Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

Prediction of AR marker's position: A case of study using regression analysis with Machine Learning method

A dissertation presented by

Yazmín Sarahí Villegas Hernández

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Engineering Science
Major in Advanced Manufacturing

Monterrey Nuevo León, May 15th, 2017

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

The committee members, hereby, certify that have read the thesis presented by Yazmín Sarahí Villegas Hernández and that it is fully adequate in scope and quality as a partial requirement for the degree of Doctor of Philosophy in Engineering Science, with a major in Advanced Manufacturing.

Dr. Federico Guedea Elizalde
Tecnológico de Monterrey
Principal Advisor

Dr. Ciro Ángel Rodriguez González
Tecnológico de Monterrey
Committee Member

Dr. Héctor Rafael Siller Carrillo
Tecnológico de Monterrey
Committee Member

Dr. Eduardo González Mendívil
Tecnológico de Monterrey
Committee Member

Dr. Neale Ricardo Smith Cornejo
Tecnológico de Monterrey
Committee Member

Dr. Rubén Morales Menéndez,
Dean of Graduate Studies
School of Engineering and Sciences

Monterrey Nuevo León, May 15th, 2017

Declaration of Authorship

I, Yazmín Sarahí Villegas Hernández, declare that this thesis titled, "Prediction of AR marker′s position: A case of study using regression analysis with Machine Learning method" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Yazmín Sarahí Villegas Hernández
Monterrey Nuevo León, May 15th, 2017

©2017 by Yazmín Sarahí Villegas Hernández
All Rights Reserved

Dedication

To God.

Acknowledgements

I would like to thank Dr. Federico Guedea Elizalde for several reasons. Firstly,as my teacher, he motivated me to work hard for almost 10 years. Secondly, as my principal advisor, he taught me about intelligent systems and about robotic systems. Thirdly, he supported me with resources, materials, tools, software, and related journals. Fourthly, he gave me his invaluable knowledge, experience, and time. Fifthly,as a co-author, his advice and support with the publications and journals that were published. Sixthly, he helped me to define my thesis theme.

I would like to thank Dr. Héctor Rafael Siller Carillo for his advice and time to define my thesis theme.

I would also like to thank Dr. Ciro Ángel Rodriguez González for his support and access to resources for my research.

I would also like to thank Dr. Eduardo González Mendívil s for taking the time to review my thesis and advice to improve my research.

I would like to thank my family. My father to teach me everything I needed and his guidance. My mother for her love and support. My brother for his love, care and kindness. My sister for her advice, love, care, and guidance.

I would like to thank to God and every person that help me, in the past months, to have new dreams and to pursue them. Thanks for the motivation and inspiration that allowed me to continue and achieve my dreams.

Prediction of AR marker's position: A case of study using regression analysis with Machine Learning method

by

Yazmín Sarahí Villegas Hernández

Abstract

In an automated assembly process with robotic assistance, it is used vision system, which is used to monitor or control the assembly process. In the assembly process, the vision system recognizes objects and estimates the position and orientation. Furthermore, the optical tracking information in manufacturing can provide valuable support and time saving for autonomous operations, but ill environment conditions prevent a better performance of vision systems.

This thesis research presents a novel method for estimating object position under semi-controlled environment where lighting conditions change dynamically is proposed.

This method incorporates machine learning and regression analysis that combines light measurement and an augmented reality (AR) system. Augmented Reality (AR) combines *virtual objects* with *real environment*. Furthermore, every AR application uses a video camera to capture an image including a *marker* in order to place a virtual object, which gives user an enriched environment. Using a tracking system to estimate the *marker's* position with respect to the camera coordinate frame is needed to positioning a virtual object.

Most research studies on tracking system for AR are under controlled environment. The problem is that tracking systems for *markers* are sensitive to variations in lighting conditions in the *real environment*. To solve this problem, a method is proposed to estimate better a marker position based on regression analysis, where lighting conditions are taken into account. The proposed approach improves the accuracy of the marker position estimation under different lighting conditions.

List of Figures

1.1 Jittering problem 2
1.2 Machine learning process 5
1.3 Prediction process 6

2.1 Categorization of challenges in Augmented Reality 7
2.2 Classification of Augmented Reality Tracking 10
2.3 Circular marker [46] 12
2.4 ARToolKit marker [74] 12
2.5 Marker's coordinates and camera coordinates [74] 14

3.1 Machine learning types 20
3.2 Supervised learning 21
3.3 Unsupervised learning 22

4.1 Milgram's reality-virtuaity continuum 25
4.2 Coordinate system 26

5.1 Experimental setup 39
5.2 Experimental setup: CNC machine tool guide 40
5.3 Process for setting light range 41
5.4 Machine learning system for AR marker position estimation 42
5.5 Machine learning process 43
5.6 Process for finding the data distribution 44
5.7 Prediction process 45
5.8 Optimization process of predictor function 46

