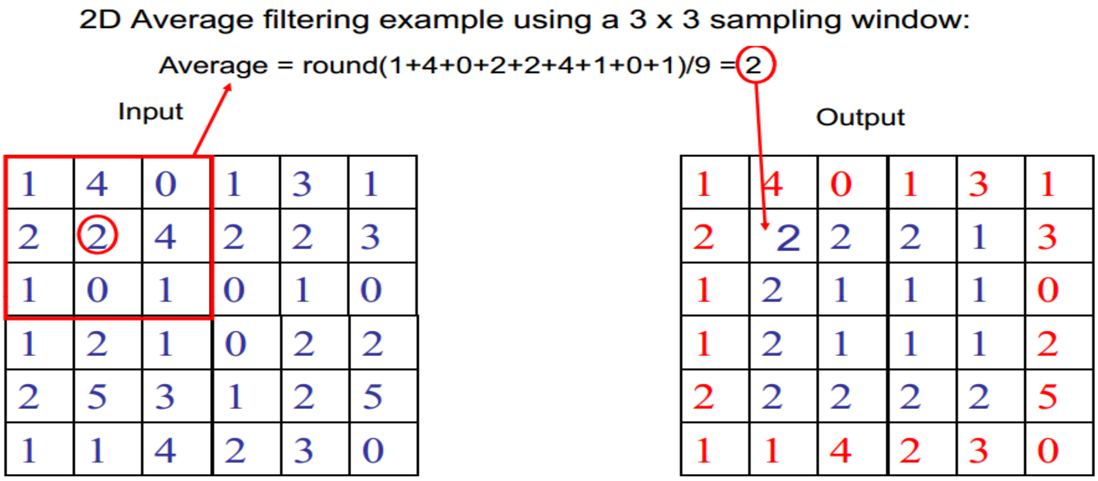
W: is the filter window.

m, n: is the size of W.

R: is the result of applying W to O.

x,y: are the coordinates of the image point.

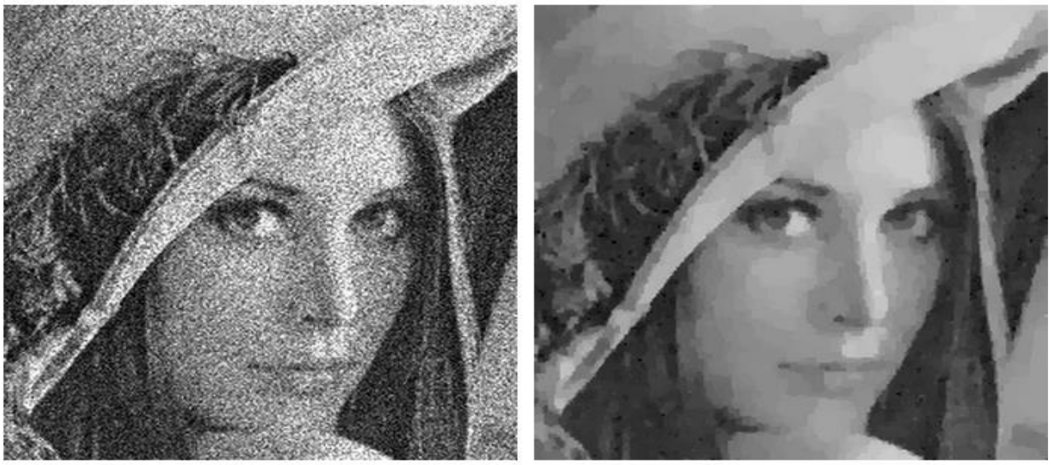
N: is the sum of values in W.


2D Average filtering example using a 3 x 3 sampling window:
Average = round(1+4+0+2+2+4+1+0+1)/9 = 2
Input
Output

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

1	4	0	1	3	1
2	2	2	2	1	3
1	2	1	1	1	0
1	2	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Figure 43 Mean filter applied in matrix form

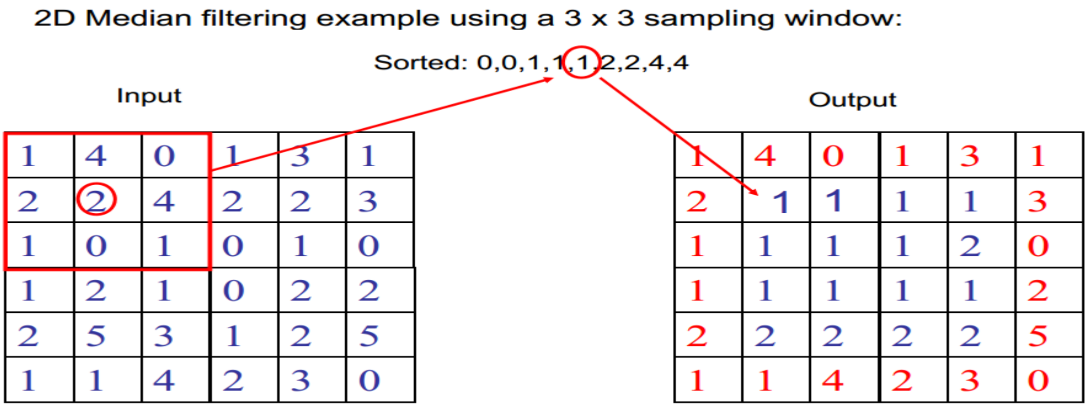


a b

Figure 44 Mean filter applied to an image: a) image with noise, b) image after mean filtering

4.2.2.2 Median Filtering

Median filtering is also used to reduce image noise, typically at the pre-processing stage prior to the application of the main processing function (especially those which are sensitive to noise) in order to improve the results of processes such as edge detection. The method is widely used as it is very effective at removing noise whilst simultaneously preserving edges, and is particularly useful for the removal of 'salt and pepper' type noise (figure 46). Median filtering works by using windows which are slid pixel by pixel over the entire image. The filter is calculated by first sorting all pixel values in the window into numerical order, then replacing the pixel under consideration with the middle pixel value as illustrated below figure:



2D Median filtering example using a 3 x 3 sampling window:

Sorted: 0,0,1,1,1,2,2,4,4

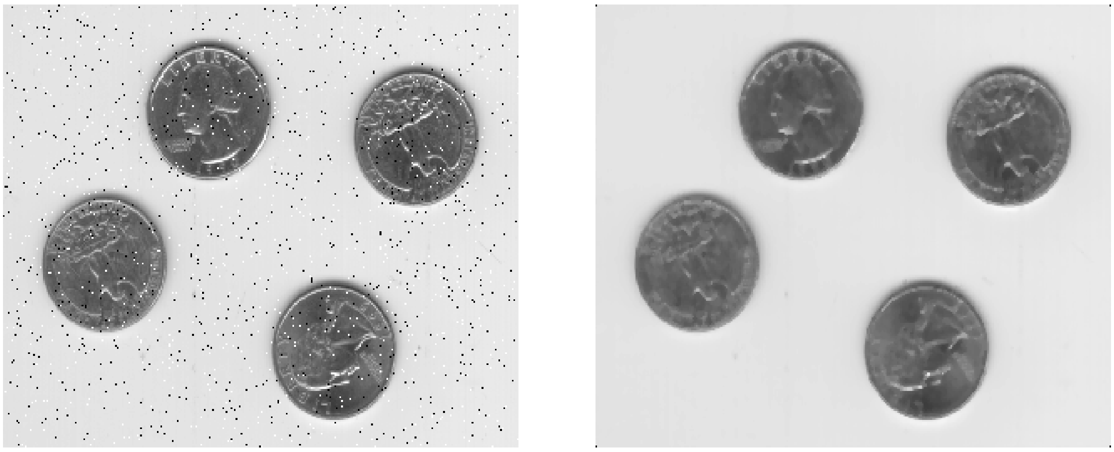
Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Figure 45 Median filter applied in matrix form



a b

Figure 46 Median filtering applied to an image: a) image with noise, b) image after median filtering

4.2.2.3 Gaussian Filtering

Gaussian filtering is a nonlinear digital filtering technique which is widely used to blur an image and thereby remove noise. The Gaussian filter works by moving through the image pixel by pixel and replacing each value with the average value of window pixels As in median filtering, this window slides pixel by pixel over the entire image, but in this case the filter is calculated using the Gaussian equation below (Table 6), which is then applied to all pixels in the image (figure 48).:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (4.61)$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Table 6 Gaussian Kernel When **σ**=1

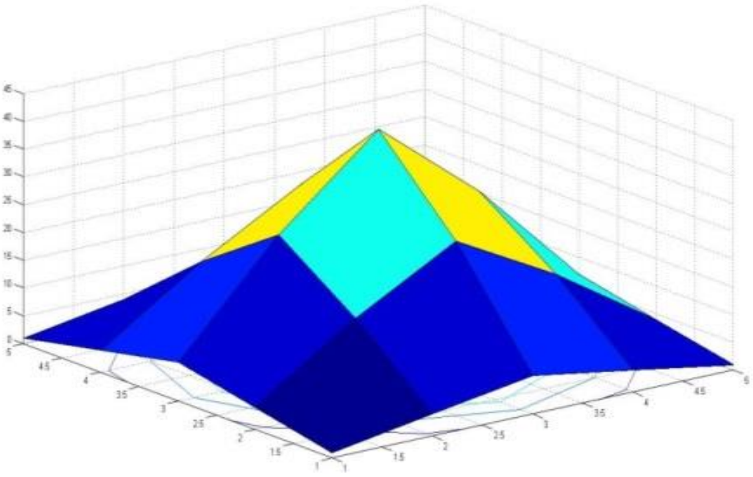
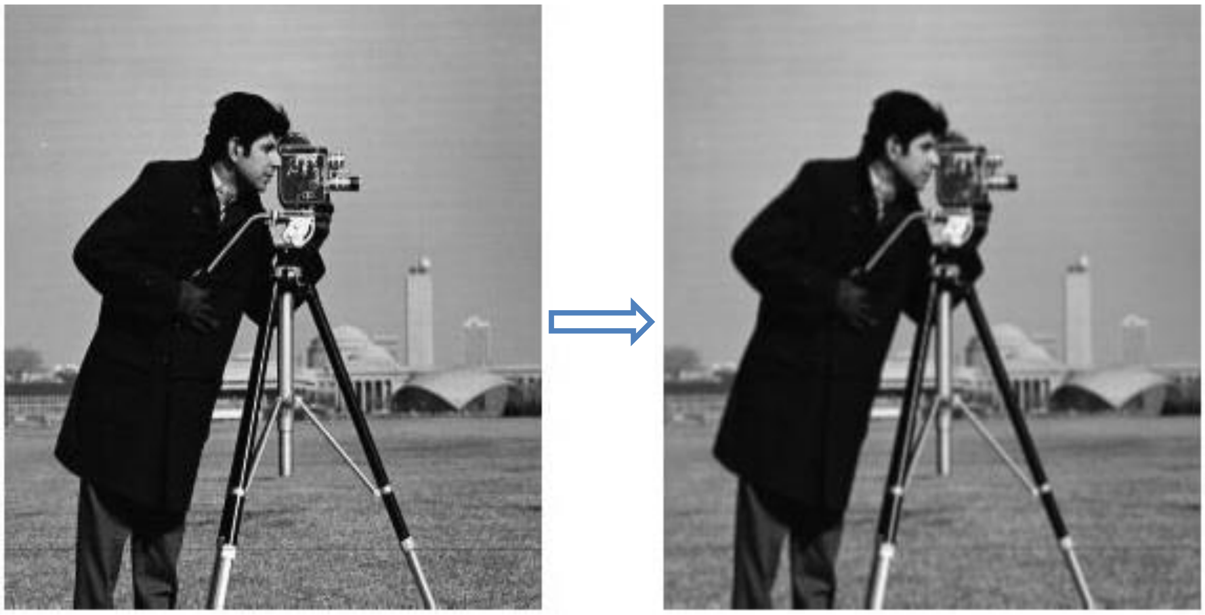


Figure 47 Plots of the Gaussian kernel when **σ=1**



a

b

Figure 48 Gaussian filtering: a) image with noise, b) image after Gaussian filtering

4.3 Edge Detection

Edge detection is used in image processing to detect the boundaries of regions inside an image, in the attempt to assign changes in brightness. The aim of edge detection is object extraction and image segmentation, such as the extraction of the foreground from the background. Edge detection has attracted the attention of many researchers and is considered one of the most important fields of study in lower level computer vision [27].

4.3.1 Canny Edge Detection

Developed by John F. Canny in 1986, canny edge detection involves the acquisition of a wide range of image boundaries via the use of multi-level processes [28]. Canny edge detection is aimed at best achieving the following standards:

1- Detection: find the largest possible number of real edge points and reduce the number of disingenuous edge points (figure 49).

2- Localization: the discovered edges should be as close as possible to genuine edges.

3- Number of responses: one genuine edge should not bring about more than one distinguished edge.

The algorithm runs in 5 separate steps:

1- **Smoothing**: using the Blurring filter on the image to reduce noise.

2- **Finding gradients**: edges should be marked where the gradients of the image are large in magnitude.

3- **Non-maximum suppression**: only local maxima should be marked as edges.

4- **Double thresholding**: potential edges are determined via thresholding.

5- **Edge tracking via hysteresis**: final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

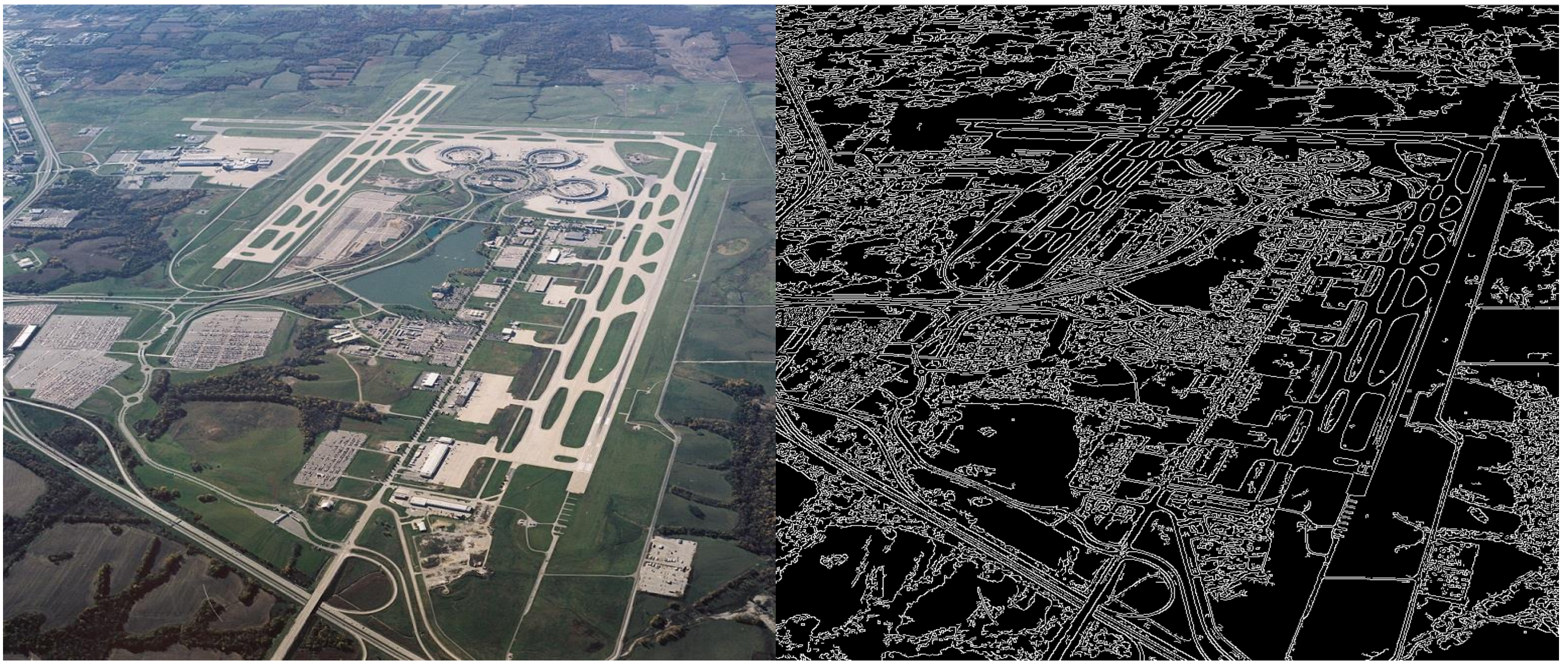


Figure 49 Example of Canny edge detection

4.3.2 Sobel Edge Detection

The Sobel operator is a discrete differential operator which utilizes two 3x3 kernels (figure 51); one kernel estimates the gradient in the x-direction, and the other estimates the gradient in the y-direction, as follows:

$$GX = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{4.62}$$

$$GY = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{4.63}$$

The image is convolved with both kernels to approximate the derivatives of horizontal and vertical change. At each given point, the gradient magnitude can be approximated as:

$$G = \sqrt{GX^2 + GY^2} \tag{4.64}$$

Due to the Sobel operator's smoothing effect (Gaussian smoothing), it is less sensitive to any noise present in images. However, this smoothing affects the accuracy of edge detection. In other words, although the Sobel method does not produce an image with sufficient accuracy for edge detection, it is adequate for use in numerous other applications as illustrated below figure:

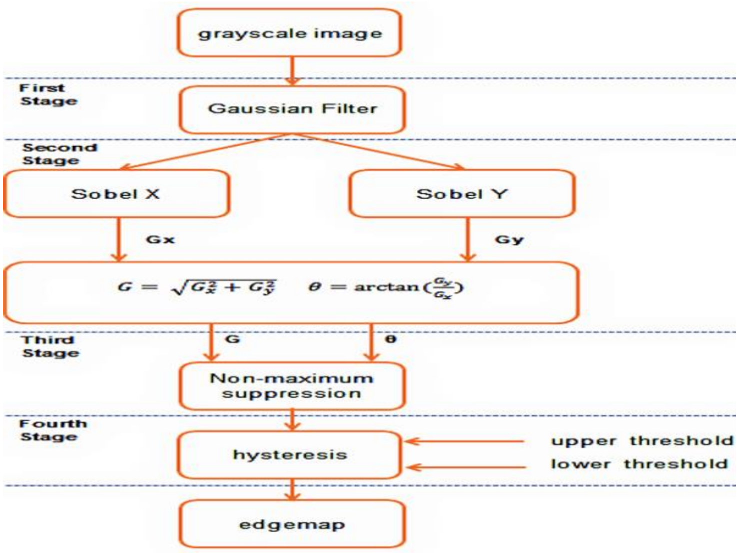


Figure 50 Block diagram of the Sobel edge detection algorithm

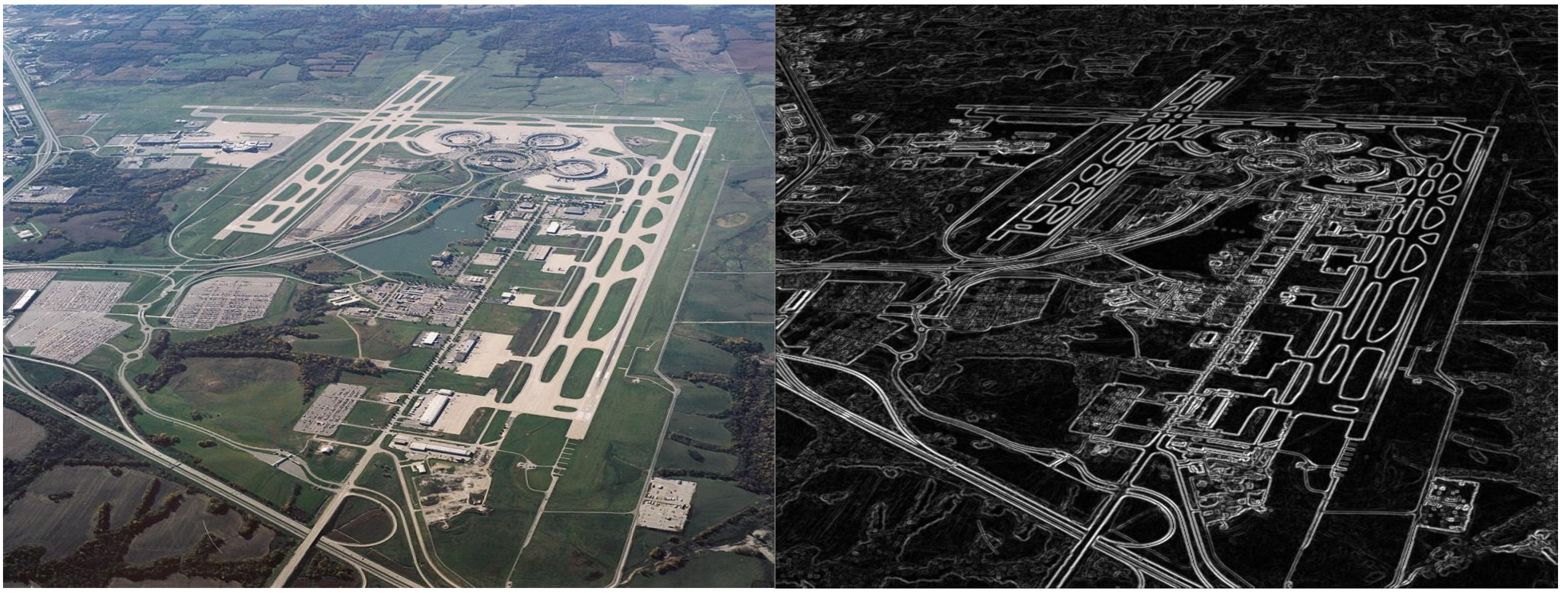


Figure 51 Example of Sobel edge detection

4.3.3 Prewitt Edge Detection

The Prewitt edge filter is used to detect edges based on the application of a horizontal and a vertical filter in sequence as illustrated figure 52. Both filters are applied to the image and summed to form the final result. The Prewitt operator produces values that are symmetric around the center (x,y) coordinates. The two filters are basic convolution filters which take the following form:

$$GX = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \tag{4.65}$$

$$GY = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \tag{4.66}$$

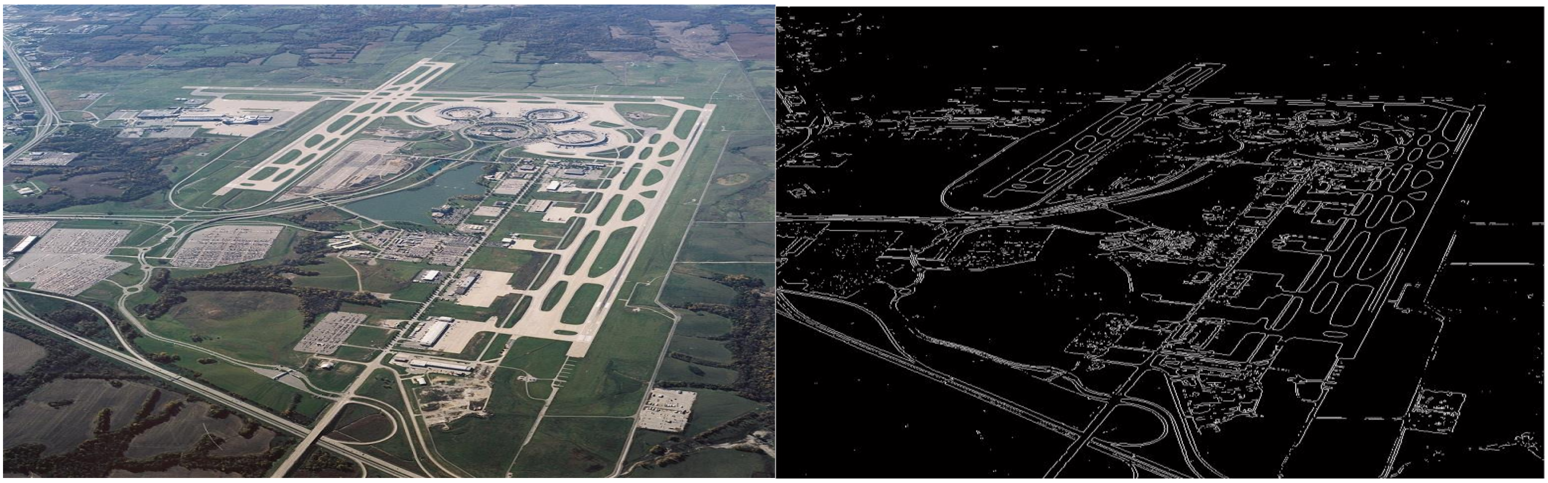


Figure 52 Example of Prewitt edge detection

4.4 Region Segmentation

Image segmentation is a complementary approach to the edge detection methods outlined above. Whereas in edge detection we segment an image by identifying the object boundaries and variation in image intensity, region segmentation involves identifying regions occupied by objects. Pixels are grouped into regions of similar properties.