6.1 Experimental setup 48
6.2 Experimental results of position detection error on Sample 1 50
6.3 Experimental results of position detection error on Sample 2 51
6.4 Experimental results of position detection error on Sample 3 51
6.5 Experimental results of improvement of proposed approach and simple regression approach on Sample 3 53
6.6 Experimental results of improvement of proposed approach and simple regression approach on Sample 3 with underfitting 54
6.7 Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample 3 55

6.8 Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample 3 with underfitting 56
6.9 Experimental results of position detection error on the sample using a camera web Acteck 57
6.10 Experimental results of improvement of proposed approach and simple regression approach on Sample using a camera web Acteck 58
6.11 Experimental results of improvement of proposed approach and simple regression approach on Sample using a camera web Acteck with underfitting 59
6.12 Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample using a camera web Acteck 60
6.13 Experimental results of improvement % of proposed approach vs simple linear regression approach on the Sample with underfitting using a camera web Acteck 61
6.14 Experimental results of position detection error on the sample using a Kinect 62
6.15 Experimental results of improvement of proposed approach and simple regression approach on Sample using a kinect 63
6.16 Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample using a kinect 65
6.17 Experimental results of improvement of proposed approach and simple regression approach on Sample using a kinect with underfitting . 66
6.18 Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample with underfitting using a kinect 67

7.1 Light measurement system 71
7.2 Study case: Pieces to be assembled 72
7.3 Study case: Pick-and-place robot and camera system 72
7.4 Study case: Vision system interface 73

List of Tables

2.1 Tracking sensors 17
2.2 Tracking system approaches 17
2.3 Tracking system approaches 18

5.1 Evaluation of different algorithms 37

6.1 Light ranges 49
6.2 Safe distances for each operation 50
6.3 Results of fit and underfit models for marker′s position estimation . . 68

7.1 Tracking system approaches 73

F.1 Marker′s position estimation functions 111
F.2 Underfit marker′s position estimation functions 111

Contents

Abstract **v**

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem statement . . . 2
- 1.3 Augmented Reality market forecast . . . 2
- 1.4 Thesis statement . . . 3
  - 1.4.1 Methodology proposed . . . 4
  - 1.4.2 Scope of the thesis . . . 6

2 Literature Review **7**
- 2.1 Challenges in Augmented Reality . . . 7
  - 2.1.1 Performance challenges . . . 8
  - 2.1.2 Alignment challenges . . . 8
  - 2.1.3 Interaction challenge . . . 8
  - 2.1.4 Mobility challenges . . . 8
  - 2.1.5 Visualization challenges . . . 8
  - 2.1.6 Issues in Augmented Reality Tracking . . . 9
- 2.2 Tracking system in Augmented Reality . . . 9
  - 2.2.1 Sensor-based tracking . . . 9
  - 2.2.2 Vision based tracking . . . 12
  - 2.2.3 Hybrid tracking . . . 16

3 Machine Learning **20**
- 3.1 Machine learning definition . . . 20
- 3.2 Supervised Learning: Regression . . . 22
  - 3.2.1 Multiple linear regression . . . 23

4 Augmented Reality **24**
- 4.1 Augmented Reality: An overview . . . 24
- 4.2 Computer Vision Methods in Augmented Reality . . . 25
- 4.3 Marker's Tracking using ARToolKit . . . 27

4.4 Augmented Reality Devices 28
4.4.1 Displays 28
4.5 Input Devices 29
4.5.1 Tracking 29
4.5.2 Computers 30
4.6 Augmented Reality Interfaces 30
4.7 Collaborative Augmented Reality Interfaces 30
4.7.1 Hybrid Augmented Reality Interfaces 31
4.7.2 Multimodal Augmented Reality Interfaces 31
4.7.3 Augmented Reality Systems 31
4.7.4 ARToolKit 32
4.7.5 ARToolKit algorithm to superpose an object using a marker . . 32

5 Methodology **36**
5.1 Criteria of decision of evaluation of different algorithms to reduce marker's position error 36
5.2 Proposed solution 38
5.2.1 Initial preparation offline 38
5.3 Light range setting 40
5.4 Machine learning 42
5.4.1 Prediction using machine learning 44
5.4.2 Optimization of the models 44

6 Experimental setup and results **47**
6.1 Experimental setup 47
6.2 Light range selection 48
6.3 Standard distance for each operations 49
6.4 Experiments using a Logitech C920 camera 49
6.5 Experiments using a camera web Acteck CW-760 standard 56
6.6 Experiments using a Kinect 360 Xbox 61
6.7 Overfitting and Underfitting results 67

7 Conclusions **69**
7.1 Contributions 69
7.2 Initial questions 70
7.3 Limitations 71
7.4 Scope of Applicability 72
7.5 Comparative issues 73
7.6 Future research 73

A Optimal Linear Prediction **75**

B Optimal linear predictor **76**

C ARToolKit-Kinect code for marker's position estimation **77**

D Machine learning code for marker's position estimation 94

E Marker's position estimations and light measurements 101

F Marker's position estimation functions 110

G ARToolKit algorithm (to get position data) 112

H Machine learning algorithm (for training) 113

Bibliography 121

Chapter 1

Introduction

This dissertation presents a novel method to estimate marker's position under semi-controller environment where lighting conditions change dynamically.