4.4.1 Simple Segmentation

In simple cases where the image contains one object in the gray level, we can convert it into a corresponding binary image in which the object pixels are denoted as 1's and the background pixels as 0's. To determine a binary image we need to use a threshold value T, as follows:

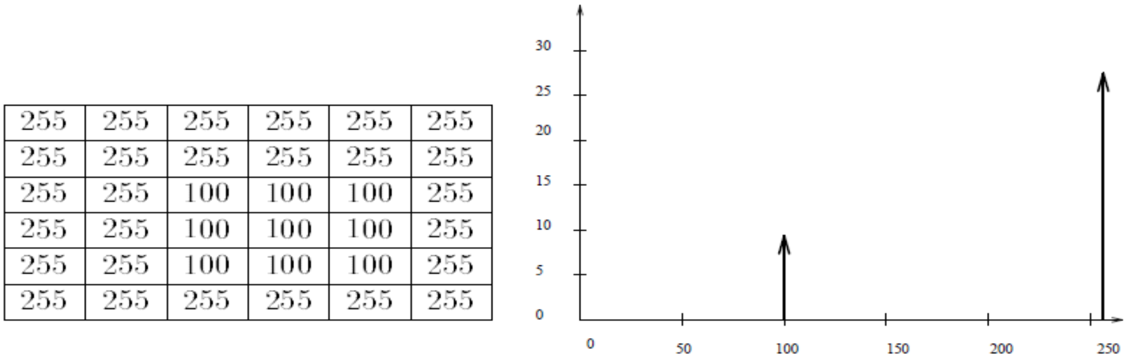
$$B(x,y) = \begin{cases} 1, & if\ \ F(x,y) > T \\ 0, & if\ F(x,y) \le T \end{cases} \tag{4.67}$$

where 'B' is a binary image, 'F' is a gray image, and 'T' is a threshold. Occasionally we need to use two thresholds to convert an image from the gray level to the binary level as follows:

$$B(x,y) = \begin{cases} 1, & if\ \ T1 < F(x,y) < T2 \\ 0, & otherwise \end{cases} \tag{4.68}$$

4.4.1.1 Thresholds and Histograms

The distribution of gray levels can be used to determine the threshold used in binary images as illustrated below figure:



255	255	255	255	255	255
255	255	255	255	255	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	255	255	255	255

a

b

Figure 53 Histograms: a) simple synthetic image with one projection, b) histogram of image shown in (a).

The histogram of a synthetic image will contain distinct spikes. When such a histogram has two spikes, it can be considered bimodal. In contrast, the histogram of a real image may not contain clear spikes but instead may consist of peaks and valleys. This pattern is the result of gray levels at the edges changing gradually from background to foreground as illustrated below figure:

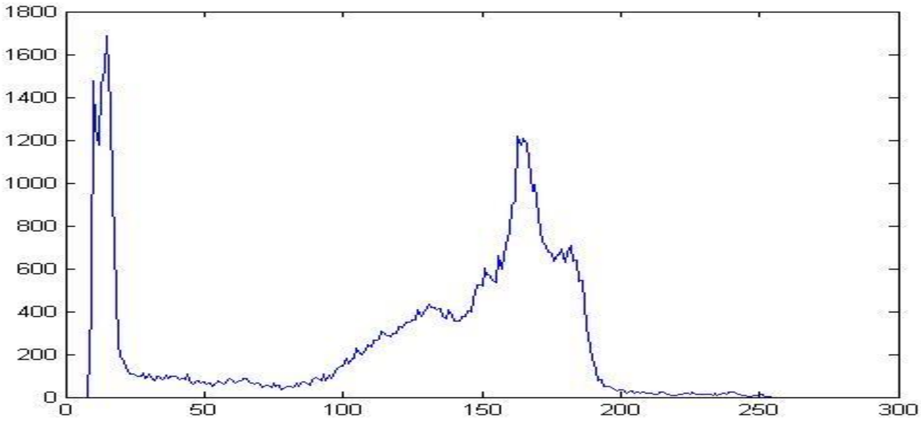
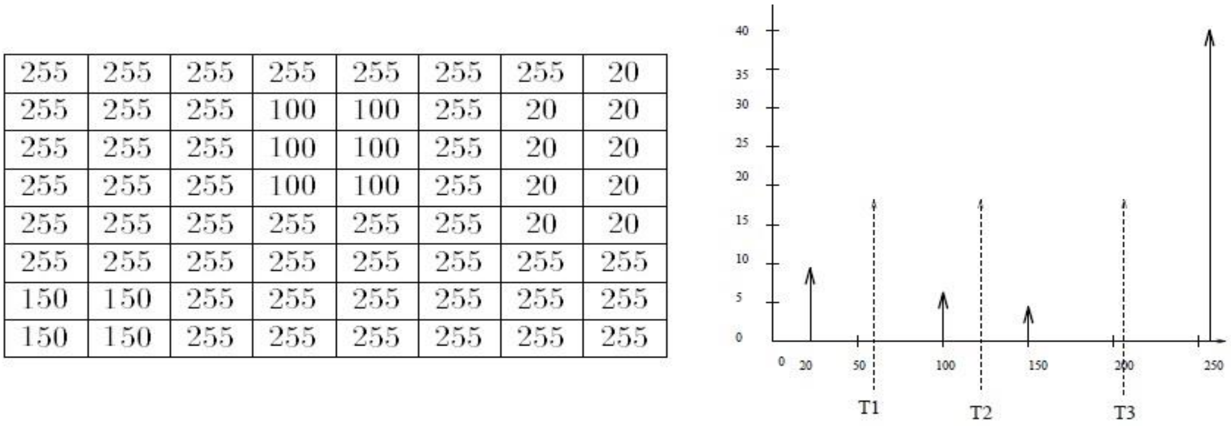


Figure 54 Example of a bimodal histogram

Objects with approximately the same range of gray levels are grouped to form a class. The histogram of an image will therefore include a peak for each class of objects and one large peak corresponding to the background as illustrated below figure:



255	255	255	255	255	255	255	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	255	255	255	20	20
255	255	255	255	255	255	255	255
150	150	255	255	255	255	255	255
150	150	255	255	255	255	255	255

a

b

Figure 55 Simple thresholds: a) image consisting of three objects, b) histogram of image shown in (a)

$$B1(x,y) = \begin{cases} 1, & if \;\; 0 < F(x,y) < T1 \\ 0, & otherwise \end{cases} \tag{4.69}$$

$$B2(x,y) = \begin{cases} 1, & if\ T1 < F(x,y) < T2 \\ 0, & otherwise \end{cases} \qquad (4.70)$$

$$B3(x,y) = \begin{cases} 1, & if\ T2 < F(x,y) < T3 \\ 0, & otherwise \end{cases} \qquad (4.71)$$

It should be noted that the histogram can only distinguish between classes of objects; it does not contain any spatial information. As a result, a "chess board" image comprising an equal number of alternating black and white blocks, and an image comprising random black and white dots of equal distribution will produce the same histogram.

4.4.1.2 Peakiness Test

The histogram of a real image may contain small peaks due to noise. Therefore, not all peaks can be used in segmentation. Before histogram peaks can be used in segmentation, we need to identify genuine peaks which correspond to object regions. We will use the peakiness test for that purpose as illustrated below figure:

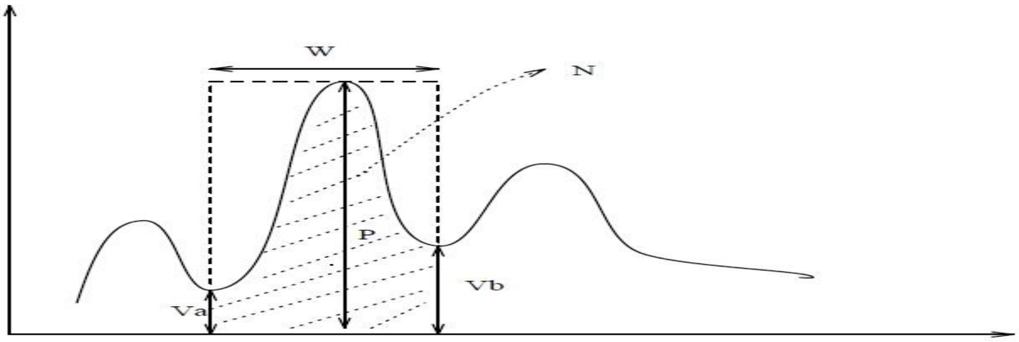


Figure 56 Peakiness test. V_a, V_b are the heights of valleys, W is width of peak, P is the height of peak, N is the number of pixels between V_a and V_b.

A peak can be considered 'good' if it is sharp and deep. A peak is sharp if the area under it is small. Peak sharpness can be expressed as the ratio of the area of the rectangle enclosing the peak to the number of pixels N under the peak, with peak depth the relative height of the peak. The peakiness test uses the following information:

1- W : the width of the peak in terms of gray level range from valley to valley.

2- P : height of the peak.
3- Va,Vb : the two valley points either side of the peak.
4- N : the number of image pixels covered by the peak.

The sharpness of a peak can be defined via the ratio:

$$Sharpness = N / (P * W) \tag{4.72}$$

The maximum ratio value of 1 represents the worst possible case, with peak sharpness increasing as the value decreases toward zero. The ratio of valley height to peak height is:

$$Valleys\_To\_Peak = (Va + Vb) / 2P \tag{4.73}$$

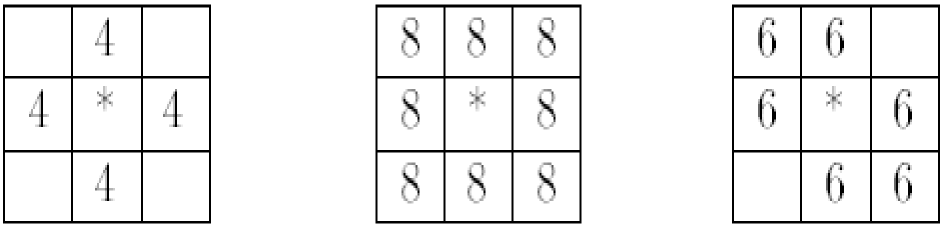
The actual peakiness test value is thus the product of the two ratios above:

$$Peakiness\_Test = (1 - Valleys\_To\_Peak) * (1 - Sharpness) \tag{4.74}$$

If the obtained peakiness value is greater than some threshold, then that peak can be used for segmentation.

4.4.2 Connected Component Algorithms

In order to find a connected group of pixels in an image we need to apply a connected component algorithm. There are three possible degrees of connectedness as illustrated below figure:



a b c

Figure 57 Pixel connectedness: a) 4-connected, b) 8-connected, c) 6-connected

In 4-connectedness, a pixel is considered with respect to four of its immediate neighbors (Left, Right, Up, Down), because these pixels are located at a distance of one from the original pixel. In 8-connectedness the four diagonally adjacent pixels are also considered. The following two subsections contain a discussion of two algorithms used to find connected components in an image: recursive and sequential.

4.4.2.1 Recursive Algorithm

The recursive algorithm works well and is easy to implement. However, being recursive, when run on a small computer with a limited stack this algorithm may easily run into stack overflow, which therefore must be taken care of by the programmer. For example, for a 512 * 512 image comprising a quarter of a million pixels, the recursive algorithm may have several thousand recursive calls. The algorithm involves the following steps:

1- Scan the binary image from left to right, top to bottom.
2- If an unlabeled pixel is found with a value of '1', assign a new label to it.
3- Recursively check the neighbors of the pixel in step 2 and assign the same label to these pixels if they are unlabeled with a value of '1'.
4- Stop when all pixels of value '1' have been labeled.

4.5.2.2 Sequential Algorithm

The sequential algorithm is a two-pass algorithm which labels image regions. The first pass scans the binary image and assigns any unlabeled pixel a new label. In the assignment of these labels the labels of neighboring pixels are also considered. During the second pass the labels of pixels are changed to the labels of their respective equivalence classes. The algorithm proceeds as follows:

1- Scan the binary image from left to right, top to bottom.
2- If an unlabeled pixel has a value of '1', assign a new label to it.
3- Determine the equivalence class labels.
4- In the second pass, assign the same label to all elements in each equivalence class.

4.5.3 Seed Segmentation

The first step of seed segmentation is to compute the histogram of an image. This histogram may contain a number of small peaks due to image noise and uneven illumination. During the second step, the histogram is smoothed to remove these small peaks by averaging over 3 elements. In the third step, candidate peaks and valleys are identified simply by detecting local maxima and minima in the histogram. ‘Good’ peaks are then detected via the peakiness test. In the fifth step, the image is segmented using thresholds at the valleys between peaks. The last step is carried out to determine a set of connected components in the image as illustrated below figure:

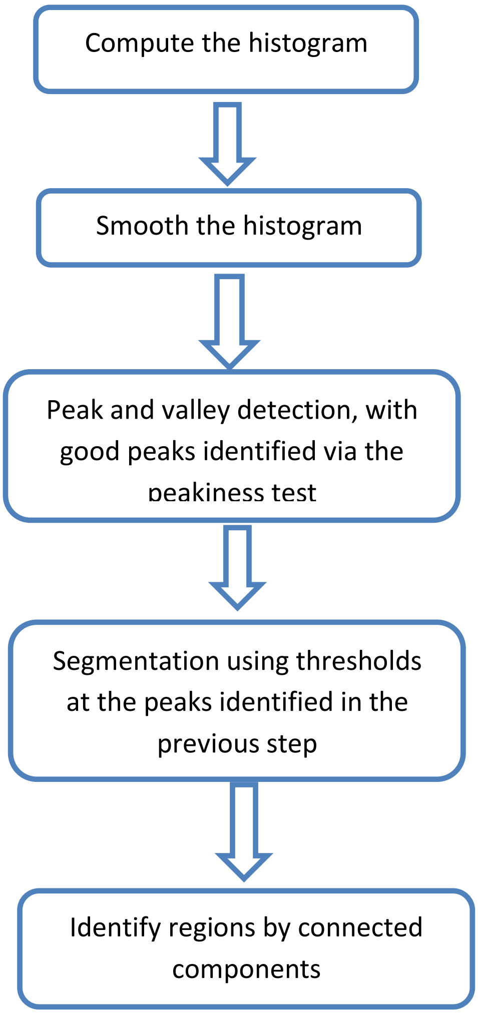

Compute the histogram
Smooth the histogram
Peak and valley detection, with good peaks identified via the peakiness test
Segmentation using thresholds at the peaks identified in the previous step
Identify regions by connected components

Figure 58 Block diagram of the seed segmentation algorithm

4.5 Template Matching

Template Matching is used to examine and detect the position of the image template in a target image. This method can generate a better result when applied only to image edges or to images in the gray space. Template matching is usually applied after choosing a region from the original image to use as the image template. In this case 'S' refers to the target image, (x, y) to the coordinate system of the pixels in the target image, 'T' to the image template, and (x_t, y_t) to the coordinate system of the pixels in the image template. We then simply move the center (or the origin) of the template $T(x_t, y_t)$ over each (x, y) point in the search image, and calculate the sum of the products of the coefficients in $S(x, y)$ and $T(x_t, y_t)$ over the whole area spanned by the template. As all possible positions of the template with respect to the search image are analyzed, the position with the highest score is considered the best position. This method is sometimes referred to as 'Linear Spatial Filtering', and the template a ‘filter mask’, as illustrated below figure:

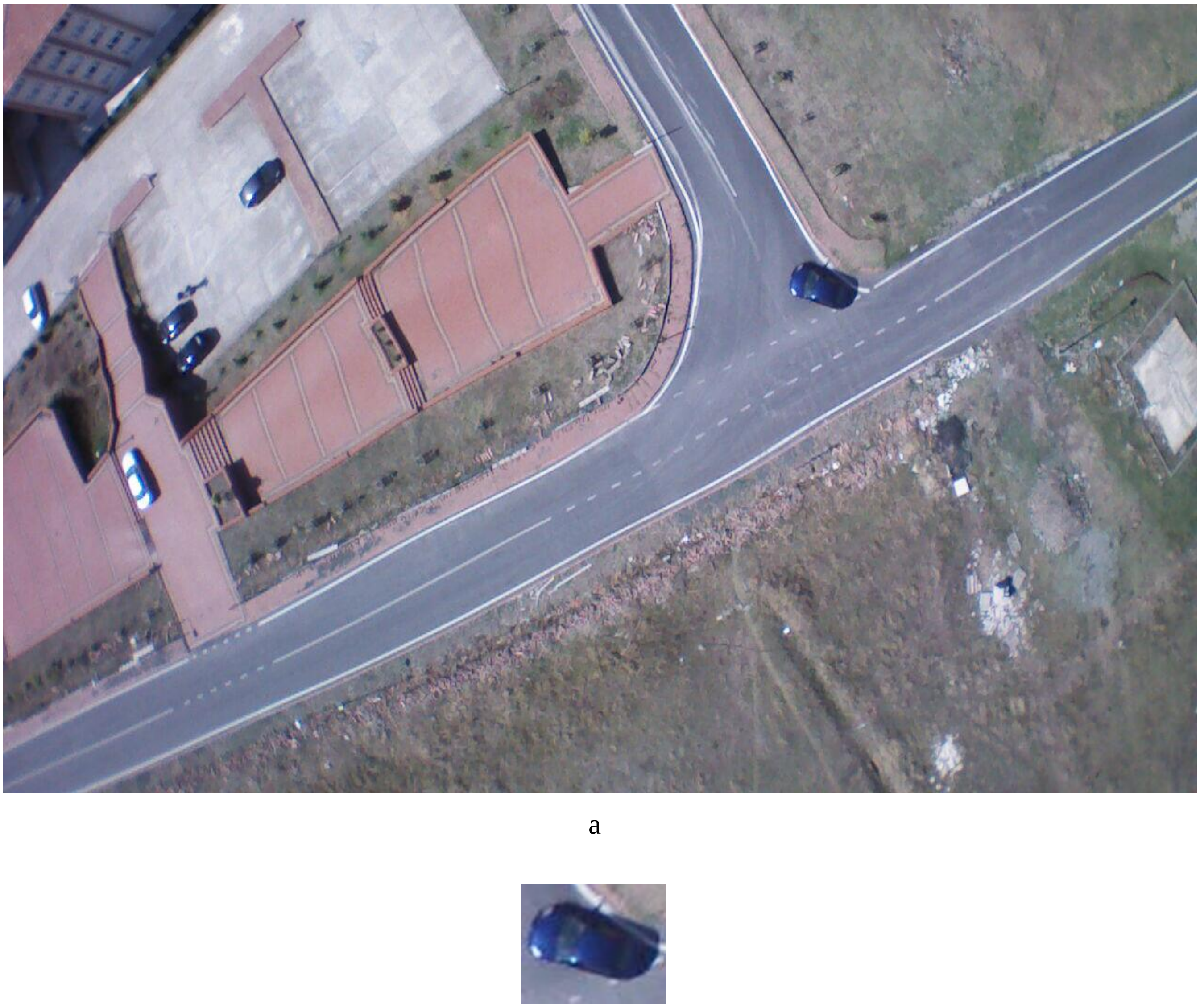


Figure 59 Template matching test: a) source image, b) target image

The following expressions are employed in the most famous template matching methods:

1- Squared difference

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \tag{4.75}$$

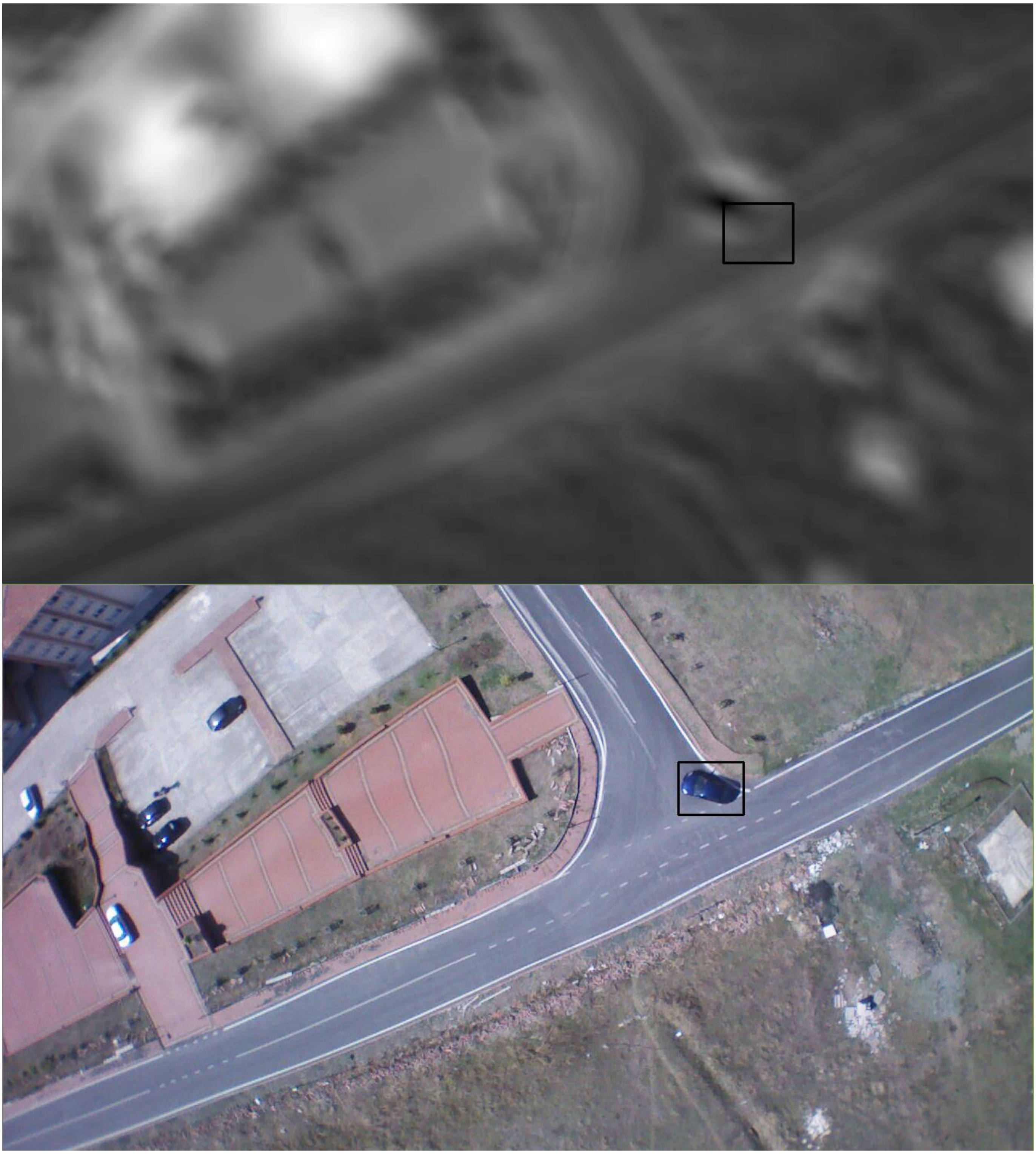


Figure 60 Example of squared difference application

2- Normalized squared difference

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{4.76}$$

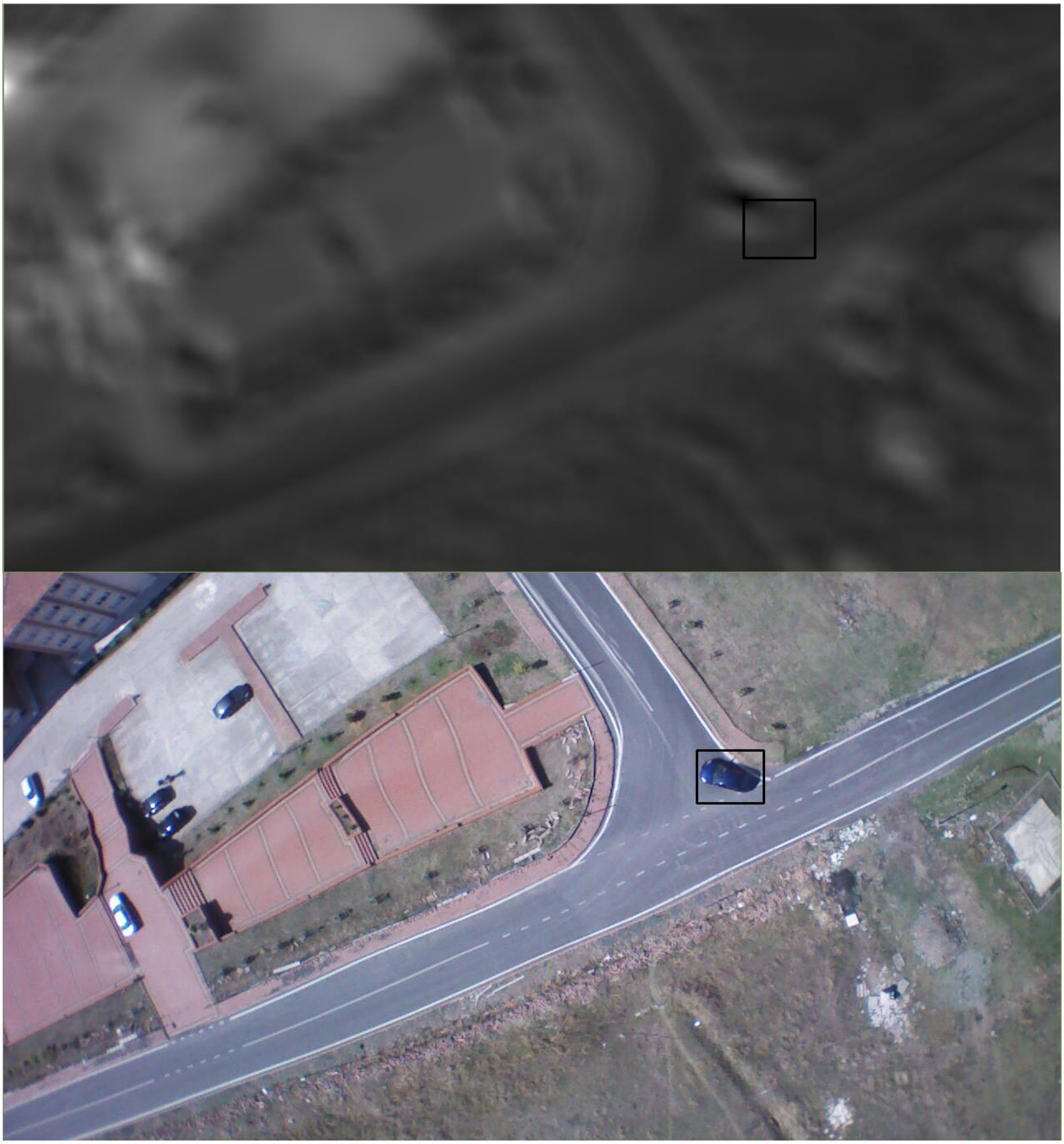


Figure 61 Example of normalized squared difference application

3- Cross-correlation

$$R(x,y) = \sum_{x',y'} \left(T(x',y') \cdot I(x+x', y+y') \right) \qquad (4.77)$$

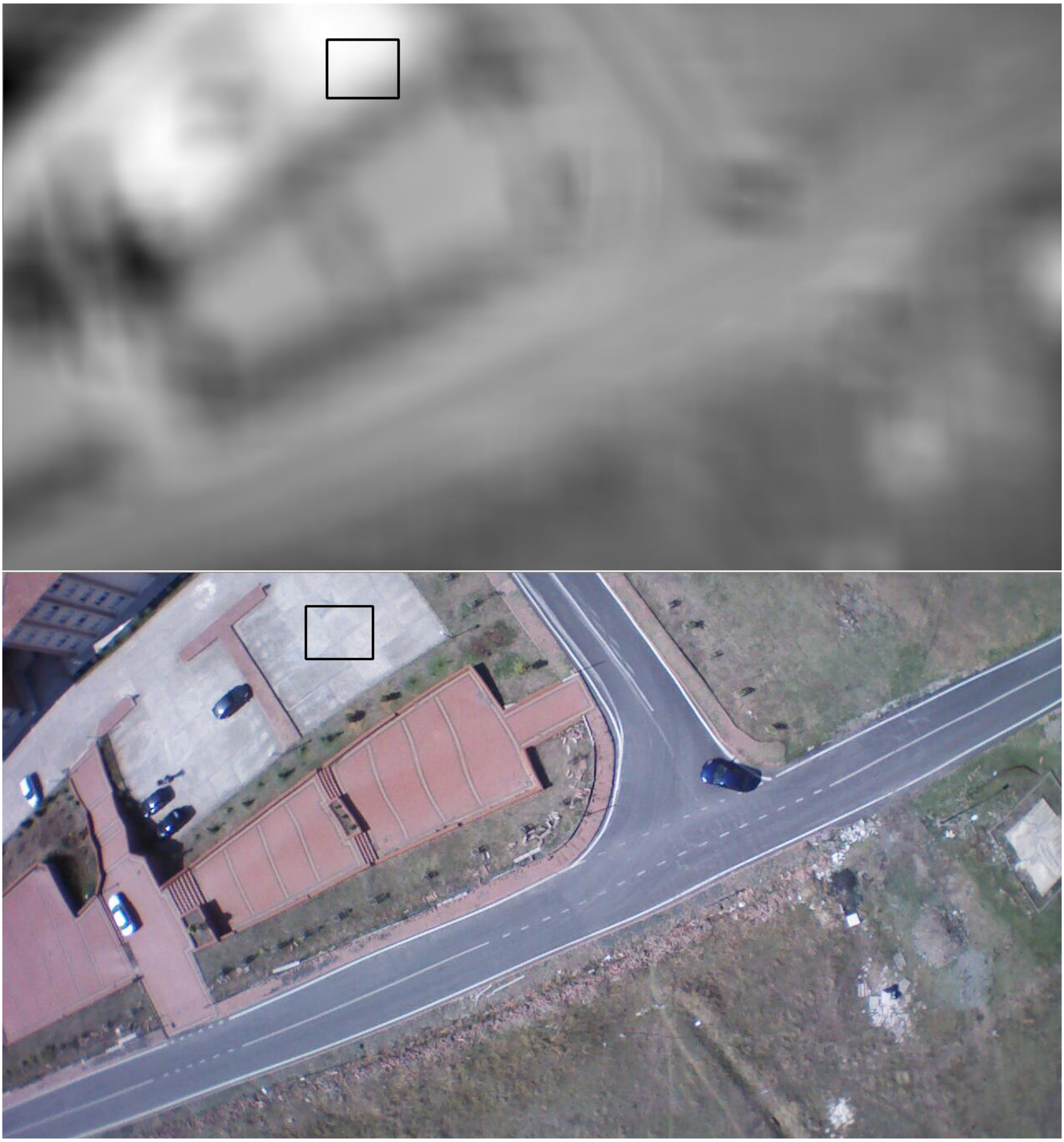


Figure 62 Example of cross-correlation application

4- Normalized cross-correlation

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \qquad (4.78)$$

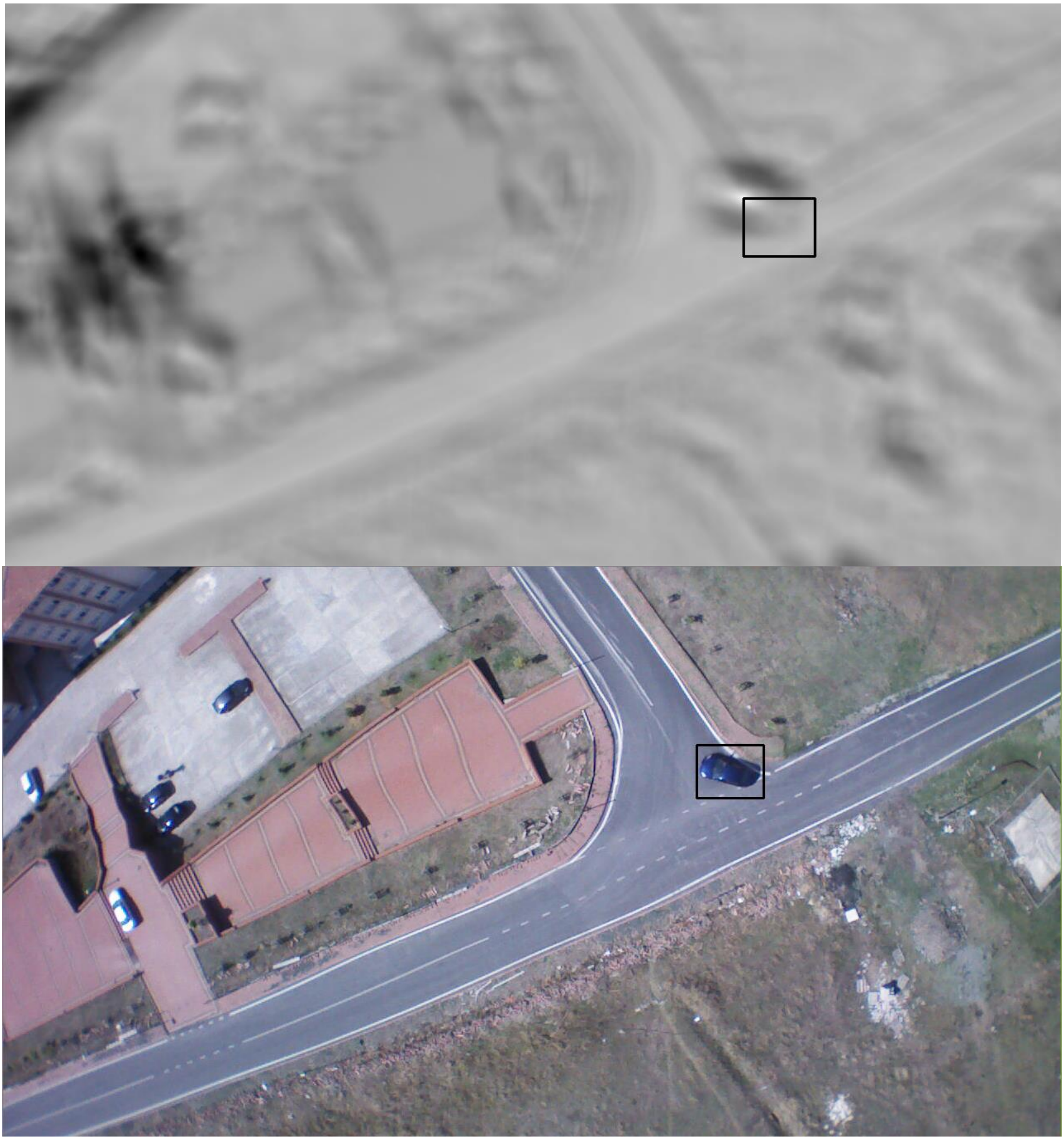


Figure 63 Example of normalized cross-correlation application

5- Correlation coefficient

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \textstyle\sum_{x'',y''} T(x'',y'') \tag{4.79}$$

$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \textstyle\sum_{x'',y''} I(x+x'',y+y'')$$

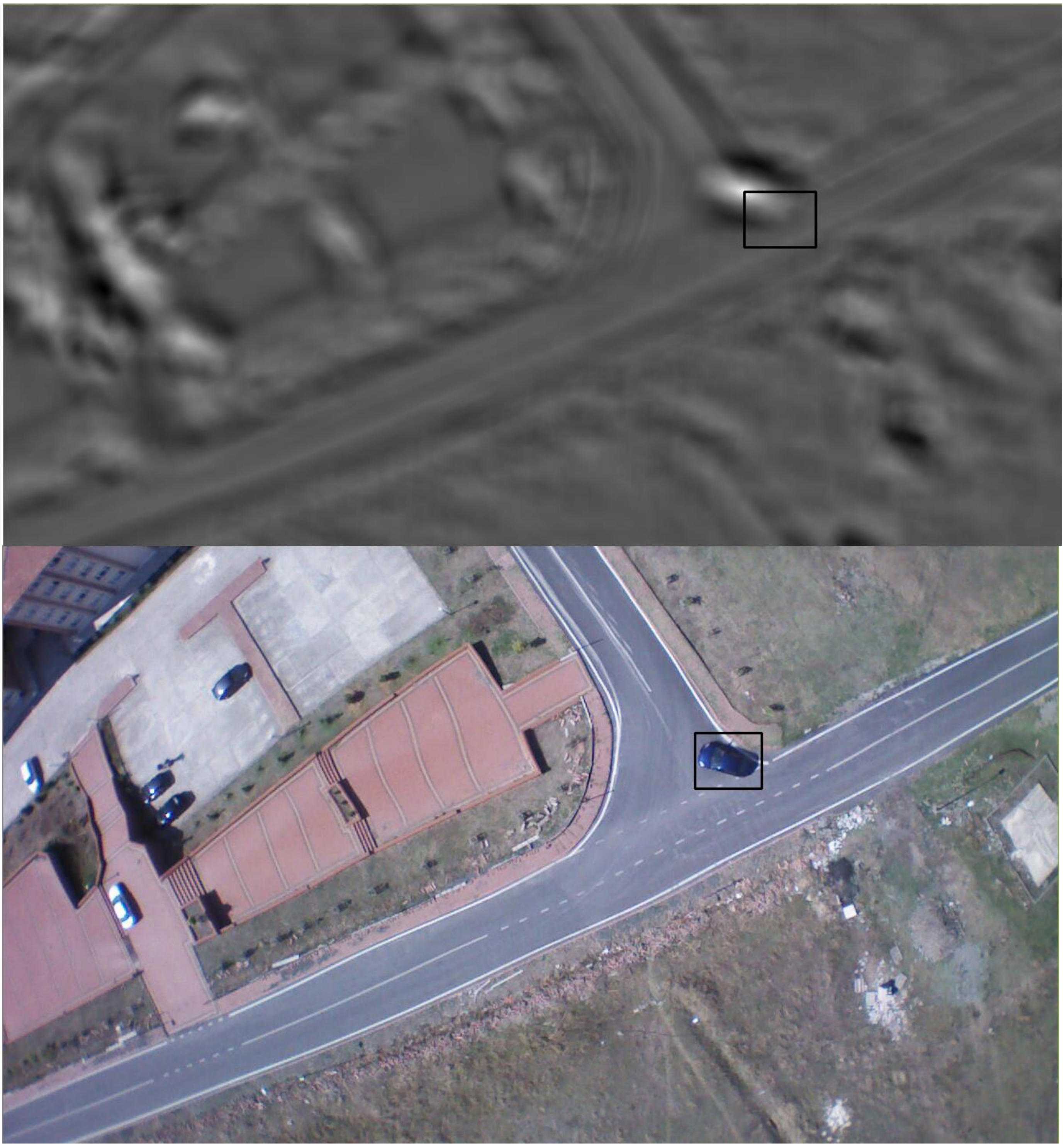


Figure 64 Example of correlation coefficient application

6- Normalized correlation coefficient (NCC - Fast normalized cross-correlation)

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{4.80}$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

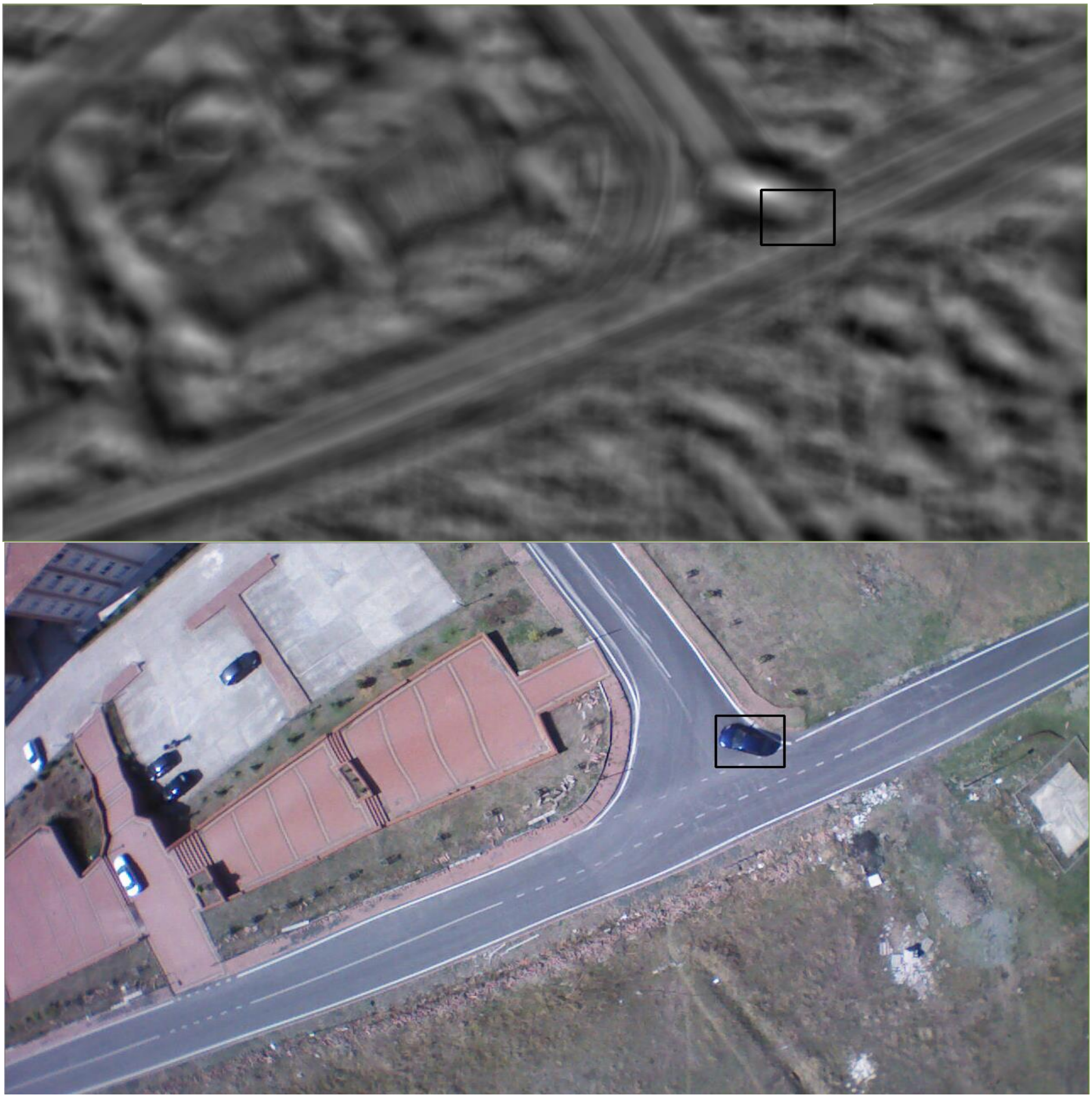


Figure 65 Example of NCC application

4.6 Integral Images

The summed area table or integral image is an effective method with which to calculate the sum of pixel values in a gray image. Originally developed in 1984, this method was not employed actively in the field of computer vision until 2001, when it was introduced by Viola as part of the "Viola-Jones Object Detection Framework". The goal of an integral image is to achieve the rapid computation of both box convolutions and intensities for any rectangle in an image which is not sensitive to rectangle size. In integral imaging, computation time is independent of filter size, thus allowing the use of any box filter application. An integral image is the same size as the original image, with its value at any point (x,y) being the sum of the intensity values for all points in the original image with locations less than or equal to (x,y) as illustrated below figure:

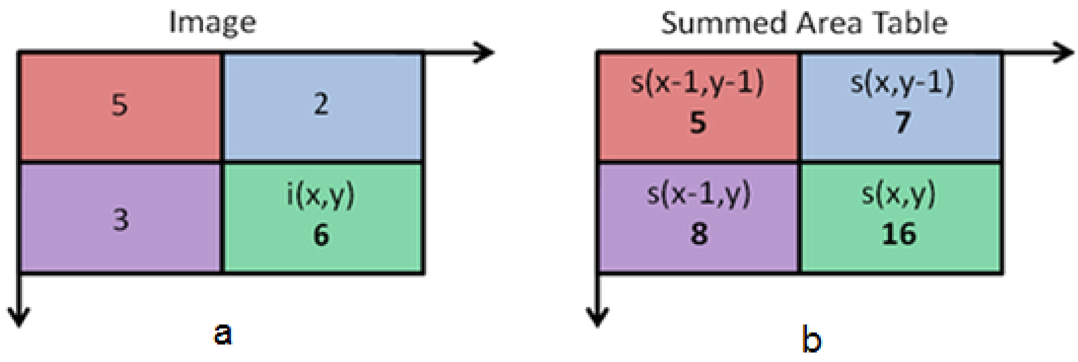


Figure 66 Example of an integral image: a) image, b) summed area table

$$S(x, y) = I(x, y) + S(x-1, y) + S(x, y-1) - S(x-1, y-1) \tag{4.81}$$

where ‘S’ is the integral image and ‘I’ is the original image. The image is then divided into four rectangles as follows:

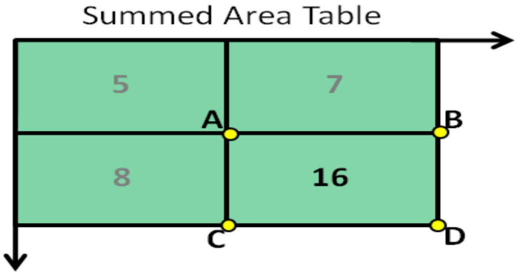


Figure 67 Summed Area Table

The sum of $I(x', y')$ is calculated using the following equation:

$$I(x', y') = S[A] + S[D] - S[B] - S[C] \tag{4.82}$$

The result of this equation for the above example is equal to 6.