In an automated assembly process with robotic assistance, the assembly process are made by robots where the human interaction is minimized. In order to reduce the human interaction, vision system are developed in order to control the assembly process. These vision systems are for measurement of the assembly pieces and for position estimation of the pieces. There are many ways to estimate the position of one piece as: recognizing the geometry of the piece, using sensors, etc. In this work, it is used QR code patterns on the assembly pieces in order to recognize the piece type and to estimate the position and orientation.

The optical tracking information in manufacturing can provide valuable support and time saving for autonomous operations, but ill environment conditions prevent a better performance of vision systems. Most research studies on tracking system for AR are under controlled environment. The problem is that tracking systems for *markers* are sensitive to variations in lighting conditions in the *real environment*.

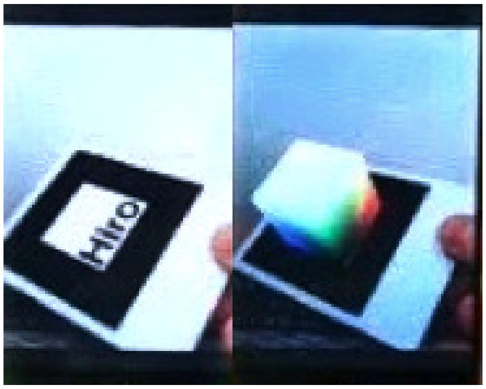
In this chapter, it is outline the main motivations of the research, the specific problem to be addressed, the methodology we are proposing, the scope of the thesis, and an outline of the dissertation.

1.1 Motivation

This research is addressed to reduce error of marker's position estimation under uncontrolled environment. The precision is needed in the following areas:

- In work spaces for automatized assembly process, it is needed vision systems for fast and easy detection of the assembly pieces. In these systems, it is needed precision in the detection process in order to grab objects, to place objects and to do any type of assembly in an automated way.
- In augmented reality is needed precision in the detection process of a marker in order to avoid *jittering* problems as shown on figure 1.1 (when the 3D object added appears and disappears because the marker is not detected).

Figure 1.1: Jittering problem



In the study area of *tracking* (position and orientation estimation) of a marker must be accurate. The main questions in this research are the following:

- How to reduce error or noise of the position estimation of a marker?
- Does the light variable affect the marker′s position estimation?
- How to measure the light variable that affects the marker′s position estimation?

1.2 Problem statement

Automated assembly consists of many components working together in order to do an assembly without or minimal human interaction. These components are: robots, automated table, vision system, controllers and a computer (there are many other components like sensors that are not taking into account because these will not be used in this research).

In automatized welding assembly for rectangular tubes, it is needed a robot system and a vision system at least. The vision system is used to detect pieces to be welded and to track the seam while the robot is welding. This research is focused on the detection of the piece in order to estimate its position and orientation, which is called *tracking*.

This tracking problem is focused on the detection of QR code markers that are over the rectangular tube (in order to be detected and estimated its position and orientation in an easy way). The problem consists on the accurate estimation of the position and orientation of the marker on the piece in order to manipulate and do an assembly to the piece using robots.

1.3 Augmented Reality market forecast

Augmented Reality (AR) market size [1] was over USD 570 million in 2015 with 80.8% compound annual growth rate (CAGR) estimation from 2016 to 2024. The areas of

AR applications are: automotive, medical, aerospace & defense, gaming, retail, industrial, and others.

Industrial applications accounted for over 25% global/world augmented reality marker share in 2015. Several types of applications for AR are developing in the industrial sector. The demand to handle complex machinery, assembly, maintenance and training will rise augmented reality industry growth.

Several companies such as Bosch, Boeing and Airbus invest to improve their manufacturing capabilities. For example, Boeing uses Google Glass to assist aircraft wire harnessing, Bosh uses augmented reality technology from stat-up Reflekt for its various applications including maintenance and Airbus uses Smart Augmented Reality Tools (SART) for error prevention. This technology is well used for quality inspection, training, work instruction and in the future is expected to be used extensively in development across industries.

Automotive augmented reality market size is projected to grow significantly with a compound annual growth rate (CAGR) of over 80% from 2016 to 2024. Augmented Reality used to improve user experiences using conventional showroom visit with virtual experience. For example, Ferrari created AR showroom app using 3D tracking technology that lets customers in the show rooms to choose vehicle and virtually change brakes, rims and paint job.

The marker forecast for augmented reality is predicted to grow significantly in Asia Pacific AR industry over the next few years. It is expected that the fonds increase and investment in technology would increase the demand of regional augmented reality technology. With the increment of penetration of smartphones & tablets and the amount of manufactures in Asia, it is anticipated that it contributes the rising augmented reality market size. The investment of local vendors and the acquisitions activities provide tremendous growth opportunities. Companies as Tencent Holdings Ltd. and Lenovo Group Ltd. joined to buy Silicon Valley augmented reality start-up Meta.

The forecast for America is that U.S. augmented reality market share will be the biggest in North America and it will drive regional industry. The forecast market for North America is going to exceed USD 24 billion by 2024. An increased adoption is forecast in the industrial and automotive sector in U.s, which are both set to grow faster than the rest of North America over the forecast time frame.

1.4 Thesis statement

In order to give a solution for an accurate estimation of the marker's position, a machine learning based methodology is proposed. This method is based on machine learning techniques. In this problem, it is used multiple lineal regression in order to fit the data and get a better estimation of the marker's position. The samples taken had a normal distribution, then a multiple lineal regression can be used. This technique is commonly used to fit data and understand the relationship between variables. Having this information, the estimations can be more accurate and reduce other errors like jittering, as shown in Figure 1.1 (when the 3D objects appear and disappear because the marker cannot be detected or is detected in other place by error).

The machine learning method uses a simple algorithm of multiple linear regression and optimization. This algorithm was written in C++. The proposed method is focused on the usage of machine learning and multiple linear regression.

1.4.1 Methodology proposed

This dissertation proposes a method of marker's position estimation under uncontrolled environment for augmented reality (where the lighting conditions change dynamically).

In this methodology, a previous work (offline) must be made before it's usage in order to estimate marker's position. This methodology consists of two phases: Firstly, the samples (of estimated marker positions under different light conditions) are taken and saved in a text file when the fiducial along the CNC machine tool guide is moving and the light measurement is taken with a luxmeter tool. It was used CNC machine tool guide in order to get the ground-truth position of the marker. The data is divided into three categories of light range and saved into three different files. Secondly, the data saved in a text file by the ARToolKit application is the input of the machine learning application written in c++, which made the regression analysis with the data.