4.7 Speeded Up Robust Features (SURF)

Originally developed in 2006 by Herbert Bay, the SURF (Speeded Up Robust Features) algorithm is a robust local feature detector which is useful for various computer vision functions, such as 3D reconstructions and object recognition. The SURF algorithm is derived from the SIFT algorithm, but is faster than the latter by several times and is more robust for use in image translation. As the SURF algorithm is based on the efficient construction of an integral image and a 2D Haar wavelet, it can be used to calculate the determinant of the Hessian for rapid blob detection. For features, the algorithm uses the sum of the Haar wavelet response around the point of interest. Again, these values can be computed with the aid of integral images [29].

SURF roadmap:

1- Find interest points (using the determinant of the Hessian matrix) as illustrated figure 68.

A Hessian matrix in two dimensions consists of 2*2 matrices containing the second-order partial derivatives as follows:

$$\begin{bmatrix} \frac{\partial I^2}{\partial x^2} & \frac{\partial I^2}{\partial x \partial y} \\ \frac{\partial I^2}{\partial y \partial x} & \frac{\partial I^2}{\partial y^2} \end{bmatrix}$$

As a symmetric matrix, for any square matrix the determinant of the matrix is the product of the eigenvalues. Therefore, for the Hessian matrix, the eigenvectors form an orthogonal basis showing the direction of image curve features (gradient): if both eigenvalues are positive, local minima; if both eigenvalues are negative, local maxima; and if the eigenvalues have mixed signs, the saddle point. Therefore, if the product of the eigenvalues is positive, then the latter were either both positive or both negative, and we are at a local extremum. Typically, some kind of threshold is applied to the determinant value so we only detect major features; we can also control the number of interest points in this way. The integral image is then used to obtain the sum of intensities for the square, by multiplying by the weight factor and adding the resulting sums for the box filter together after normalizing the filter. The matrix containing the thresholded determinants for a particular filter size is known as the 'blob response map'. This blob at location X=(x,y,σ):

$$\det(H_{approx}) = D_{xx}D_{yy} - (.9D_{xy})^2 \tag{4.83}$$

For a 9*9 matrix, when σ=1.2:

$$\sigma = (filtersize/9)*1.2 \tag{4.84}$$

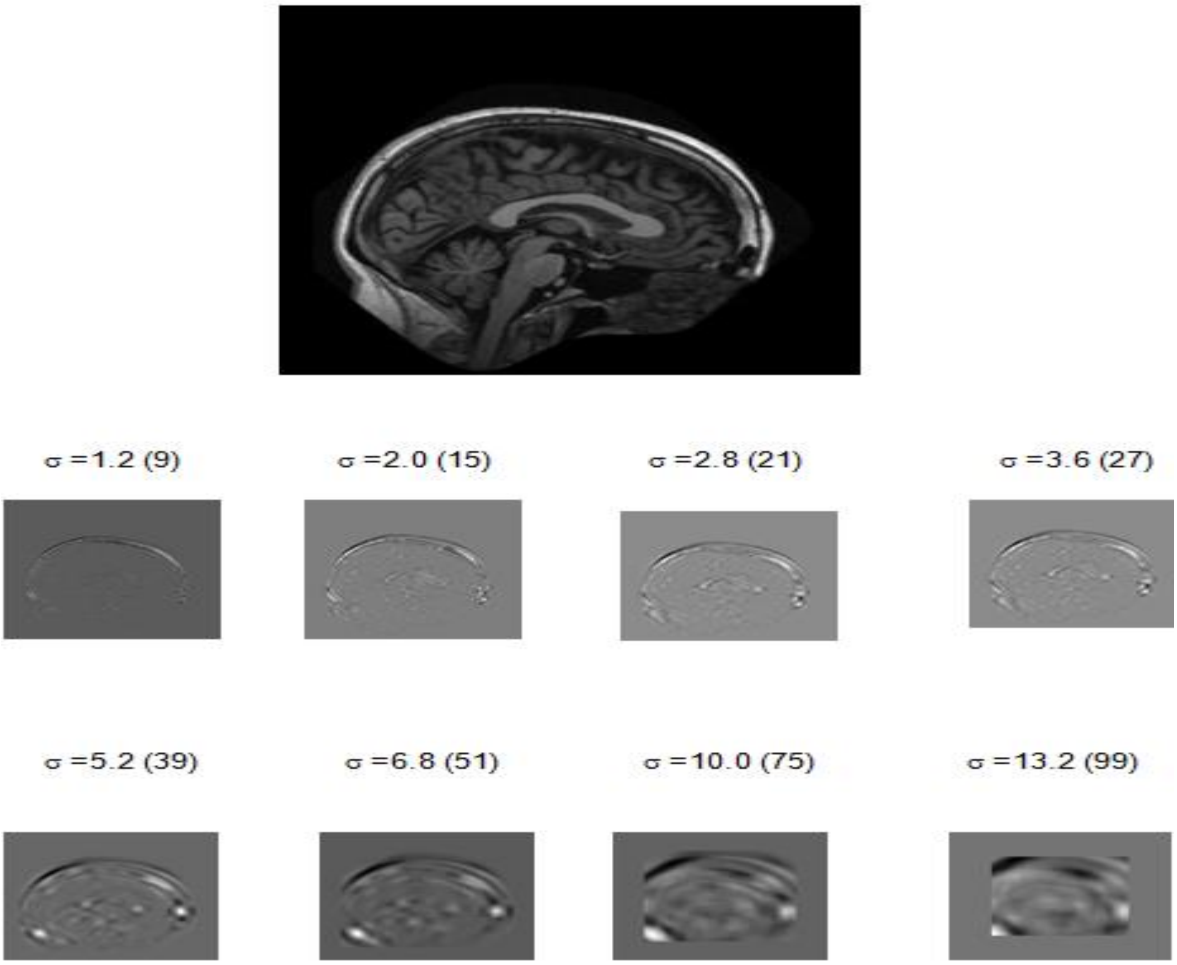


Figure 68 Example of Hessian matrix application

2- Find major interest points in scaled space, with non-maximal suppression on scaled interest point maps.

An octave is defined as a series of filters which have a range which approximates a doubling of scale. By computing 3 octaves with the option of going to 4 octaves, the octaves overlap to ensure full coverage of each scale as illustrated below figure:

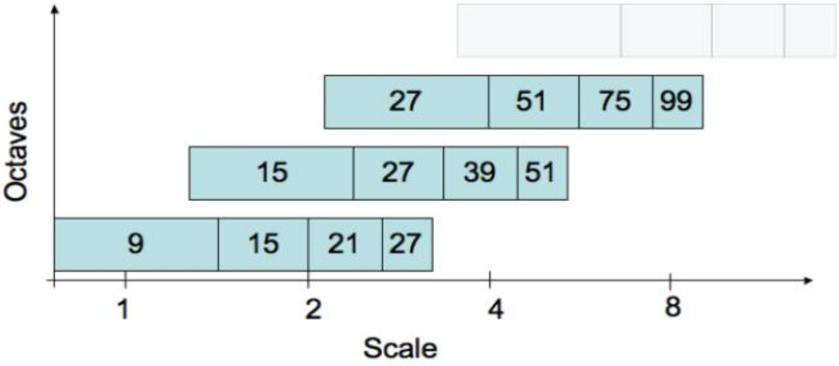


Figure 69 graphical representations of filter side lengths for three different octaves

To find the main image features we can apply non-maximal suppression, with normal 3*3 non-maximal suppression carried out within the same blob response map, and non-maximal suppression on the blob response maps above and below the image in scale space for each octave as illustrated below figure:

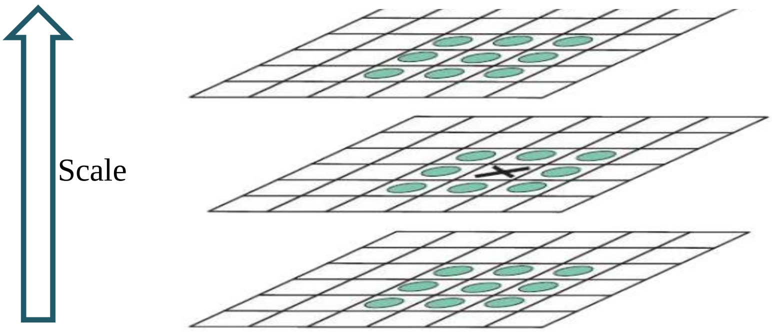


Figure 70 Non-maximal suppression

Because of the coarse scale of the scale space, we need to interpolate the interest point to arrive at the correct scale (σ), expressing the Hessian as a Taylor expansion as follows:

$$H(\mathbf{x}) = H + \frac{\partial H}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 H}{\partial \mathbf{x}^2}\mathbf{x} \tag{4.85}$$

Differentiating and setting to 0 gives:

$$\hat{x} = -\frac{\partial^2 H}{\partial \mathbf{x}^2}^{-1} \frac{\partial H}{\partial \mathbf{x}} \tag{4.86}$$

So the interpolated interest points are:

$$\frac{\partial^2 H}{\partial \mathbf{x}^2} = \begin{bmatrix} d_{xx} & d_{yx} & d_{sx} \\ d_{xy} & d_{yy} & d_{sy} \\ d_{xs} & d_{ys} & d_{ss} \end{bmatrix} \tag{4.87}$$

$$\frac{\partial H}{\partial \mathbf{x}} = \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix}. \tag{4.87}$$

3- Find features.

We can use Haar transforms (figure 71) to assess the primary direction of each feature, with the intuition being that they provide a sense of the direction of the change in intensity. They are also resistant to overall luminance changes. The figure below shows simple box filters or 'integral images'.

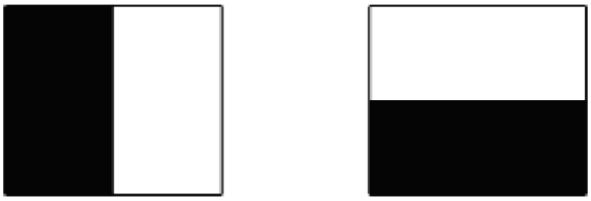


Figure 71 Haar wavelet filters employed to compute the responses in the x and y directions.

To compute the rotation we need to carry out the following steps for each feature:

- Look at pixels in a circle of 6*σ radius as illustrated figure 72.

- Compute the x and y Haar transform for each point.
- Use the resulting values as x and y coordinates in a Cartesian map.
- Rotate a wedge of π/3 radians about the circle.
- Choose the direction of maximum total weight.

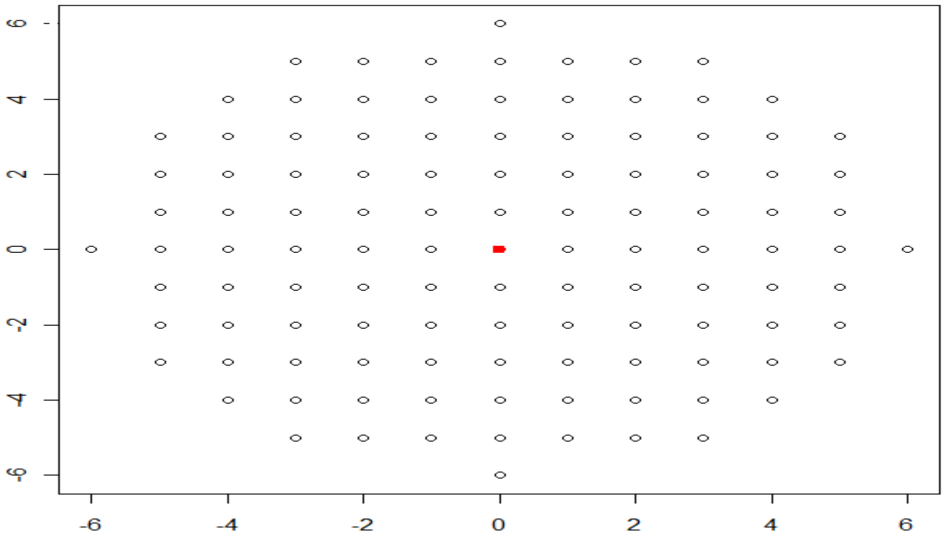


Figure 72 Neighborhood of radius 6s around the interest point

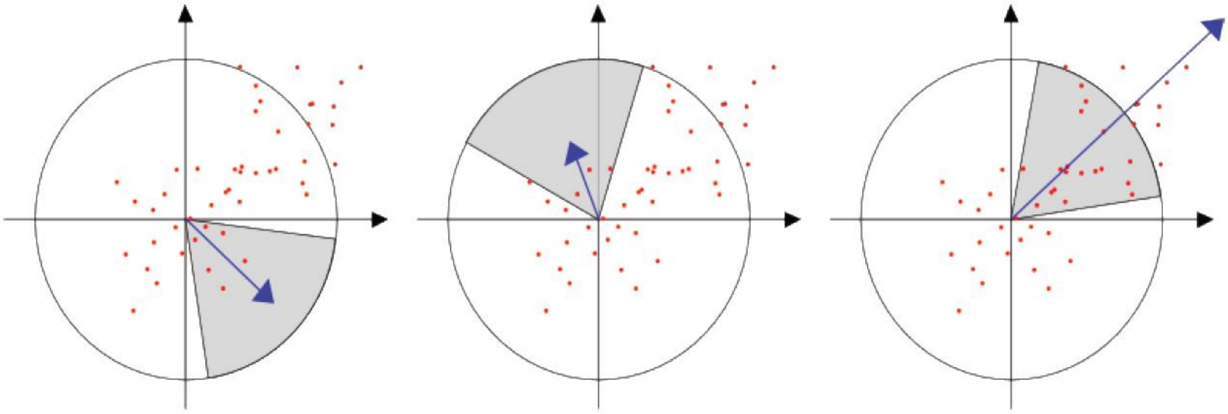


Figure 73 Orientation assignment

4- Generate feature vectors.

A square descriptor window is constructed with a size of 20* σ centered on each interest point and with an orientation based on the derived rotation figure 73. The descriptor window is then divided into 4*4 sub-regions for which:

- Each sub-region is a 5* σ square
- Haar wavelets of size 2* σ are computed for 25 regularly spaced points in each sub-region.
- For each of the 16 sub-regions we compute 4 values (sum dx, sum dy, sum abs(dx), and sum abc(dy)).
- The feature vector is a 64 dimensional vector consisting of the above 4 values for each of 16 sub-regions as the following figures:

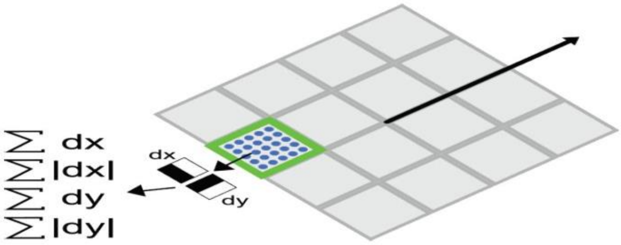


Figure 74 Feature descriptor in SURF with 64 dimensions

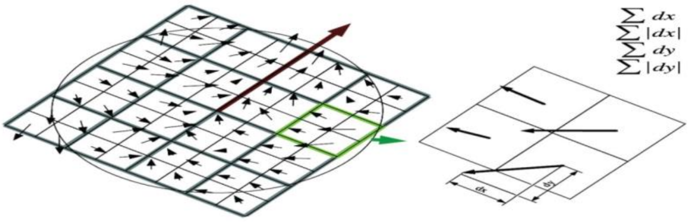


Figure 75 Building the descriptor

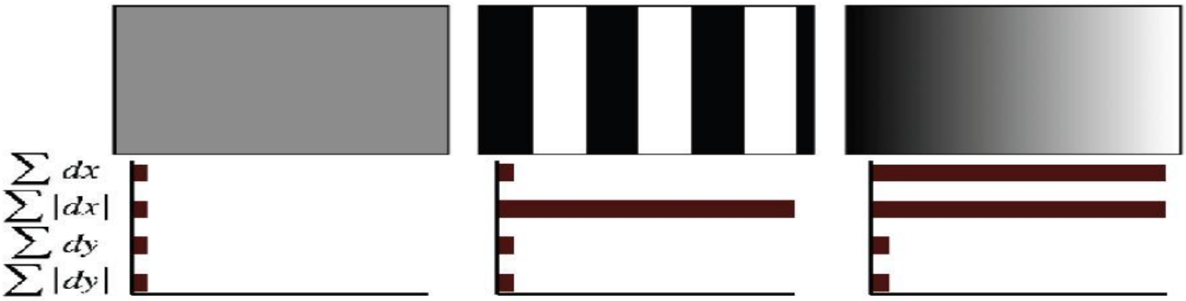


Figure 76 Descriptor entries of a sub-region representing the nature of the underlying intensity pattern.

Chapter 5

THE APPLICATION

Introduction

The present chapter describes the position estimation system which was designed for aircraft and UAV in particular figure 77. Originally, the system was developed under the assumption that the camera would be attached to the aircraft, relying solely on binocular vision to apply simple obstacle avoidance. However, to simplify system integration, the system is now implemented in a payload module consisting of two cameras and a general purpose CPU. The following section contains a detailed discussion regarding the development of an algorithm for obstacle avoidance. The system consists of two parts: hardware and software. The hardware includes two cameras and a computer, while the software covers the following seven functions (figure 78):

1- Vision system calibration: to produce an estimate of both extrinsic and intrinsic camera parameters for stereo camera calibration.
2- Image pre-processing: image enhancement, color segmentation, and region segmentation.
3- Stereo vision system: extracting the depth and size of objects in the real world.
4- Real-time information: the result of the image pre-processing and stereo vision system steps.
5- Database information: all information about objects located under the flight path .
6- Comparison: critical for making decisions with which to recognize objects in the real word by comparing system-derived information with database information.
7- Position estimation: to determine the distance of the UAV from the object, and the former’s orientation.

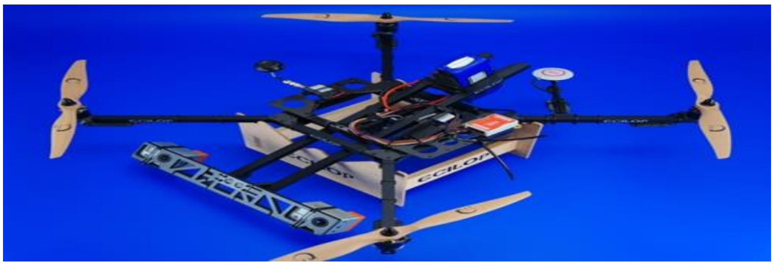


Figure 77 Project hardware (quadcopter with two cameras)

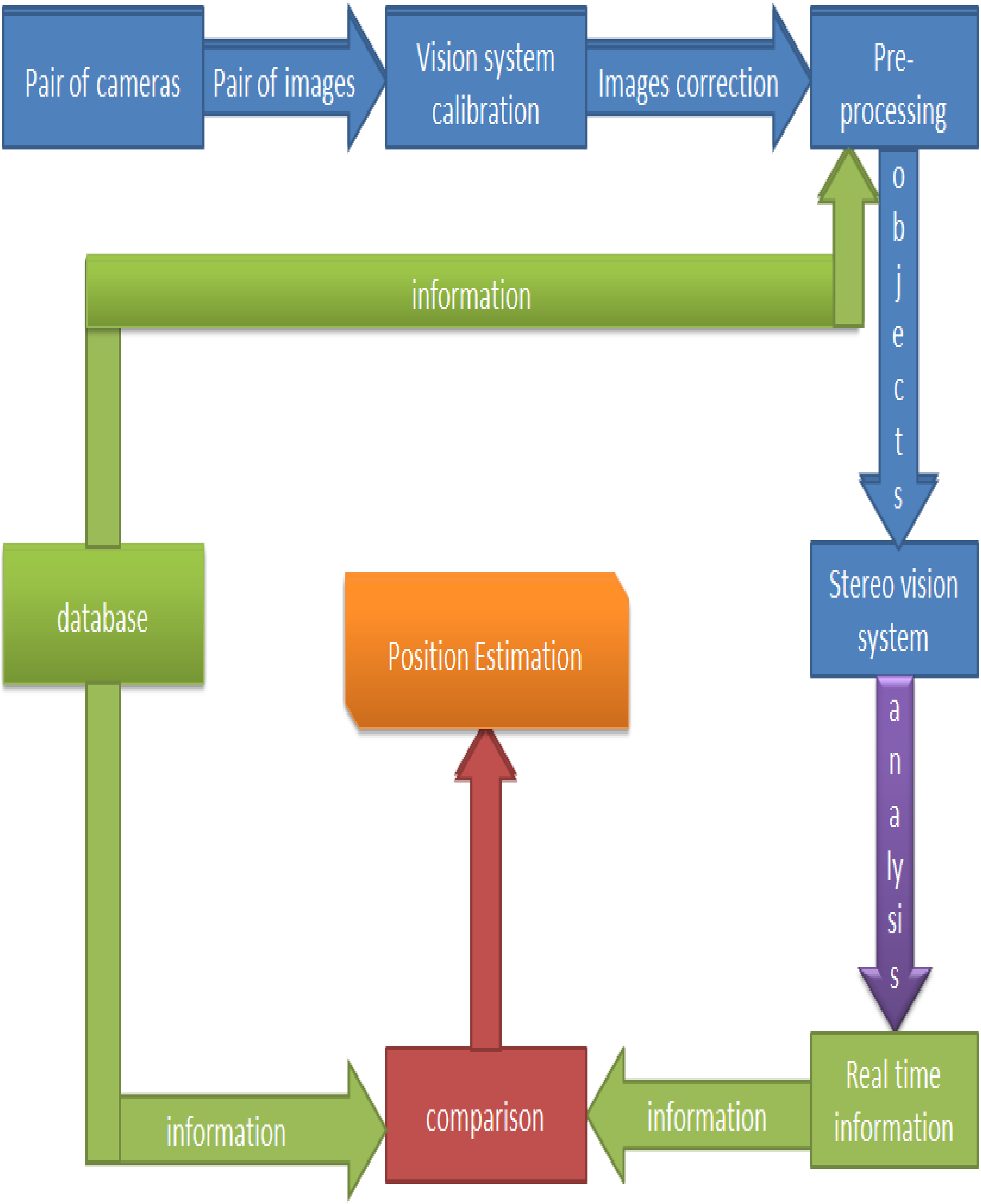


Figure 78 Block diagram of project software

5.1 Vision System Calibration

The vision system calibration process consists of the following four steps:

5.1.1 Single Camera Calibration

This step is necessary in computer vision in order to extract metric information from 2D images. The proposed application uses ‘Plane-based internal parameter estimation’ for image correction and the extraction of camera parameters such as focal length, size of pixel, and point of origin. This information then assists in determining the properties of objects in the real world. Single camera calibration can be described by the following block diagram:

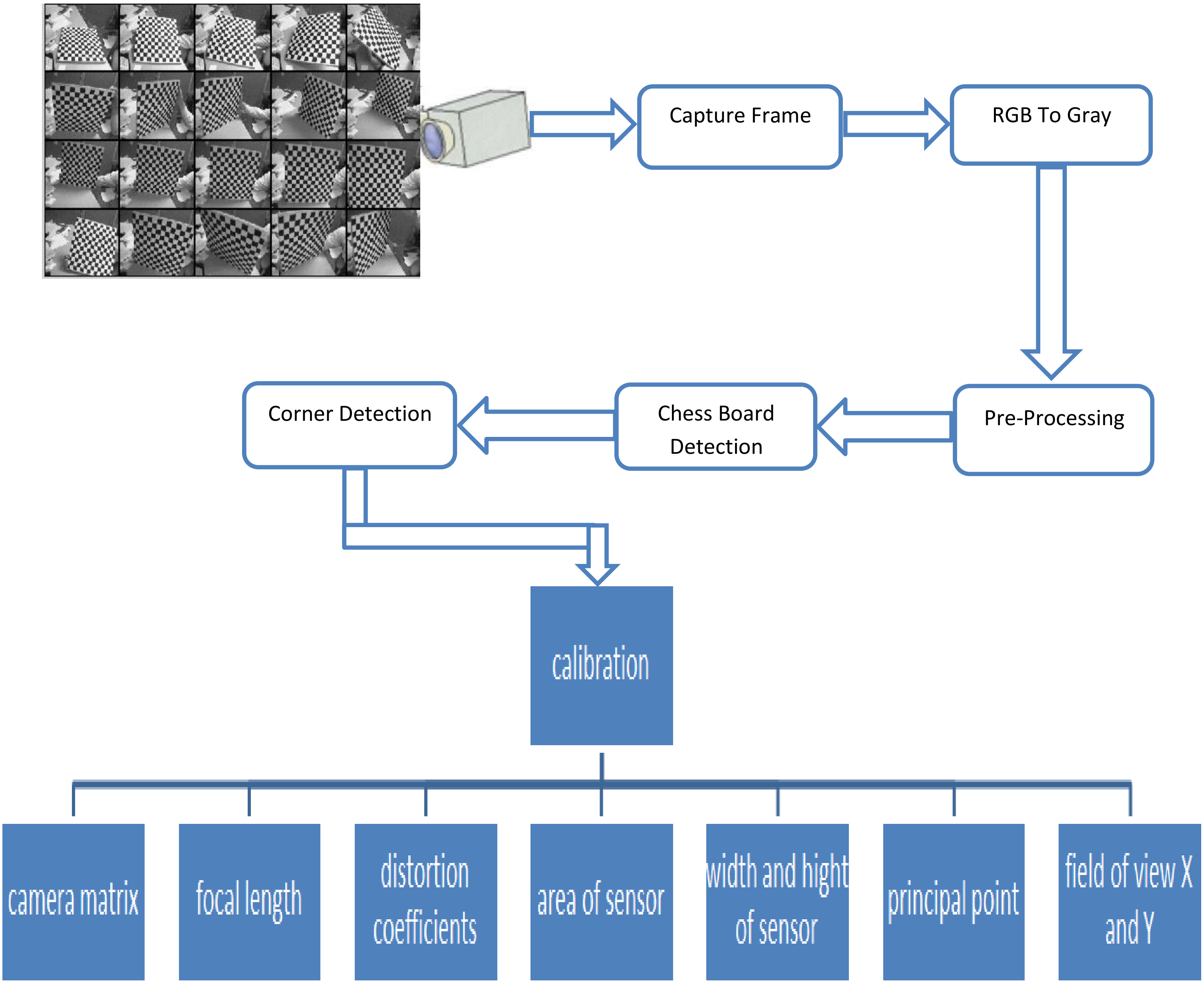


Figure 79 Block diagram of single camera calibration

To calibrate the camera based on this algorithm we here use the chess board pattern (figure 80) presented below during the following steps:

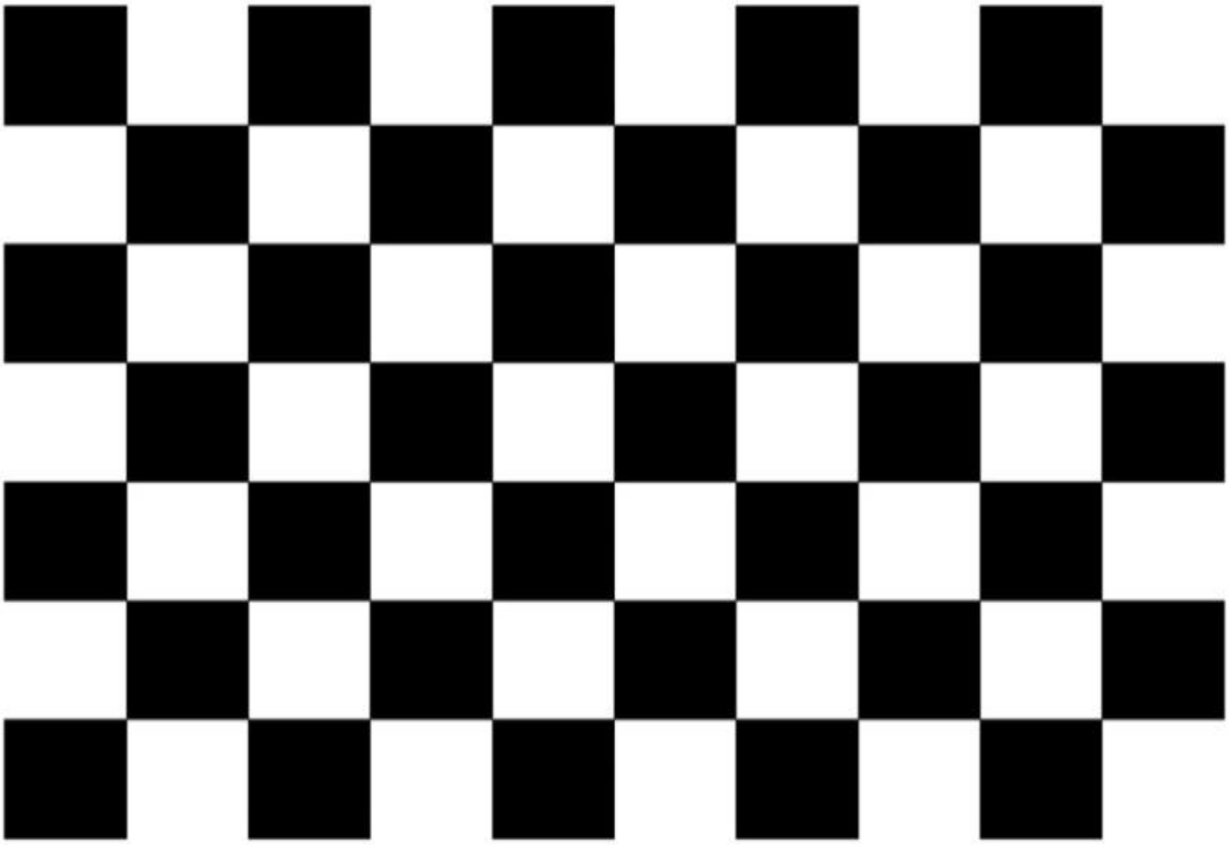


Figure 80 Calibration pattern (9*6 chess board)

1- Capture an image ‘frame’ from a live video feed in the RGB color space system.
2- Convert the image from the RGB color space system to the Gray color space system. The latter is considered more effective for image processing due to increased speed and accuracy.
3- Attempt to reduce noise by using a median filter with window size ‘5’.
4- Improve the picture and make full utilization of its conceivable qualities by using a contrast-stretching algorithm.
5- Detect the chessboard via the pattern localization algorithm (this algorithm consists of two algorithms: 1-corner detection and line detection using the Hough line transform).
6- Identify and label the corners inside the chessboard pattern.
7- Repeat the above process (from step 1 to step 6) 20 times.
8- Apply the camera calibration algorithm as outlined in Chapter 3.
9- Generate XML file content for all camera information.

5.1.2 Image Rectification and Mapping

As explained in Chapter 3, lens distortion is a significant problem for real cameras. The developed system acts to rectify these distortions as follows figure:

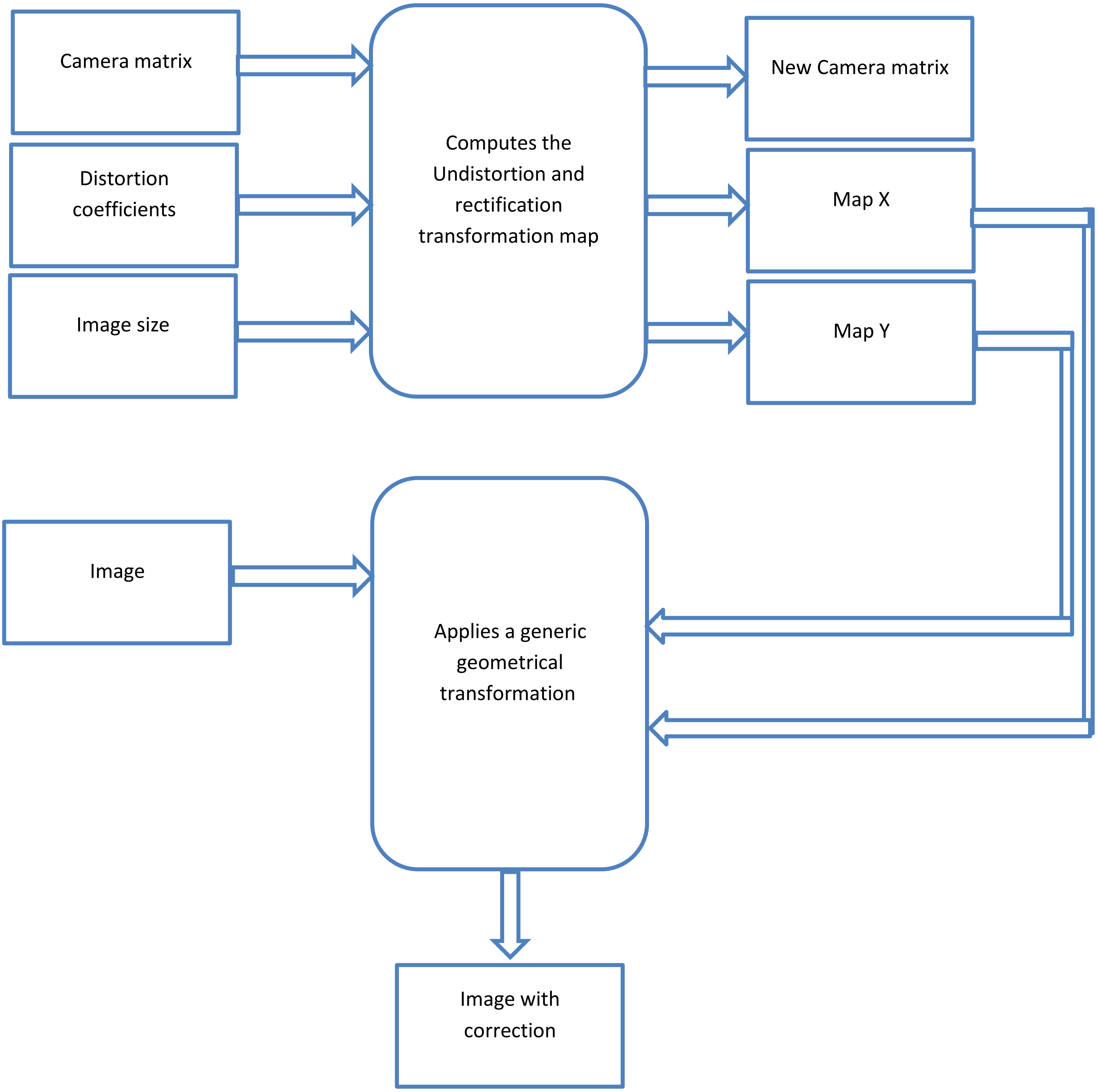

Camera matrix
Distortion coefficients
Image size
Computes the Undistortion and rectification transformation map
New Camera matrix
Map X
Map Y
Image
Applies a generic geometrical transformation
Image with correction

Figure 81 Block diagram of image rectification and mapping

5.1.3 Stereo Camera Calibration

As discussed in Chapter 3, the aim of stereo calibration is to correct the position of the two cameras, and extract the essential matrix and fundamental matrix as illustrated below figure:

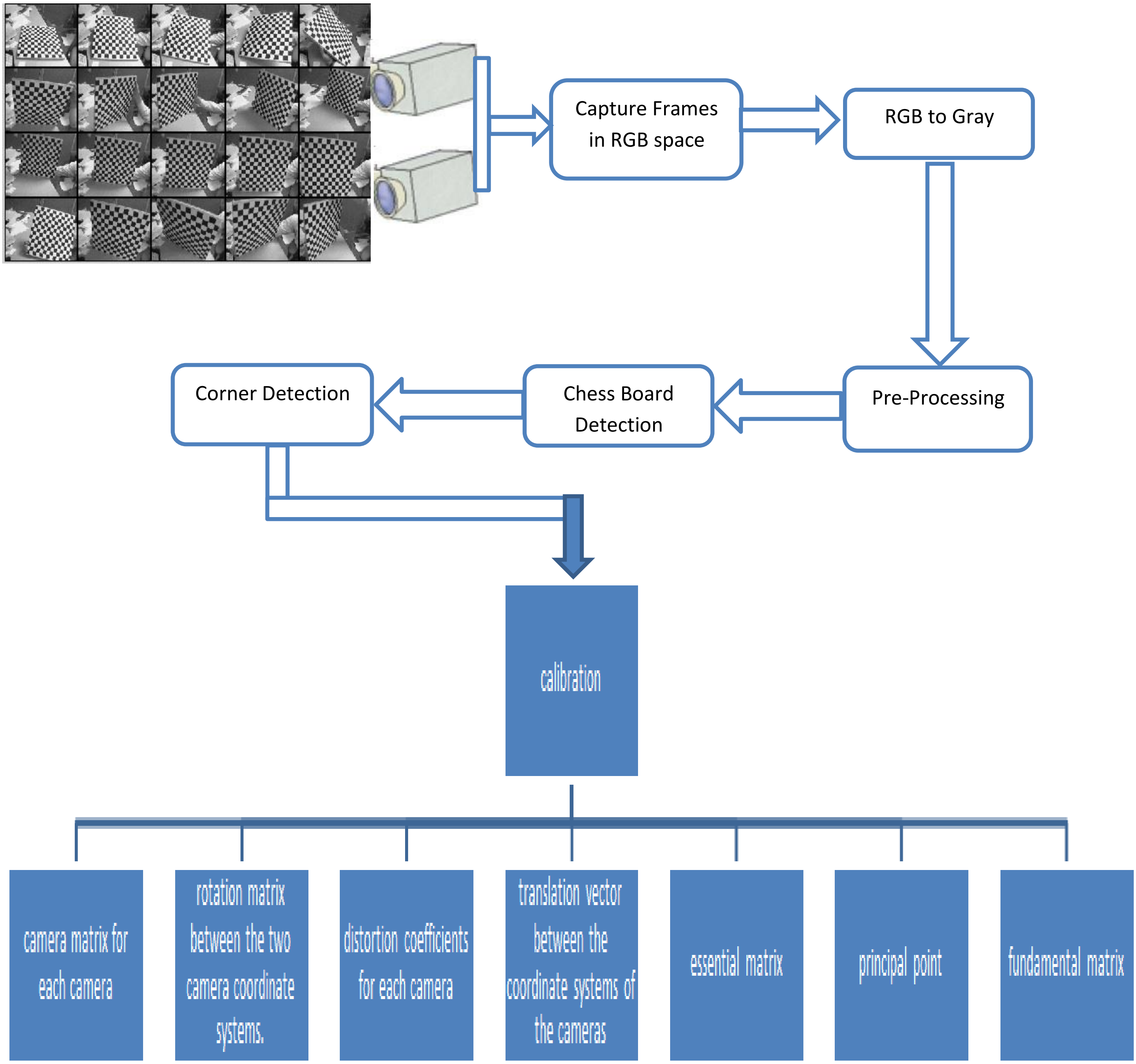

Capture Frames in RGB space
RGB to Gray
Pre-Processing
Chess Board Detection
Corner Detection
calibration
camera matrix for each camera
rotation matrix between the two camera coordinate systems.
distortion coefficients for each camera
translation vector between the coordinate systems of the cameras
essential matrix
principal point
fundamental matrix

Figure 82 Block diagram of stereo camera calibration

1- Capture image ‘frames’ from a live video feed in the RGB color space system using both cameras.
2- Convert the images from the RGB color space system to the Gray color space system. The latter is considered more effective for image processing due to increased speed and accuracy.
3- Reduce image noise by using a median filter with window size ‘5’ in both images.
4- Improve the picture and make full utilization of its conceivable qualities by using the contrast-stretching algorithm.
5- Detect the chessboard via the pattern localization algorithm as a figure 83 (this algorithm consists of two algorithms: 1-corner detection and line detection using the Hough line transform) in both images.
6- Identify and label the corners inside the chessboard pattern in both images.
7- Repeat the above process (from step 1 to step 6) 20 times.
8- Apply the stereo camera calibration algorithm, as outlined in Chapter 3.
9- Generate XML file content for all camera information.

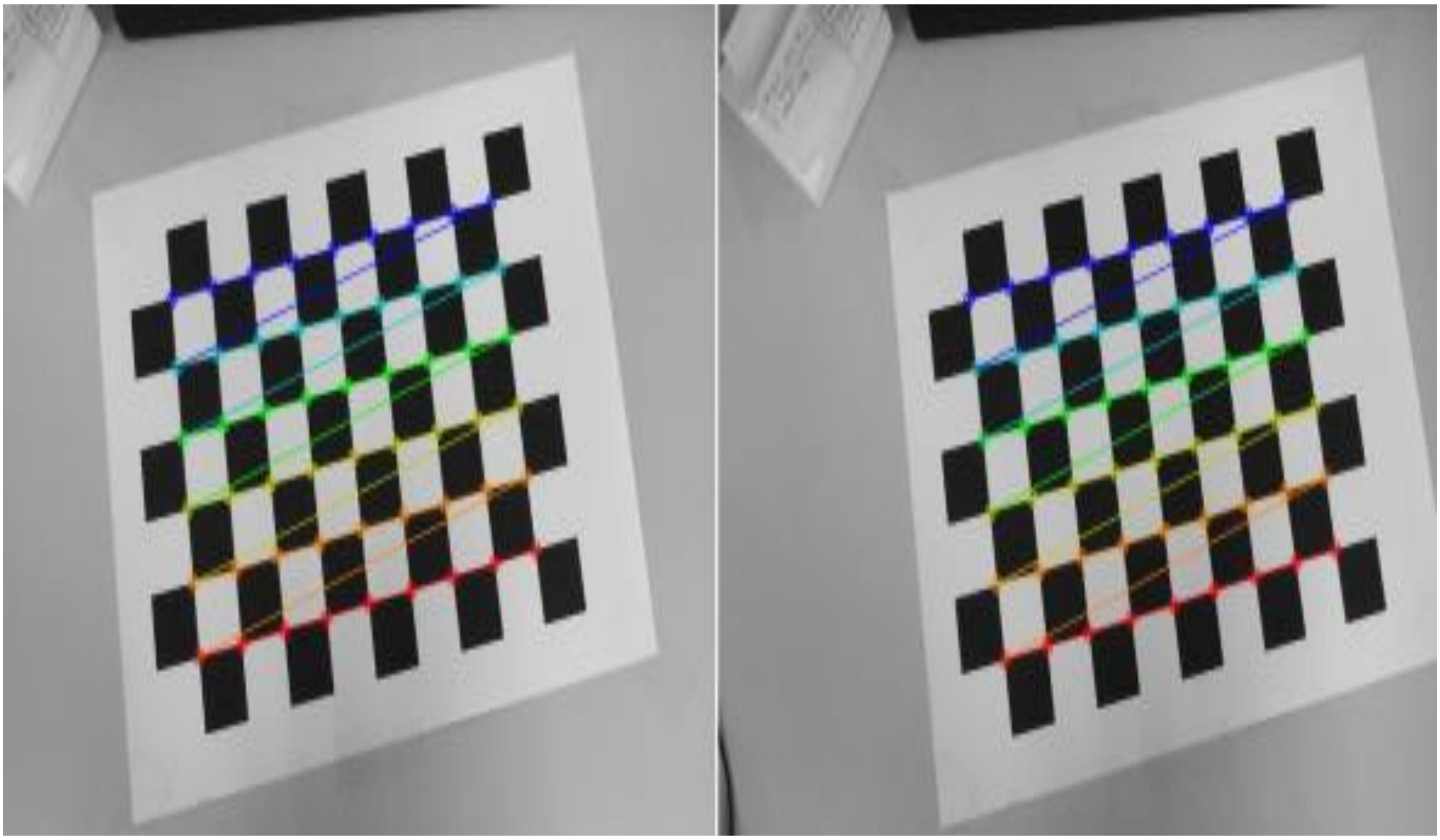

a
b

Figure 83 Stereo camera calibration: a) left-hand camera, b) right-hand camera

5.1.1.4 Stereo Image Rectification and Mapping

As discussed in Chapter 3, to obtain corresponding lines in the images produced by two cameras, we built the following model:

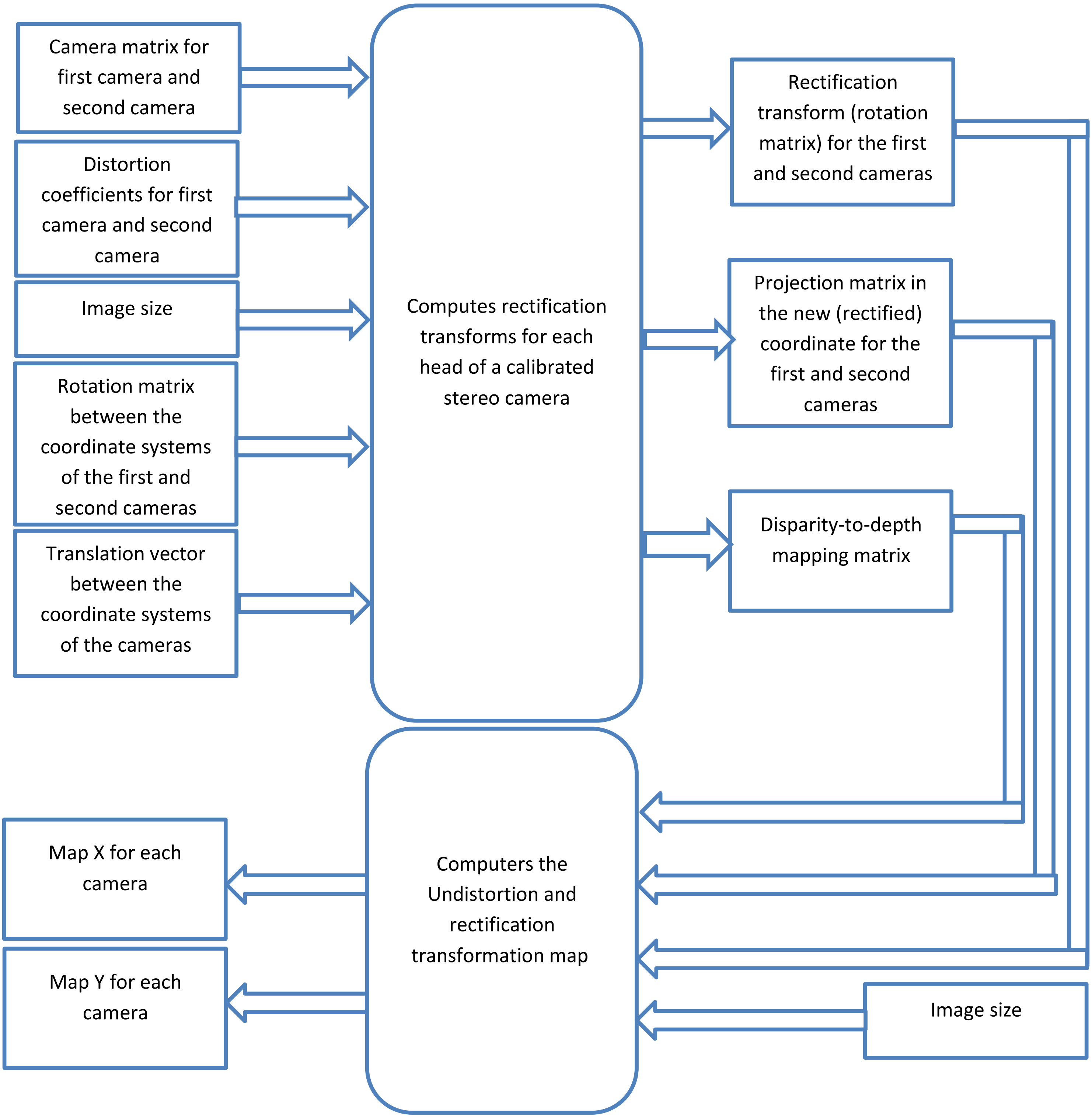

Camera matrix for first camera and second camera
Distortion coefficients for first camera and second camera
Image size
Rotation matrix between the coordinate systems of the first and second cameras
Translation vector between the coordinate systems of the cameras
Computes rectification transforms for each head of a calibrated stereo camera
Rectification transform (rotation matrix) for the first and second cameras
Projection matrix in the new (rectified) coordinate for the first and second cameras
Disparity-to-depth mapping matrix
Map X for each camera
Map Y for each camera
Computers the Undistortion and rectification transformation map
Image size

Figure 84 Block diagram of stereo image rectification and mapping

5.2 Image Pre-Processing

Image pre-processing consists of the following three steps:

5.2.1 Image Enhancement

Median filtering is employed to reduce noise in image pairs, thereby increasing the accuracy of the results as illustrated below figures:

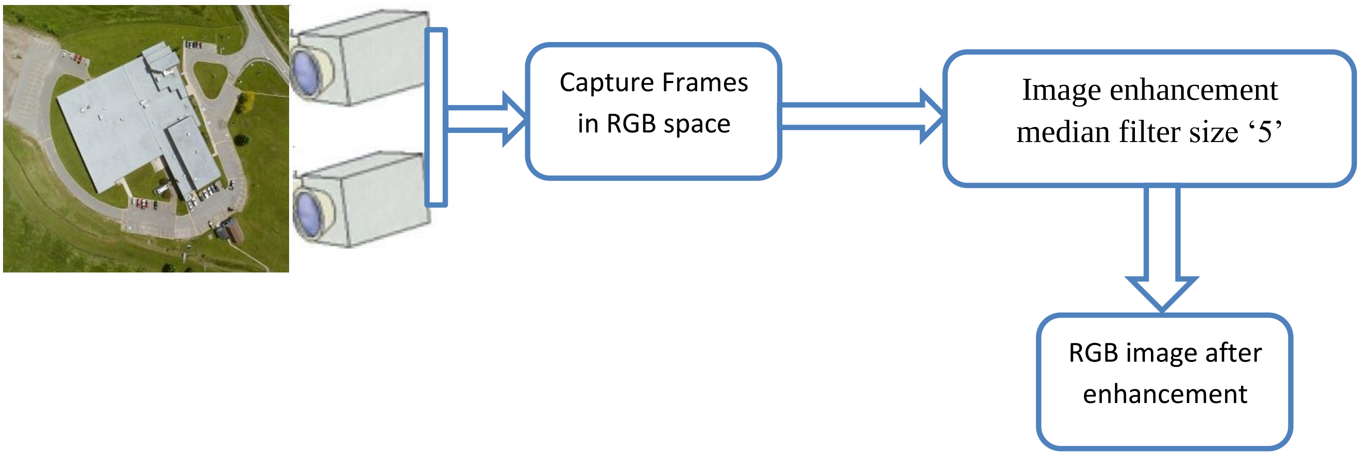
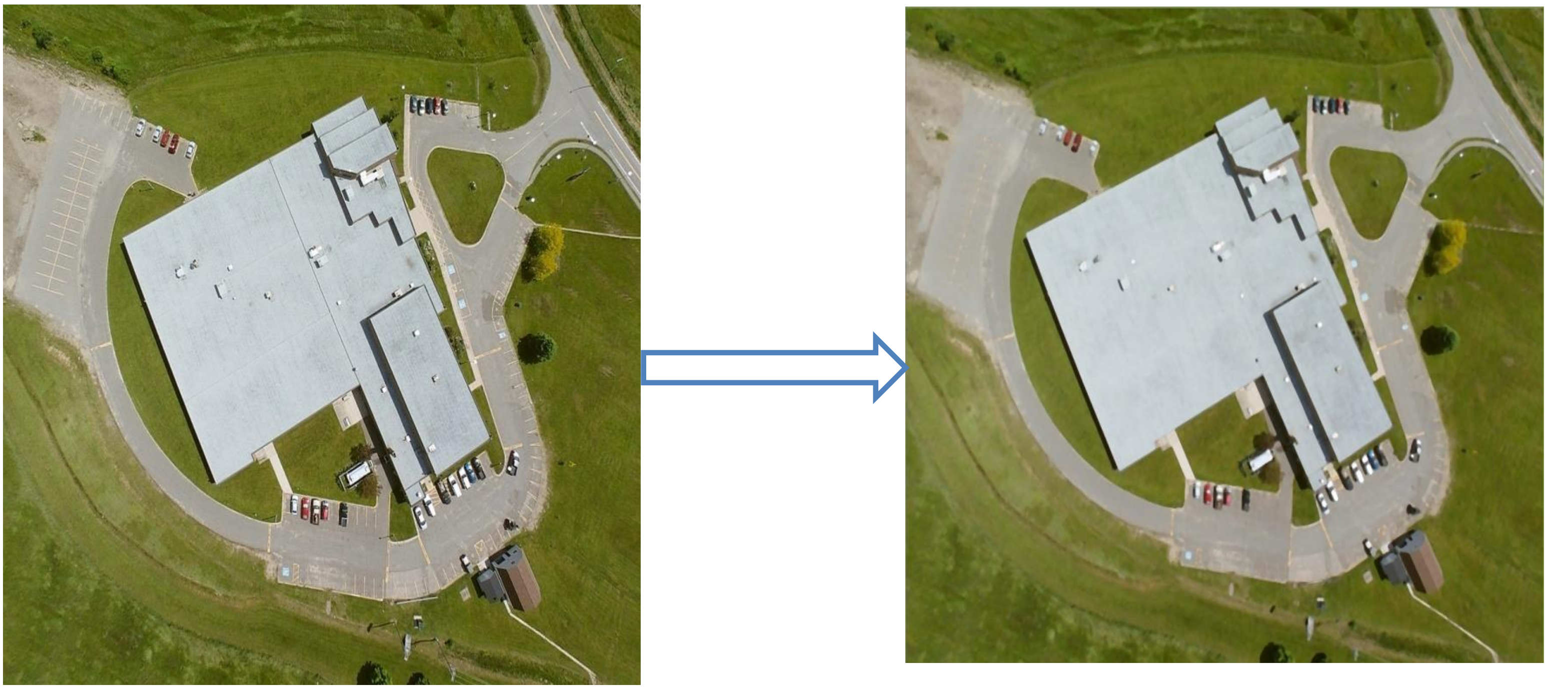

Capture Frames in RGB space
Image enhancement median filter size ‘5’
RGB image after enhancement

Figure 85 Block diagram of image enhancement



a

b

Figure 86 Example of image enhancement: a) image before enhancement, b) image after enhancement

5.2.2 Color Segmentation

Images are converted from the RGB system space to the HSV system space, with thresholding techniques then used to segment the candidate regions in order to reduce the number of potential areas and thus increase processing speed and accuracy as illustrated below figures:

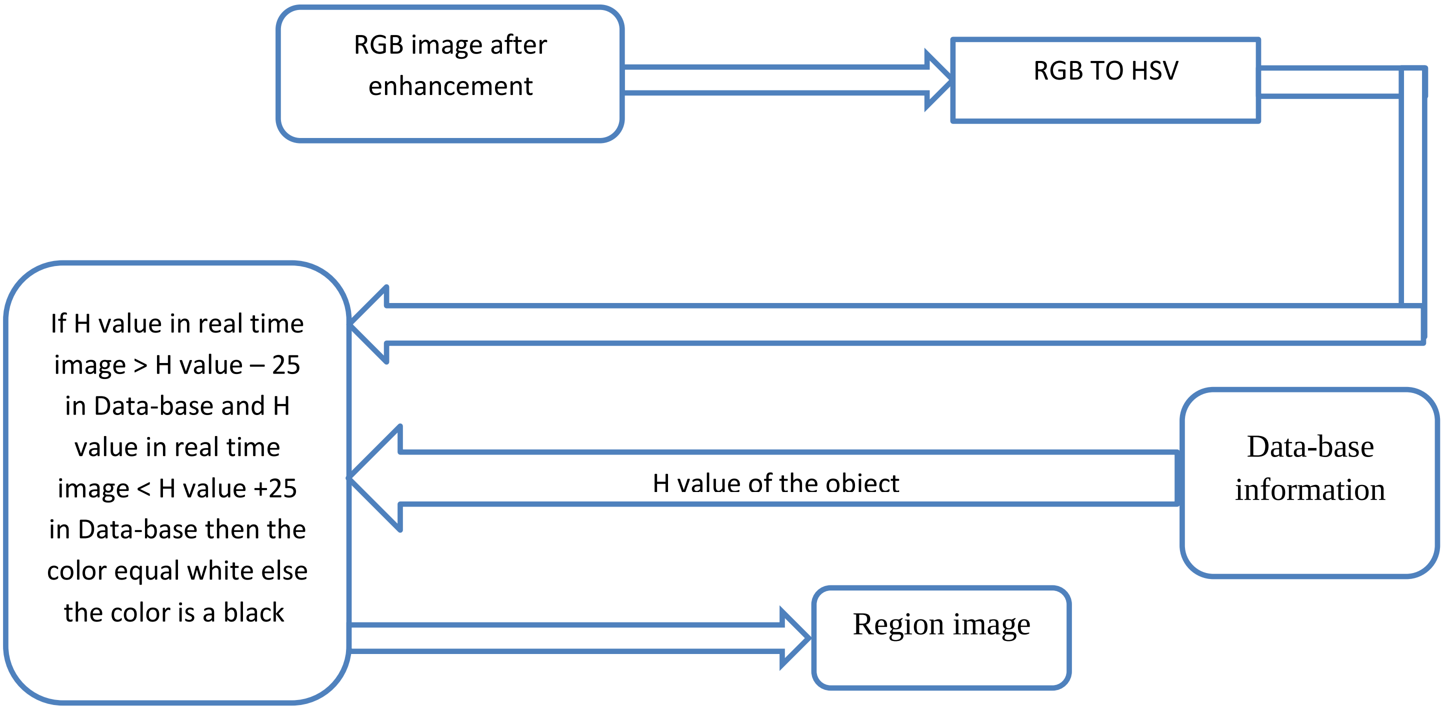
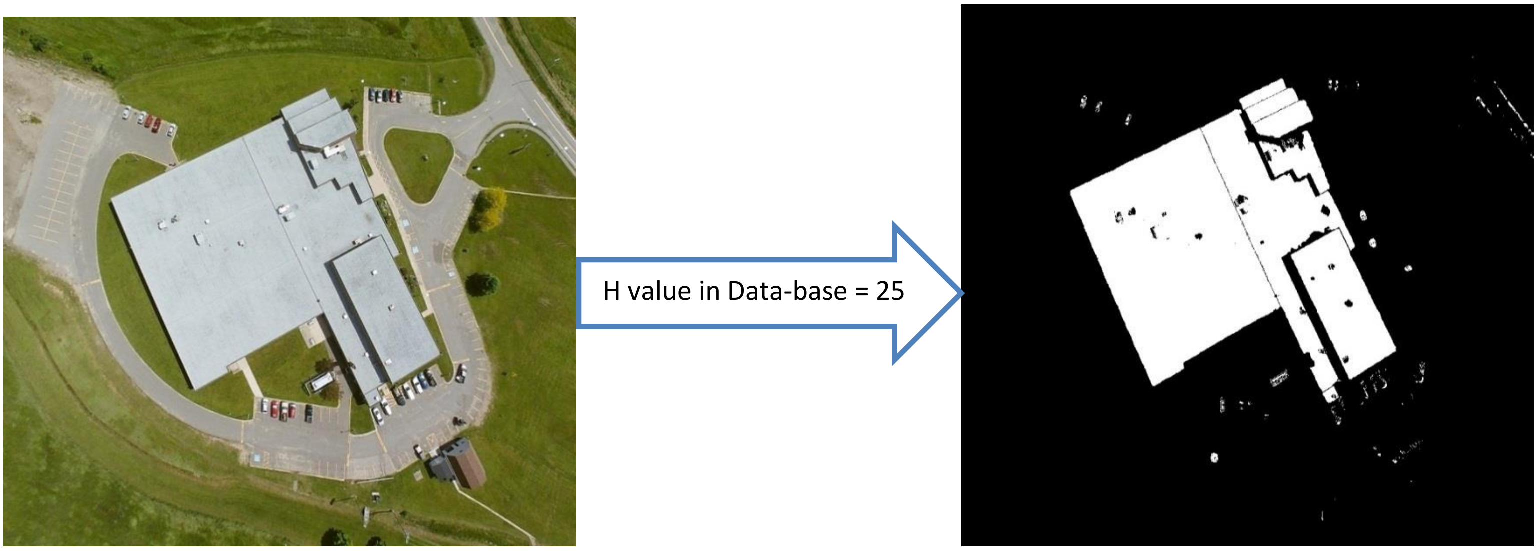

RGB image after enhancement
RGB TO HSV
If H value in real time image > H value – 25 in Data-base and H value in real time image < H value +25 in Data-base then the color equal white else the color is a black
H value of the obiect
Data-base information
Region image

Figure 87 Block diagram of color segmentation



a

b

Figure 88 Color segmentation: a) original image, b) image after segmentation

5.3 Region Segmentation

A median filter with window size ‘5’ is used to remove small regions and thus narrow down the search field. Labeling of each region is then carried out only in the left-hand image of the image pair as illustrated below figures:

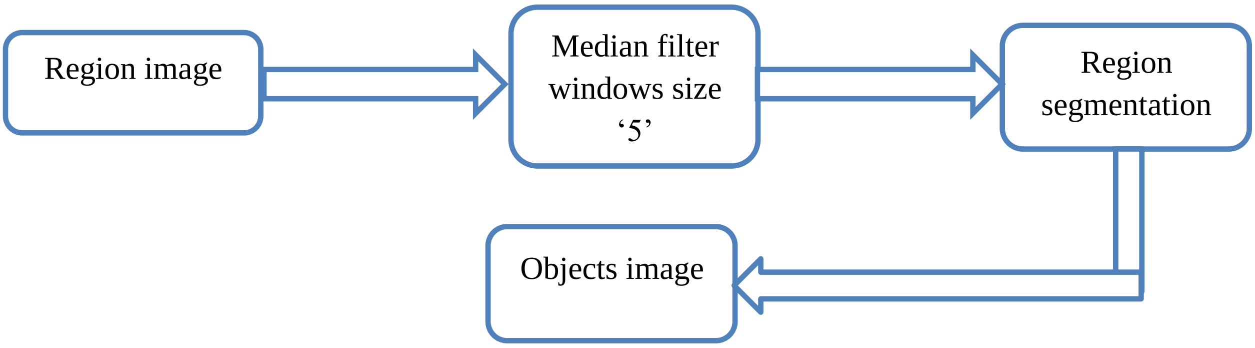


Figure 89 Block diagram of region segmentation

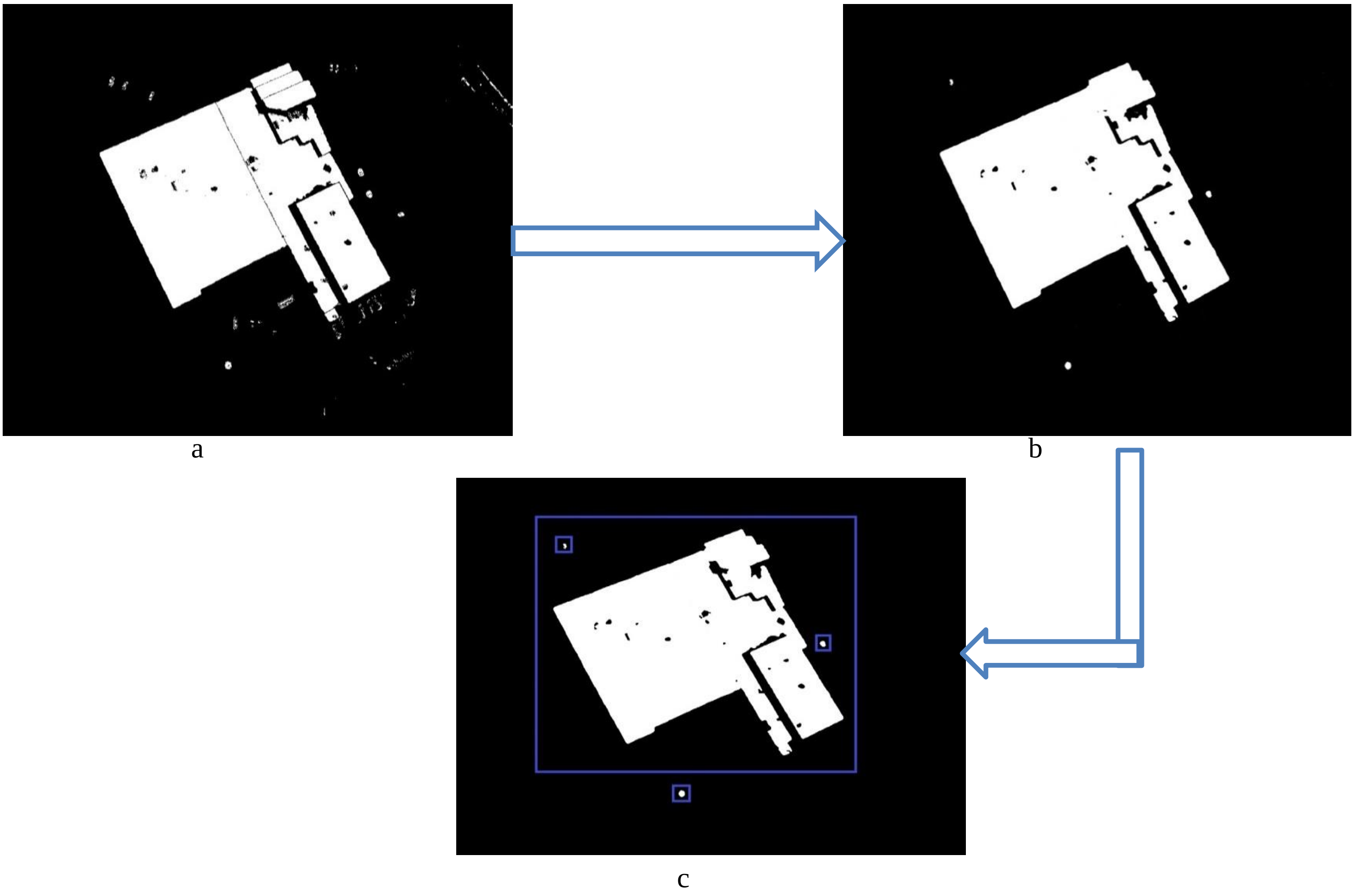


Figure 90 Region labeling: a) region image, b) enhancement image, c) region labeling

5.4 Stereo Vision

The aim of this step is to convert objects into information. Object edges are first identified using Sobel edge detection, with template matching then carried out via the normalized correlation coefficient method, followed by disparity calculations. Objects are finally converted into information via stereo vision using the techniques discussed in Chapter 3, together with the “fill ratio” method. As the ratio of the number of pixels in the object to the size of the object, the fill ratio can be used to recognize object shape, as illustrated below figures:

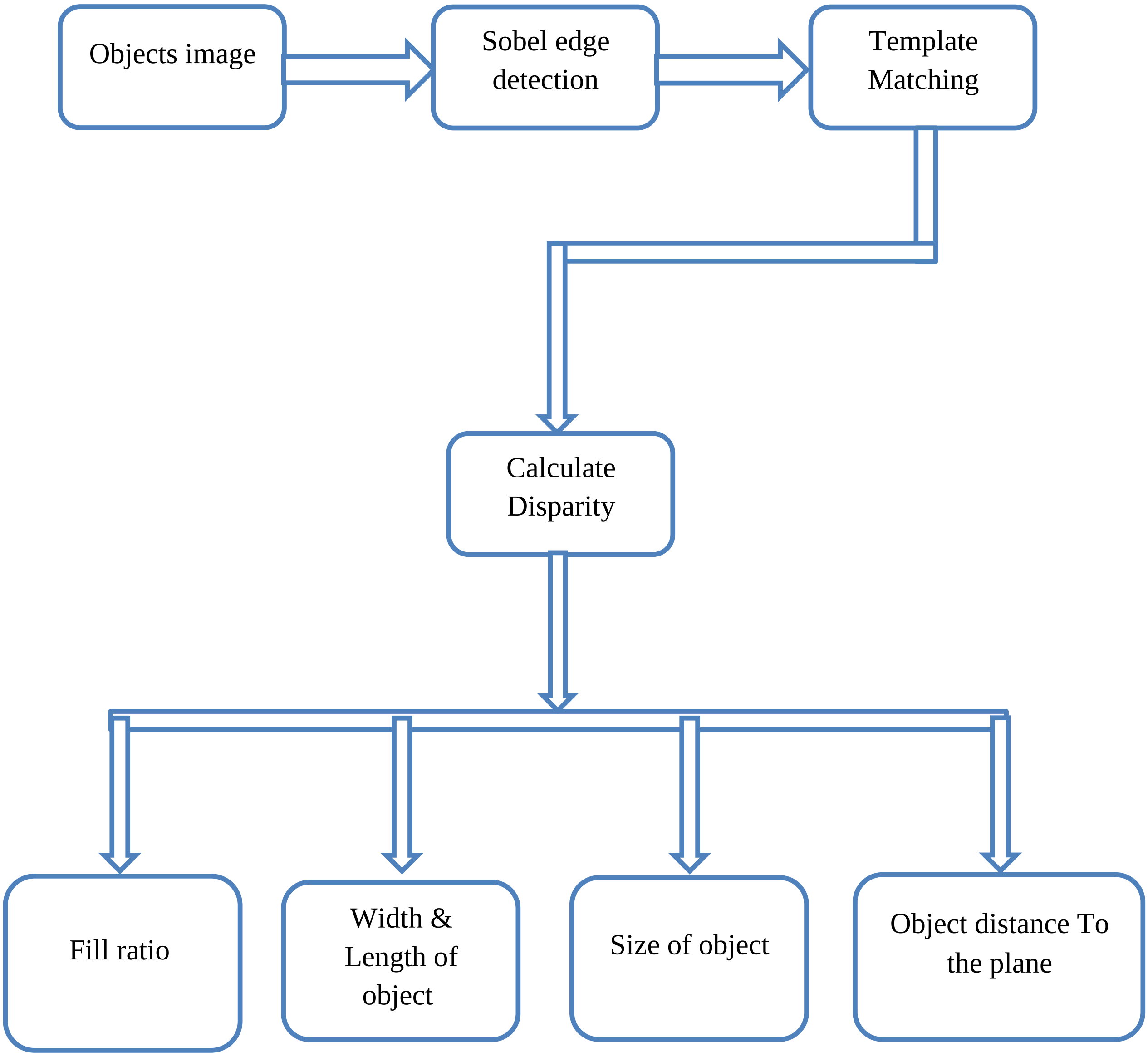


Figure 91 Block diagram of stereo vision information extraction

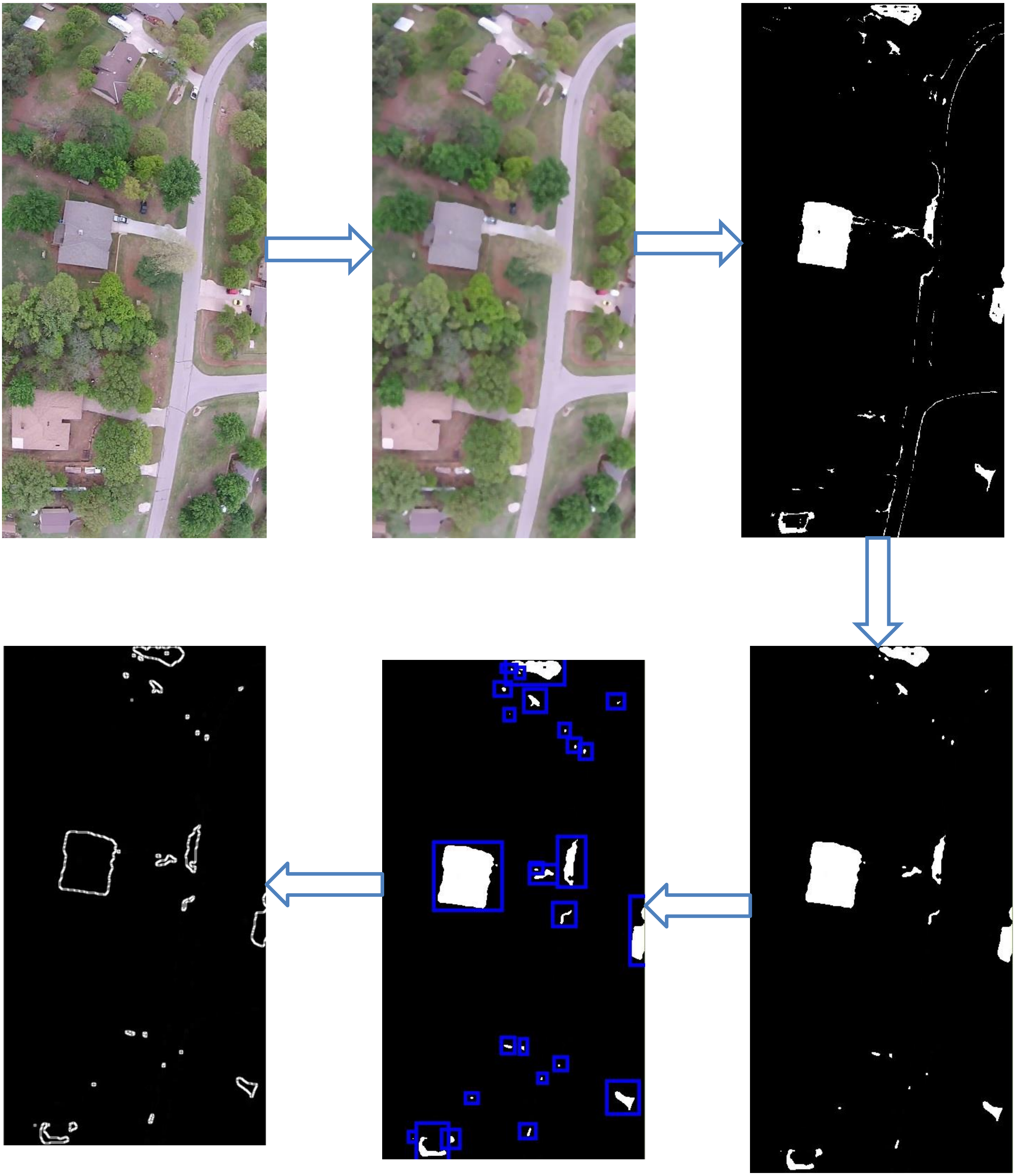


Figure 92 Extraction and labeling of objects in the left-hand camera image

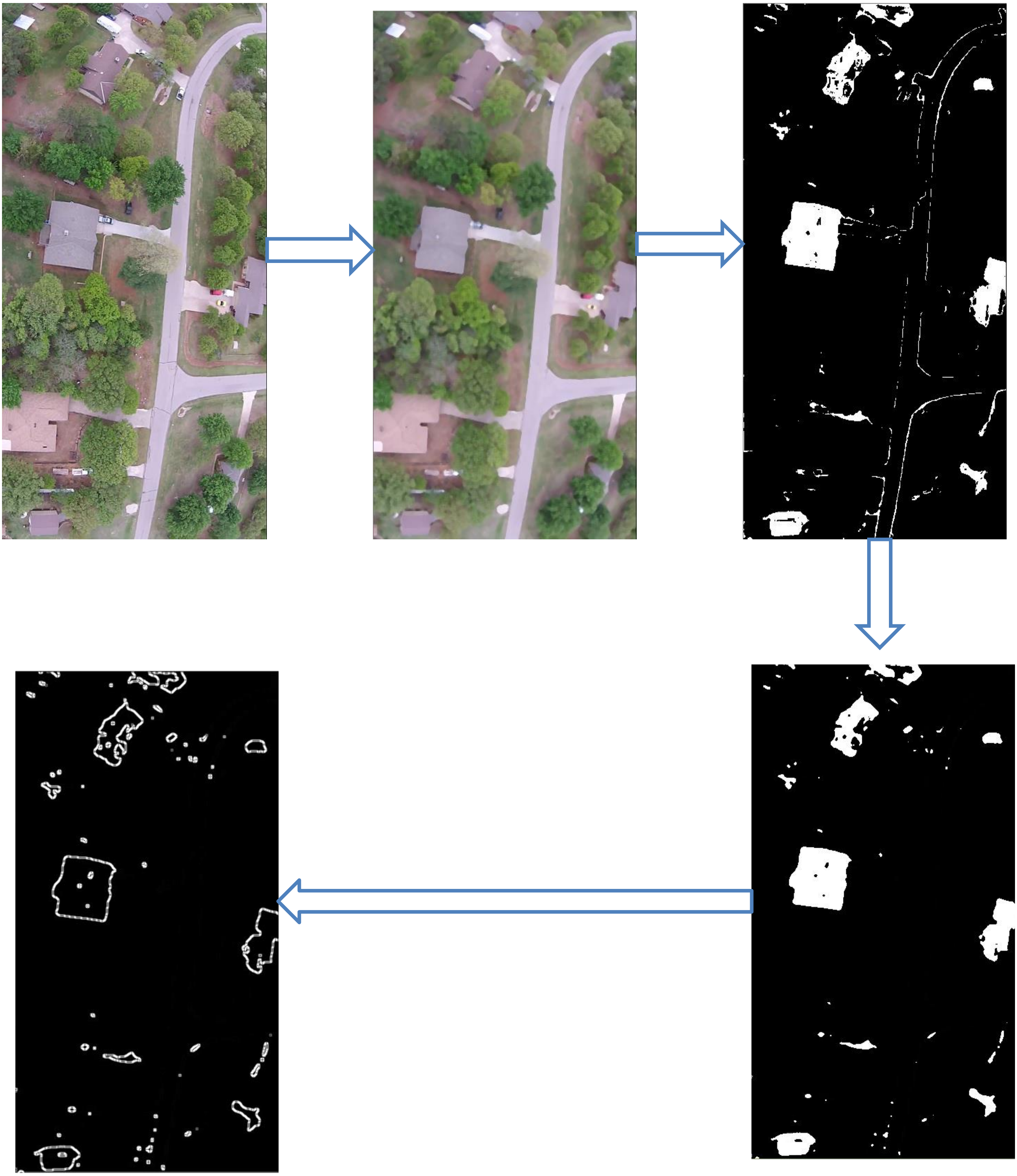


Figure 93 Extraction of objects from the right-hand camera image

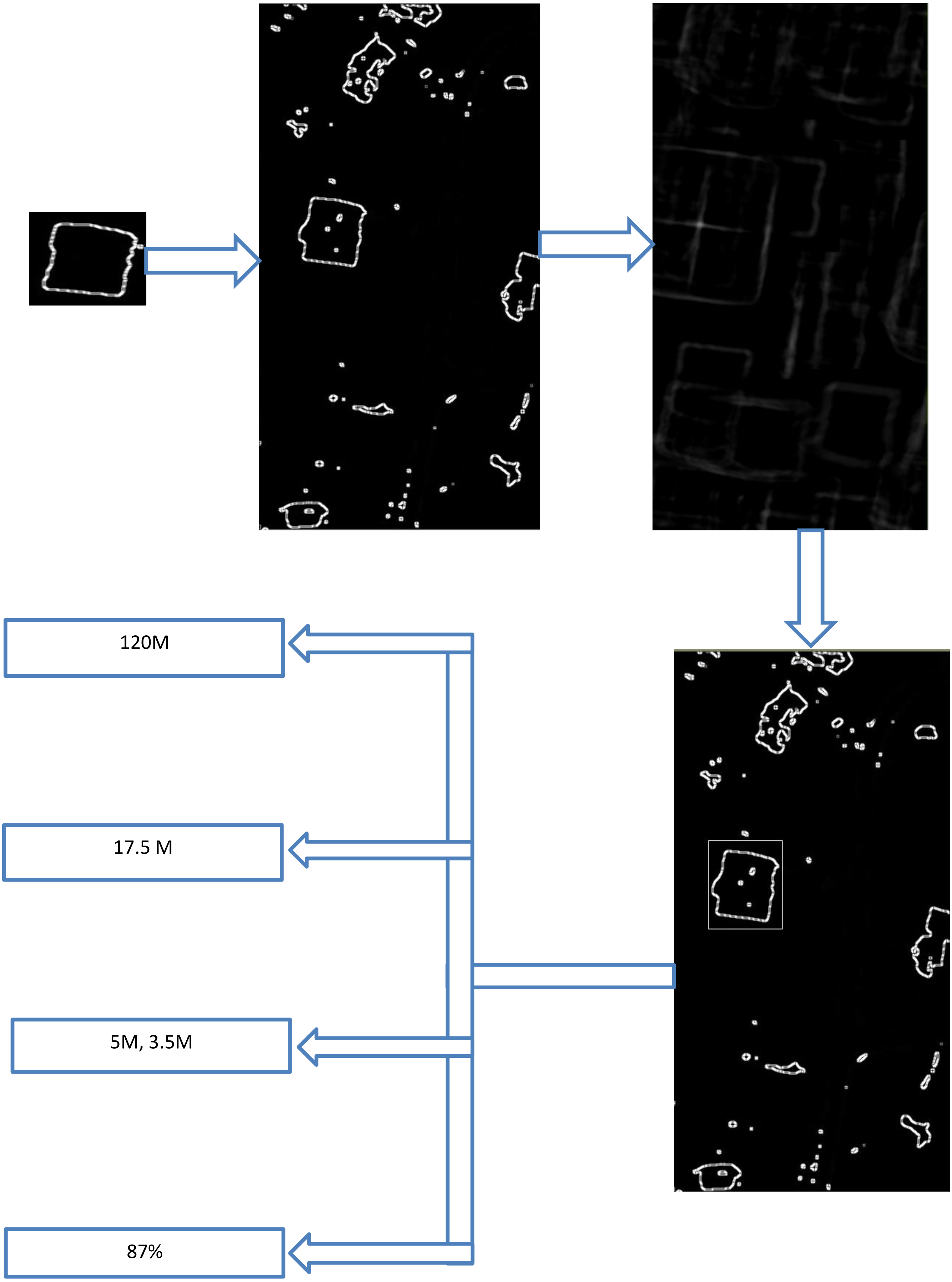


Figure 94 Stereo vision and information generation

5.5 Comparison

In this step, the real-time information generated in the previous step (size, width & length of object, and fill ratio) is compared with the corresponding object information contained in the database (figure 95). If the similarity ratio is greater than 75%, the system can be considered to have achieved its aim. The similarity ratio is calculated via the following equation:

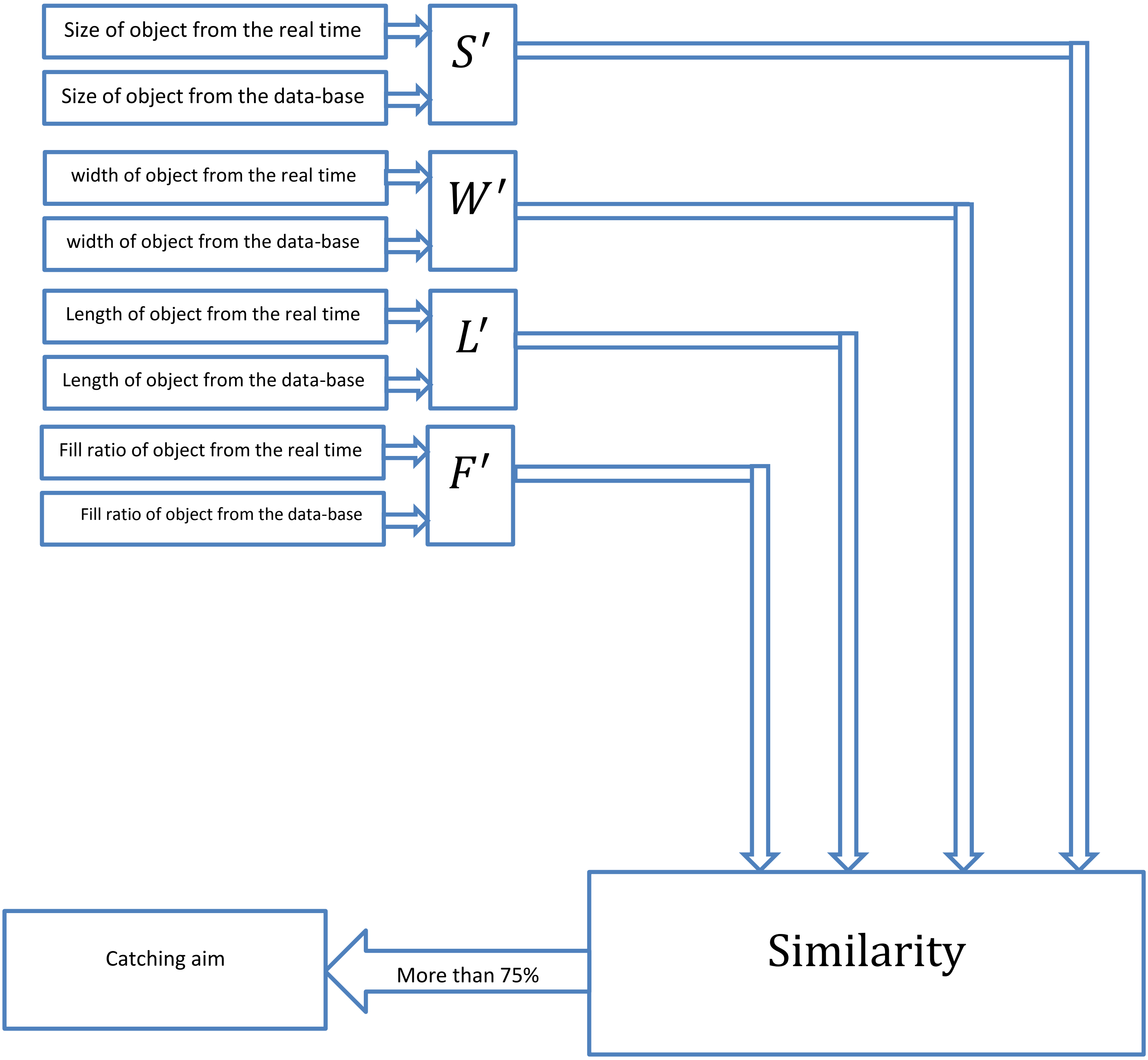

Size of object from the real time
Size of object from the data-base
S′
width of object from the real time
width of object from the data-base
W′
Length of object from the real time
Length of object from the data-base
L′
Fill ratio of object from the real time
Fill ratio of object from the data-base
F′
Similarity
More than 75%
Catching aim

Figure 95 Block diagram of real-time and database information comparison

$$S' = \frac{\text{size\_from\_real\_time}}{size\_from\_data\_base} * 100 \tag{5.1}$$

$$W' = \frac{\text{width\_of\_object\_from\_real\_time}}{width\_of\_object\_from\_data\_base} * 100 \tag{5.2}$$

$$L' = \frac{\text{Length\_of\_object\_from\_real\_time}}{Length\_of\_object\_from\_data\_base} * 100 \tag{5.3}$$

$$F' = \frac{\text{Fill\_ratio\_of\_object\_from\_real\_time}}{Fill\_ratio\_of\_object\_from\_data\_base} * 100 \tag{5.4}$$

$$\text{Similarity} = S' * w1 + W' * w2 + L' * w3 + F' * w4 \tag{5.5}$$

Where the weights are:

W1=0.20966

W2=0.304130

W3=0.302214

W4=0.209658

The weights given above are general, and can be obtained from the database for each aim. If the similarity is greater than 75%, the result can be accepted, otherwise the information must be rejected.