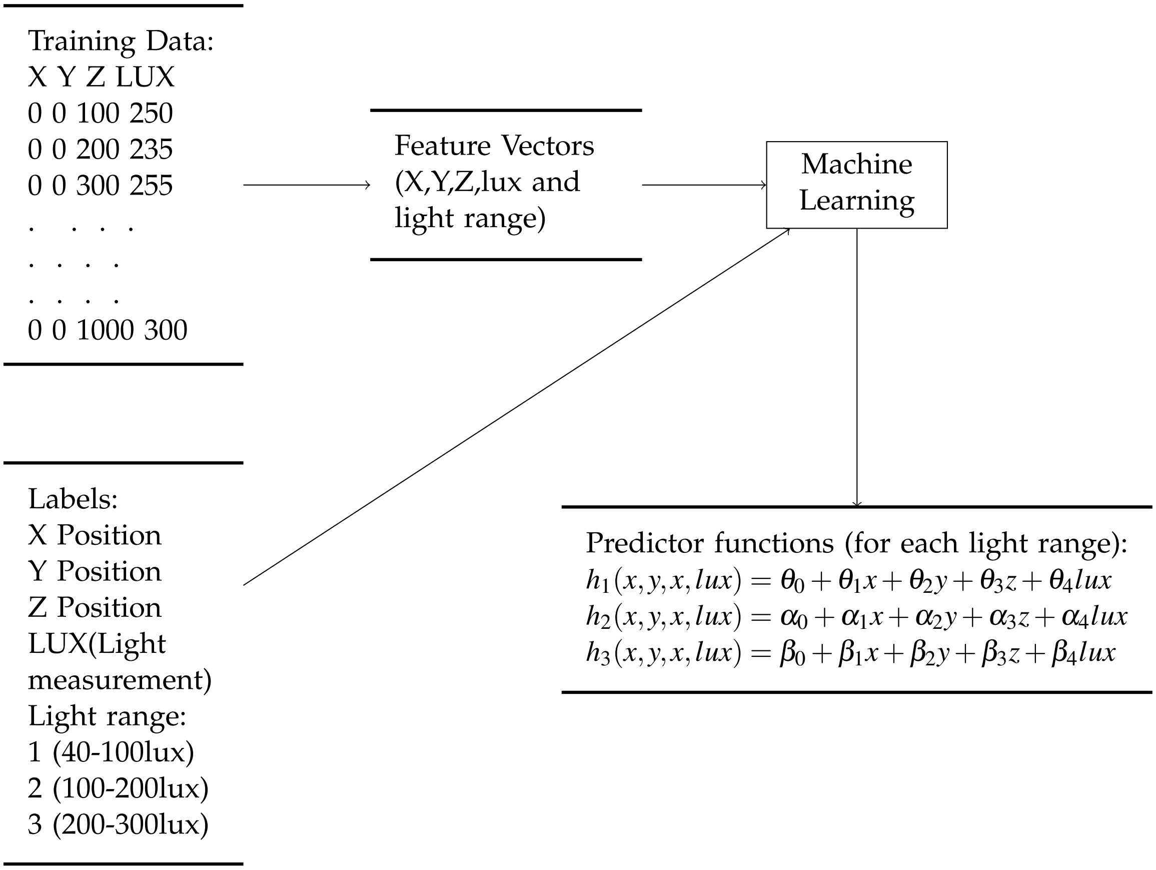
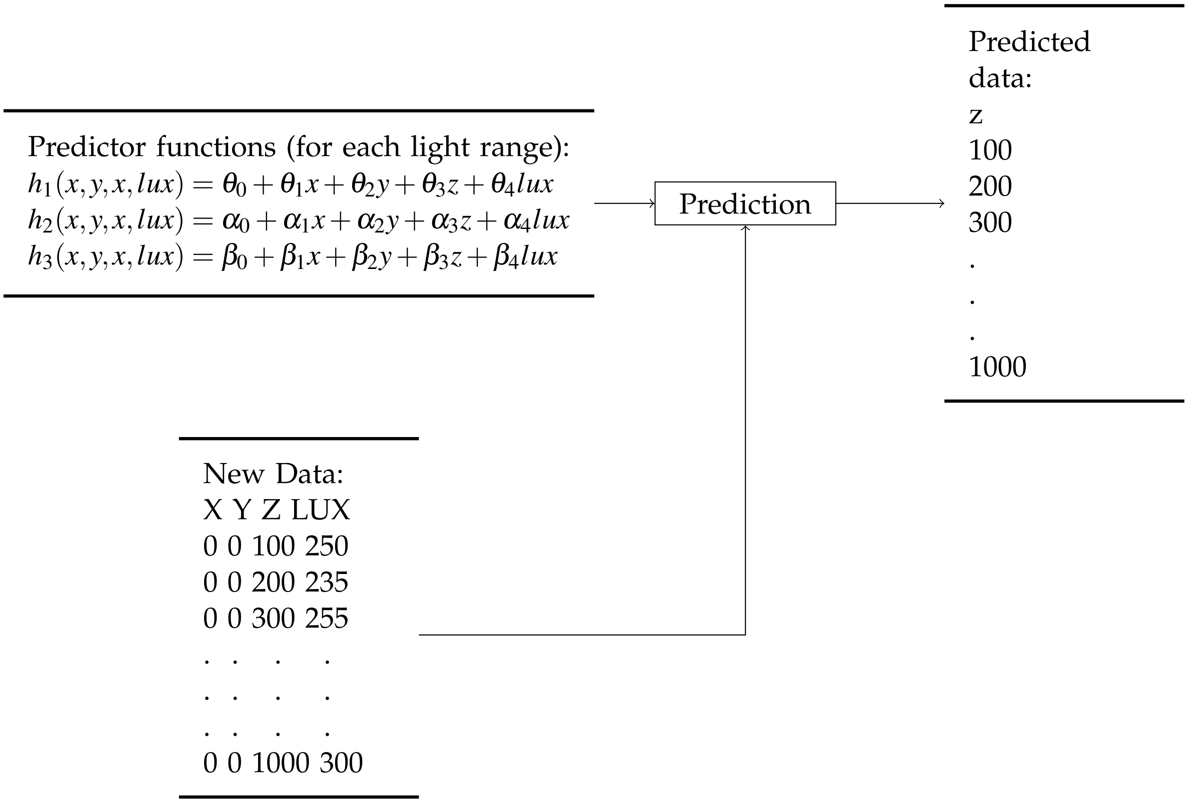

Training Data:
X Y Z LUX
0 0 100 250
0 0 200 235
0 0 300 255
. . . .
. . . .
. . . .
0 0 1000 300
Feature Vectors (X,Y,Z,lux and light range)
Machine Learning
Labels:
X Position
Y Position
Z Position
LUX(Light measurement)
Light range:
1 (40-100lux)
2 (100-200lux)
3 (200-300lux)
Predictor functions (for each light range):
$h_1(x,y,x,lux) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 z + \theta_4 lux$
$h_2(x,y,x,lux) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z + \alpha_4 lux$
$h_3(x,y,x,lux) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z + \beta_4 lux$

Figure 1.2: Machine learning process

Figure 1.2 shows the training process using multiple linear regression to generate a model for each range light.



Predictor functions (for each light range):

$h_1(x,y,x,lux) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 z + \theta_4 lux$

$h_2(x,y,x,lux) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z + \alpha_4 lux$

$h_3(x,y,x,lux) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z + \beta_4 lux$

Predicted data:
z
100
200
300
.
.
.
1000

New Data:

X	Y	Z	LUX
0	0	100	250
0	0	200	235
0	0	300	255
.	.	.	.
.	.	.	.
.	.	.	.
0	0	1000	300

Figure 1.3: Prediction process

Figure 1.3 shows the predictor functions for each light range, which were returned by the machine-learning application. These predictor functions are used to estimate the marker position.

The proposed methodology is limited by the scope of this thesis.

1.4.2 Scope of the thesis

The research scope covers how the light condition changes affect the detection algorithm. It also covers how much the light alter the marker detection algorithm. The research covers how the machine learning algorithm works in order to resolve this problem.

Yamauchi et al. [74] shows that the position errors (using ARToolKit libraries) in the x- and y- directions are relative small and that the errors in the z-direction are large. So, the z-position error is the only error variable that is going to be analyzed in this work. In these experiments, the fiducial was moving through z-direction from 220mm to 1060mm. In addition, the light conditions change dynamical from the range 41 lux to 303 lux. In this experiment, it was found that out the range of 41-303 lux the camera system cannot detect the marker.

Chapter 2

Literature Review

In the literature review of augmented reality, there are challenges or problems in the area of tracking. Furthermore, there are also several types of tracking systems in augmented reality that have different advantages and disadvantages.

Tracking systems for *markers* (*fiducials*) are used in AR applications. The tracking system can identify the *marker* using a camera in order to place the *virtual object* over the marker, and the *virtual object* must remain aligned with the position and orientation of the marker, which is called tracking. There are many kinds of markers, where each toolkit has its own marker type. Most research studies are about different tracking systems on ARToolkit or another toolkit. Actually, accurate registration and tracking between virtual objects and real world objects are crucial challenge in augmented reality.

2.1 Challenges in Augmented Reality

Tracking in Augmented Reality has different types of challenges, which can be categorized as performance challenges, alignment challenges, interaction challenges, mobility/probability challenges and visualization challenges as shown in Figure 2.1.

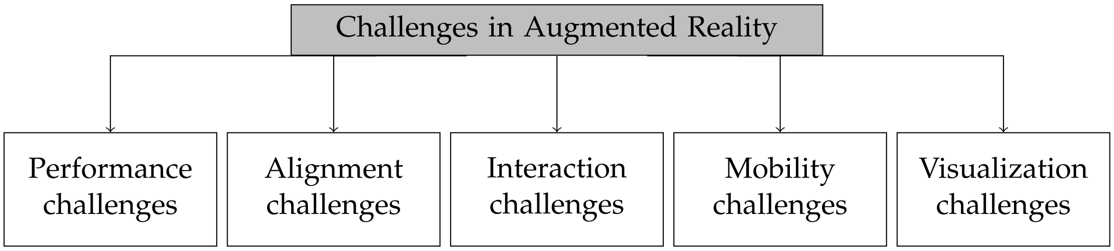


Figure 2.1: Categorization of challenges in Augmented Reality

2.1.1 Performance challenges

The performance challenges are concerned with real time processing of the AR application. The processing time of an AR application could be slow down the performance of the application. The mobile AR applications have this performance issue [75]. The visual recognition is computationally very expensive due to the 3D models used, and it's not due to complexity of the marker's design [68].

2.1.2 Alignment challenges

Alignment challenges are concerned with proper placement of a virtual object to the real world objects. The incorrect rendering of information is caused by the incorrect alignment. In the other case, registration is the proper alignment of virtual objects to the real world objects [30]. Tracking in outdoor for Augmented Reality is another problem related to alignment challenge [8]. Accurate tracking system is required in order to not cause a misalignment between virtual and real objects [71]. An accurate calibration is also required for having accuracy in augmented reality systems [26].

2.1.3 Interaction challenge

Interaction challenges refer to the interaction of users with virtual and real objects at the same time. The interfaces of interaction are: acoustic, haptic, tangible, gaze, or text-based through which the user interacts with virtual objects. The interaction with virtual object increases the amount of interaction problems [18]. There are many interaction techniques and user interfaces problems that need to be solved [77].

2.1.4 Mobility challenges

These challenges are concerned with the portability of augmented reality systems, which should be light and small so it can be used anywhere. The best type of augmented reality system is the system that is portable outside a controlled environment [9]. Schmalstieg et al. [59] developed a wearable system which needs to carry a whole set of heavy equipments for a long time.

2.1.5 Visualization challenges

Visualization challenges are concerned with the display issues (HMD based or monitor-based), contrast, resolution, brightness, and field of view. The illumination of the virtual object and real world object is required to be the same [22] [19]. Another visualization problem is occlusion, which is the process that determines which part of the image is not visible from certain view-point [71] [25]. Fischer et al. [21] developed a system for correct handling of occlusion between virtual objects and real world objects in the scene, which made a realistic view [37].

2.1.6 Issues in Augmented Reality Tracking

All tracking techniques have problems and none of them provides the best solution for the estimation of pose tracking in the outdoor unprepared environment. In unprepared out-door environment tracking is the main problem in an augmented reality system.