5.6 Position Estimation

In this step we will determine the position of the plane on the map by calculating the distance between the plane and the examined object (whose location on the map is known), and the pose of the plane to the object, as illustrated below figures:

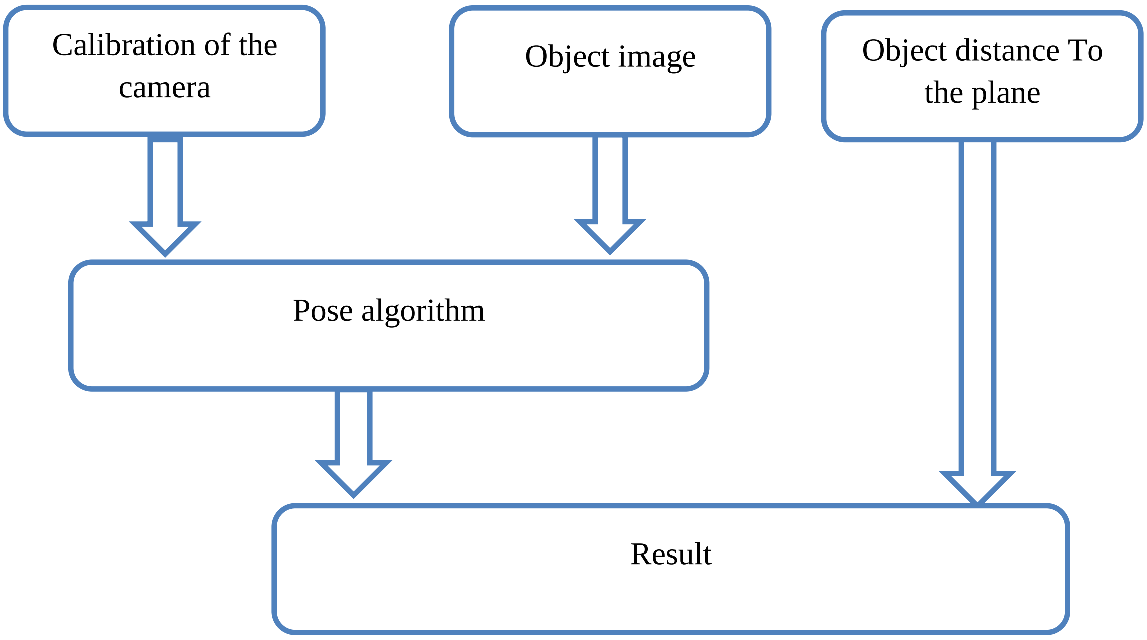


Figure 96 Diagram of position estimation

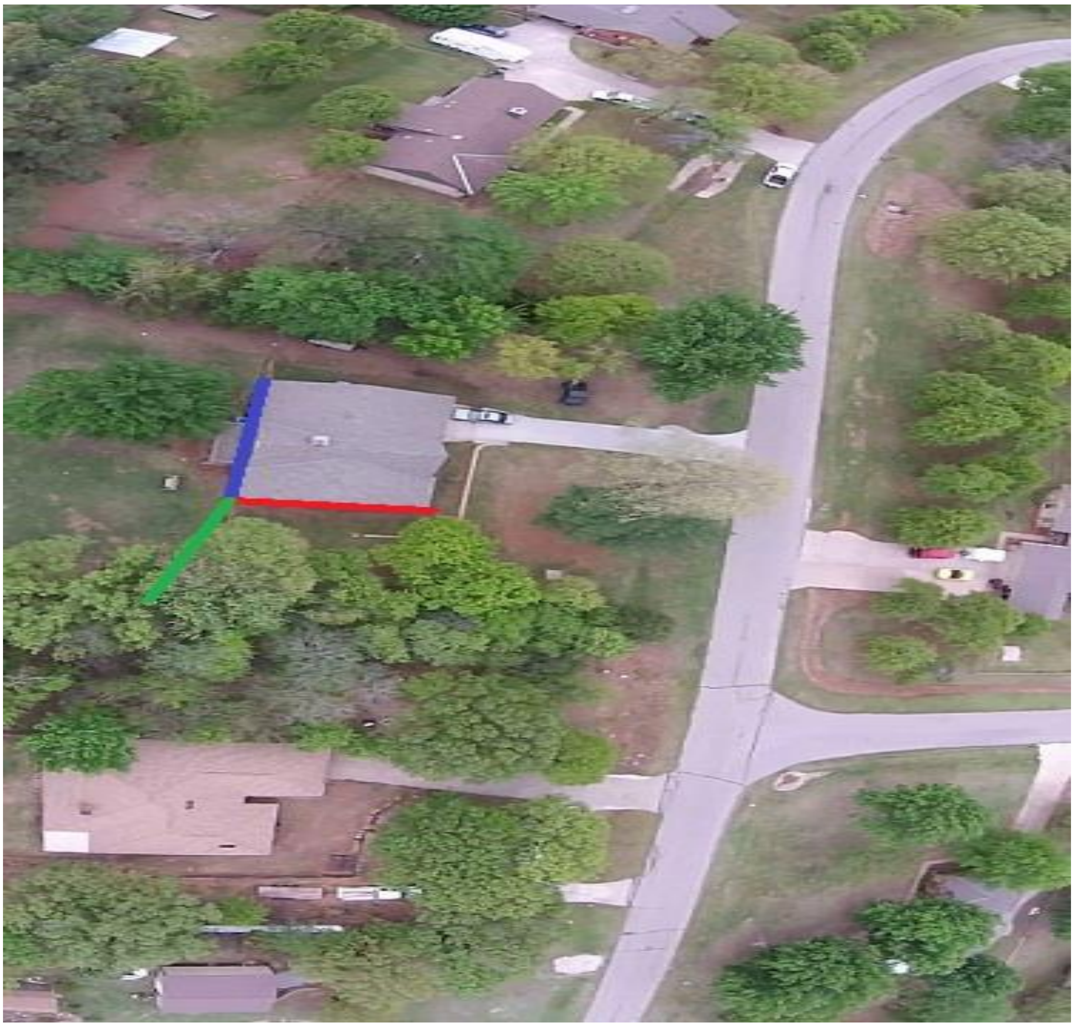


Figure 97 Pose algorithm application

Chapter 6

RESULTS

6.1 Results

In this work, a description and performance analysis of a new vision-based system is presented. This system offers local positioning based on a simplified binocular vision technique. The main testing aim was to recognize "landmarks" by comparing the images obtained by the system with those contained in a database. Tests were carried out against the SURF and FLANN matcher algorithms for many models, the results of which can be summarized as follows:

1- After testing five samples, the developed system was found to be slower but more accurate than the existing algorithms as illustrated below figure:

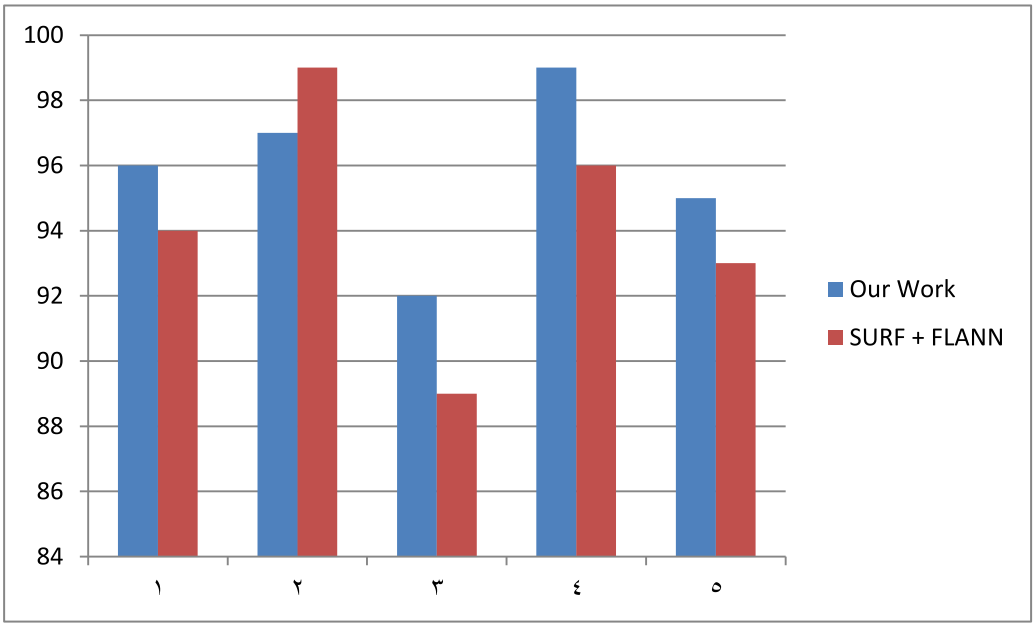


Figure 98 Comparison of the developed system vs SURF+FLANN

2- After testing five samples with the same texture but varying in size, whereas the developed system was able to recognize all of them, the existing algorithms failed to recognize any, as illustrated below figure:

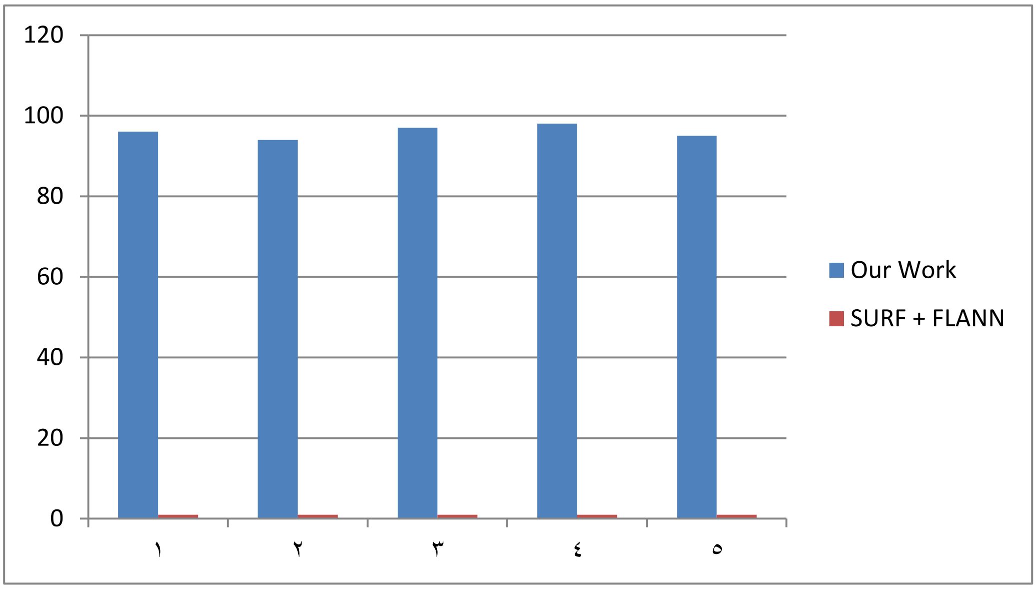


Figure 99 Comparison of sample recognition for the developed system vs SURF+FLANN

2- After testing five partial samples of which less than 70% each were captured, whereas the developed system failed to recognize any of them, the existing algorithms were able to recognize all of them, as illustrated below figure:

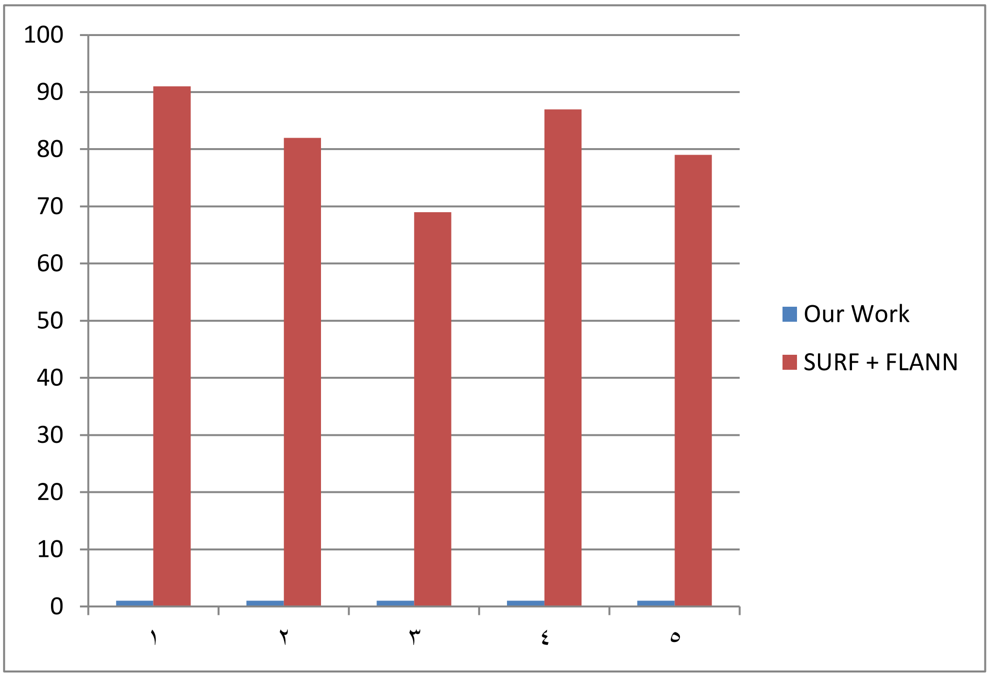


Figure 100 Comparison of partial-sample recognition for the developed system vs SURF+FLANN

3- After testing five landmarks of which only descriptions and no pictures were available, whereas the developed system succeeded in recognizing all of them, the existing algorithms were not able to recognize any, as illustrated below figure:

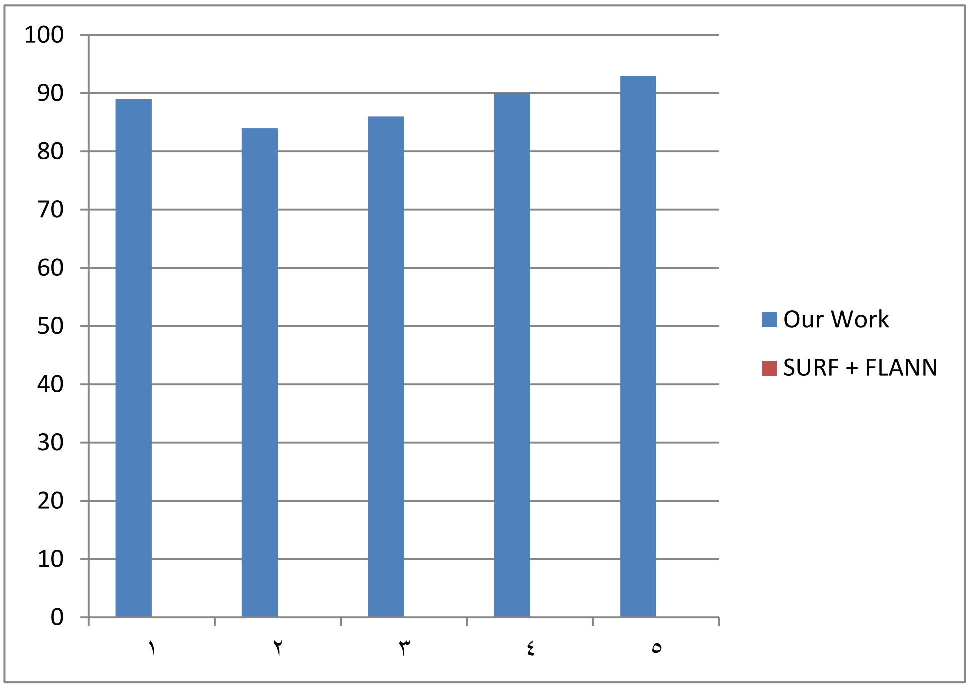


Figure 101 Comparison of sample description recognition for the devloped system vs SURF+FLANN

6.2 Conclusion

During this study we were able to convert an image into position information by passing through a number of steps, beginning with the extraction of camera parameters which were then used to determine the relationship between the imaged point in the real world and the point as shown in the image. High-accuracy measurement assisted in the conversion of each point in the image from image data into position information. The presented system enables the detection of target object “landmarks” for which no picture exists in the database. The developed method can be used not only by an autonomous robot but also by a human pilot.

6.3 Future Work

The main problem encountered in the present study was associated with extracting the landmarks from the image, with the result highly dependent on the segmentation step. Future work will involve converting the RGB cameras to thermal cameras, which will not only allow the system to be used at night, but also remove the need for the use of complex algorithms for color segmentation, because a thermal camera color system is more flexible, as illustrated below figure:

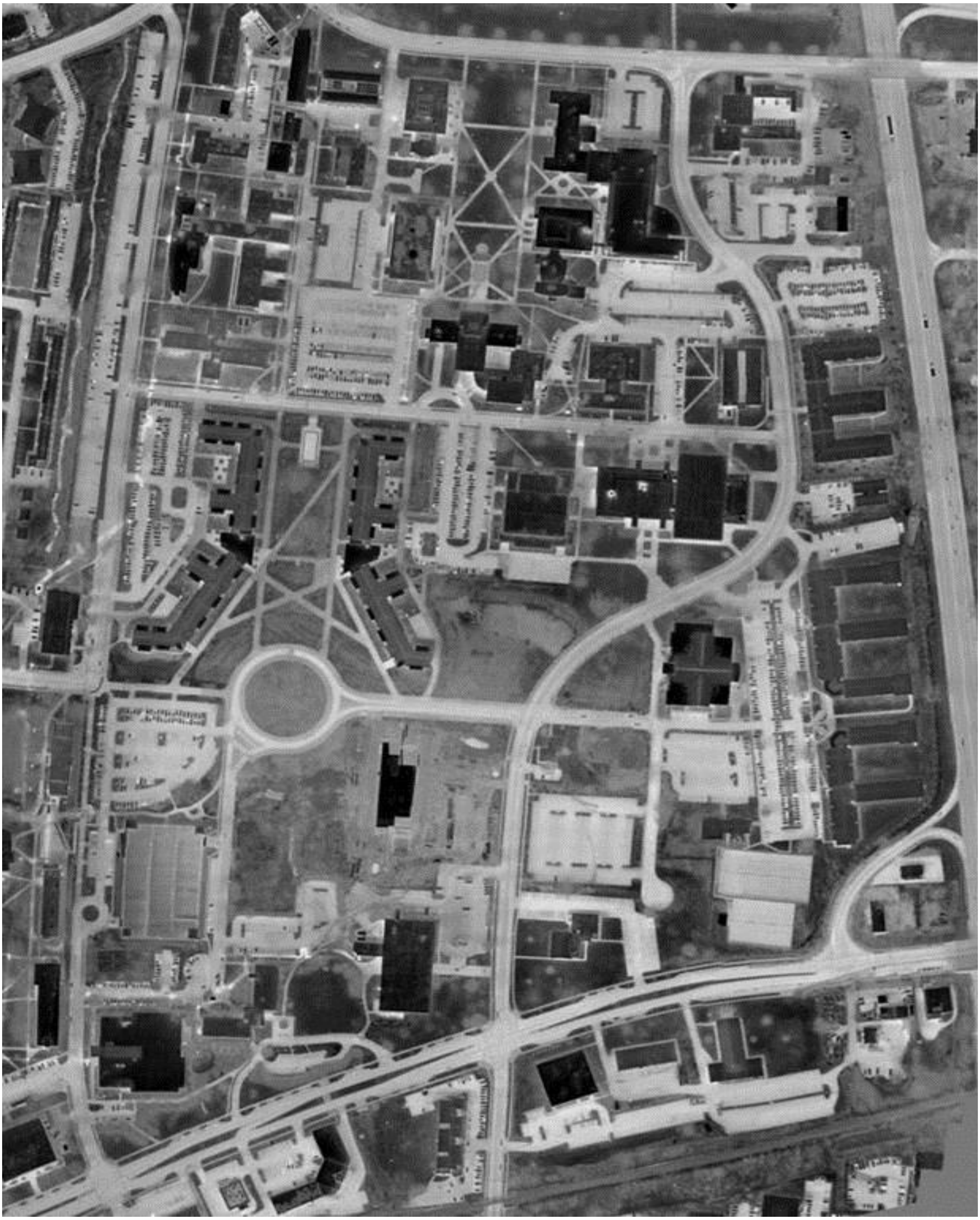


Figure 102 Aerial image of an urban area captured using an infrared camera

REFERENCES

1. **Pomerleau D. A., (1989)**, “ALVINN: An Autonomous Land Vehicle in a Neural Network”, Technical Report CMU-CS, pp. 89–107.

2. **Whitcomb** L.**, Yoerger D., Singh H.**, **(1999)**, “Advances in Doppler-Based Navigation of Underwater robotic Vehicles”, IEEE International Conference on Robotics and Automation. vol.1, pp.399-406 vol.1

3. National Space-Based Positioning, Navigation, and Timing Advisory Board, “Jamming the Global Positioning System – A National Security Threat: Recent Events and Potential Cures"." http://www.pnt.gov". (Data Download Date :11/02/2014).

4. **Conte G.**, **Doherty P.**, **(2008)**, “An Integrated UAV Navigation System Based on Aerial Image Matching”, IEEE Aerospace conference.

5. **Sakai A., Yuya T.**, **(2010)**, “Visual Odometry Using Feature Point and Ground Plane for Urban Environments”, Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK).

6. **CNN** news agency," http://edition.cnn.com/2013/12/tech/innovation/amazon-drones-questions",(Data Download Date: 03/06/2014).

7. **CNN** news agency, "http://arabic.cnn.com/scitech/2014/02/10/uae-drones", (Data Download Date: 03/06/2014).

8. **Case M., (1986)**, "Single Landmark Navigation by Mobile Robot", SPIE, Mobile Robots, vol. 727, pp. 231-238.

9. **McGillem C.D.**, **Rappaport T.S., (1988),** "Infra-red Location System for Navigation of Autonomous Vehicles", Proc. of the IEEE Int. Conf. Robotics and Automation. vol.3, pp.2190-2197

9. **Nasr H., Bhanu B.**, **(1988)**, Landmark Recognition System for Autonomous Mobile Robots. Proc. of the IEE Int. Conf. Robotics and Automation. pp.1019-1028 vol.2

10. **Sugihara K., (1988**), "Some Location Problems for Robot Navigation Using a Single Camera". *CVGIP(42),*No. 1, pp. 112-129.

11. **Hui Z., Lingtao Z., and Jing D.**, **(2012)**, "Landmark-Based Localization for Indoor Mobile Robots With Stereo Vision", Proc. IEEE Int. Conference on Intelligent Systems Design and Engineering Applications, pp. 700-702.

12. **Jang G.**, **(2005)**, "Metric Localization Using a Single Artificial Landmark for Indoor Mobile Robots", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 2857-2862.

13. **Moravec H.P.,(1981)**, "Robot Rover Visual Navigation". Ann Arbor, MI UMI Research press. pp.1198-1123 vol.2.

14. **Matthies L., Shafer S.A.**, **(1987)**, "Error Modeling in Stereo Navigation", IEEE Jour. of Robotics and Automation, vol. 3, pp. 239 – 248.

15. **Tamas L., Lazea G., (2008)**, "State Estimation Based on Kalman Filtering Techniques in Navigation", 2008 IEEE International Conference on Automation, Quality and Testing, Robotics, 2008 (AQTR 2008) .

16. **Faugeras O., Ayache N.**, (**1987)**, "Building, Registering and Fusing Noisy Visual Maps", Proc. of the first Int. Conf. Computer vision, pp. 73-82, London.

17. **Crowley J.L.**, (**1985)**, "Dynamic World Modeling for an Intelligent Mobile Robot Using a Rotating Ultra-Sonic Ranging Sensor", Proc. of the IEEE Int. Conf. Robotics and Automation, pp. 128-135, St. Louis.

18. **Chatila R., Laumond J-P.**, **(1985)**, "Position Referencing and Consistent World Modeling for Mobile Robots", Proc. of the IEEE Int. Conf. Robotics and Automation, pp. 138-145, St. Louis.

19. **Miller D.**, **(1985)**, "A Spatial Representation System for Mobile Robots", Proc. of the IEEE Int. Conf. Robotics and automation, pp. 122-127, St. Louis.

20. **Elfes A.**, **(1987)**, "Sonar-Based Real-World mapping and Navigation", IEEE Jour. Of Robotics and automation, vol. 3, no. 3, pp. 249-265.

21. **Courtney J., Magee M. and Aggarwal J.K., (1984)**, "Robot Guidance Using Computer Vision", Pattern Recognition, vol. 17, no. 6, pp. 585-592.

22. **Drake K.C., McCey E.S. and Inigo R.M.**, **(1987)**, "Experimental Position and Ranging Results for a Mobile Robot", IEEE Jour. of Robotics and Automation, vol. RA-3, no. 1.

23. **Kak A., Andress K. and Lopez-Abdia C.**, **(1989)**, "Mobile Robot Self-Location with the PSEIKI", Purdue University, Technical Report TR-EE 8 9-35.

24. **Tsuboushi T., Yuta S., (1989)**, "Map Assisted Vision System of Mobile Robots for Reckoning in a Building Environment", Proc. of the IEEE Int. Conf. Robotics and Automation, pp. 1978-1984, Raleigh NC.

25. **Lowe D.G., (1985)** "Perceptual Organization and Visual Recognition". Boston-London: Kluwer Academic publishers. pp.206-211.

26. **Shah M.**, **1997**, "Fundamentals of Computer Vision". Kluwer Academic Publishers, ISBN 0-7923-4618-1.

27. **Wikipedia**," http://en.wikipedia.org/wiki/Canny_edge_detector". (Data Download Date: 20/05/2014).

28. **Herbert B., Andreas E., Tinne T., Luc V., (2008)**, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), vol. 110, no. 3, pp. 346--359.

APPENDICES A

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Hazzaa, Omar

Date and Place of Birth: 16 September 1986, Kirkuk

Marital Status: Single

Phone: +90 5349130044

Email: omar.amil.hazzaa@gmail.com

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.Sc.	Çankaya University., Mathematical and Computer Science	2014
B.Sc.	Tikrit University., Mathematical and Computer Science	2008
High School	Tikrit Developed	2004

FOREIN LANGUAGES

Advanced English, Beginner Turkish,

HOBBIES

Programming, Travel, Books, Swimming, Robotic,