Drastic motions often lead tracking failures and recovery the tracking takes a lot of time, which leads to loss of real-time tracking [77].

In sensor-based tracking, it is used optical, magnetic, inertial, acoustic or ultrasonic sensors. The problem of optical tracking sensors is that these sensors are sensitive to optical noise, occlusion, they are computationally very costly and slow [76]. In optical tracking, it is difficult to track multiple objects in the scene at the same time.

The problem of magnetic sensors is that these sensors are disturbed by the presence of electronic devices nearby [64]. Magnetic sensors have also problems of *jitter*. This problem occurs because the accuracy degrades when their distance increases from the source and these sensors are sensitive to electromagnetic noise [76].

The problem of acoustic system is that it is a slowly system because the sound travels slow. The speed of sound in the air can change due to the change of temperature or humidity in the environment, which can affect the efficiency of the tracking system [64].

The problem of inertial tracking system is that the axis of wheel and bearing have a small friction.

The problem of hybrid systems is that it increases the complexity and the costs of tracking.

2.2 Tracking system in Augmented Reality

Zohu et al. [77] categorized the augmented reality tracking techniques in sensor-based, vision-based, and hybrid tracking techniques. Sensor-based tracking techniques are based on sensors as optical, magnetic, inertial, acoustic or ultrasonic that are placed in an environment (real world). Vision-based tracking [76] techniques used image information to track the position and orientation of a camera relative to the marker or other real object. Hybrid-based tracking techniques [10] combines several technologies in order to give a more accurate and robust solution for outdoor tracking. Figure 2.2 shows the classification of Augmented Reality Tracking.

2.2.1 Sensor-based tracking

In sensor-based tracking, it is used active sensors in order to track the position of camera movement relative to the marker or another real-world object. The sensors used for tracking are optical, magnetic, inertial, acoustic or ultrasonic. Each type of sensor has different accuracy, calibration, cost, engironmental, temperature and pressure, range and resolution.

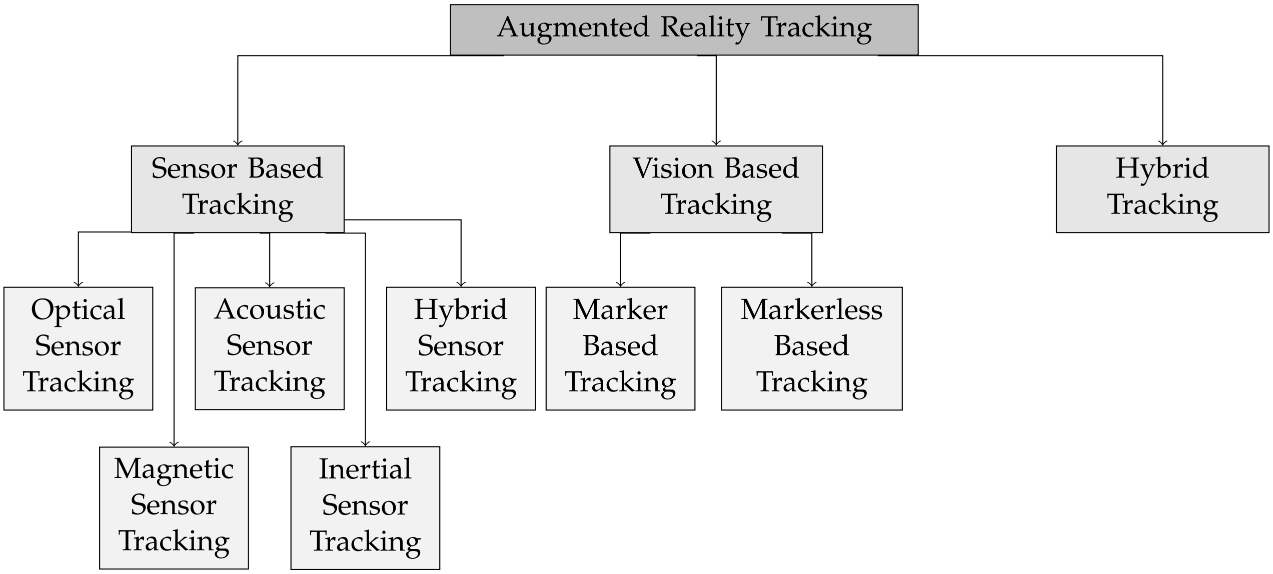


Figure 2.2: Classification of Augmented Reality Tracking

Optical tracking

In optical tracking system [64], a video camera using visible or infrared light to tracking a retro-reflective marker, which reflects the incoming infrared light back to the cameras. The infrared reflections are detected by the camera(s) and then processed by the optical tracking system in order to get the position and orientation of the camera(s). Using one camera, the system calculates the 2D marker position in camera coordinates. Using two cameras, the 3D position with 6DOF (Degrees of freedom) of each marker is computed. The cameras used are placed at different angles to view the target object (marker). The position and orientation of each camera is calculated using the epipolar geometry between two planes of images. The advantages of optical tracking are that is inexpensive and gives a more accurate and robust tracking in controlled environment. The disadvantages of optical tracking are that this tracking system is sensitive to lighting conditions and tracking is difficult when multiple objects (markers) are in the environment.

Magnetic tracking

In magnetic based tracking [76], there are used different magnetic fields. These sensors works as follows: first, the electric current is passed through coils (in the source), as a result magnetic field is created. The position and orientation of receivers are measured relative to the source. This tracking system is cheaper to implement, but it is less accurate than optical systems. Furthermore, magnetic tracking sensors have problems of jittering, accuracy (which degrades with distance), and sensitive to electromagnetic noise.

Acoustic tracking

In acoustic tracking system [64], it is used ultrasonic transmitters and acoustic sensors. The real object with ultrasound emitters and sensors are fixed in the environment. The position and the orientation of a user is calculated on the basis of time taken for sound to reach the sensors. The disadvantage of this tracking system is that acoustic tracking system is rather slow compared to other tracking sensors because the sound travels relatively slowly. Another disadvantage of this system tracking is that change of temperature or humidity in the environment, which can affect the efficiency of the tracking system.

Inertial tracking

In the inertial tracking system, it is used a mechanical gyroscope (to get the orientation of an object) and an accelerometer (to get the position of an object). Mechanical gyroscope system computes the orientation of the target using the rotational encoder angels and using the principle of conversation of the angular momentum. This system uses the axes of rotating wheel to have its own reference and is lightweight. The disadvantage of this system is that the problems of that can occur due to small friction between the axis of wheel and bearing. Accelerometer is used to measure the linear acceleration of an object with one degree of freedom.

Hybrid Sensor tracking

In hybrid tracking [56], it used a combination of various sensors. This combination of two or more sensors provide a better solution compared with the usage of only one sensor, but these hybrid systems increase the complexity and the cost of tracking. There are different types of hybrid tracking system as follows:

Auer and Pinz [6] built a hybrid tracking system by combining optical and magnetic tracking. In this tracking system, they use as main estimator of position and orientation of an object the magnetic sensor and the optical tracking sensor to refine the estimations in real-time. This hybrid tracking system is faster and reliable than an optical tracker and more precise than a magnetic tracker.

Maidi et al. [41] developed a hybrid approach for pose estimation, which mixes an iterative method based on the extended Kalman filter (EKF) and an analytical method with a direct resolution of pose parameters computation. This approach improves stability, convergence and accuracy of the pose parameters.

Dhiman et al. [16] developed a cooperative localization method (called mutual localization), which uses two cameras (each one with a fiducial (marker) in a sensor specific coordinate frame), in order to estimate the 6-Deegres of freedom pose of multiple cameras. Using this approach, it can be obviated the common assumption of sensor ego-motion. This approach uses an algebraic formulation to estimate the pose of the two-camera mutual localization setup under these assumptions. This approach can localize significantly more accurately than ARToolKit.

2.2.2 Vision based tracking

Vision-based tracking [77] is the most active area of research in augmented reality. Computer vision methods are used to calculate the camera pose relative to the real word objects (or markers). There are two types of approach in vision tracking: marker approach and markerless approach. The marker approach [47] consists on the usage of a fiducial (marker) where it is used to calculate the camera pose relative to the marker. The markerless approach [14] consists on the usage of the environment information to disappear the marker image of the environment.

Marker-based tracking

In marker-based tracking, markers (or fiducials) are placed placed in the environment or scene. These markers consist on patterns inside a black square. Naimark et al. [46] presented a circular 2D bar-coded fiducial system where the fiducial design allows having thousands of different codes, thus enabling uninterrupted tracking throughout a large building at very reasonable cost. This circular-shaped marker clusters with various parameters (number of markers, height, and radius) were developed as shown in Figure 2.3. The advantage of this marker frame configuration delivers excellent pose information, which translates to stable, jitter-free augmentation. ARToolKit, a type of library for augmented reality, has its own marker type, which are shown in Figure 2.4.

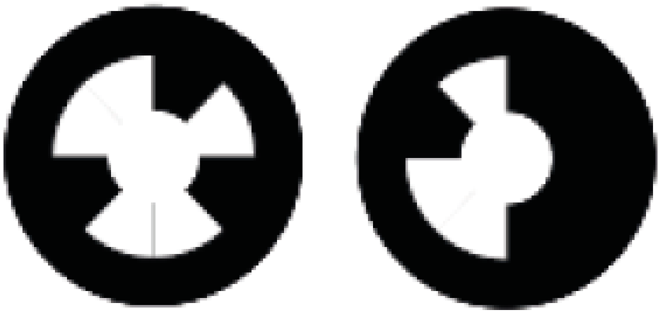


Figure 2.3: Circular marker [46]

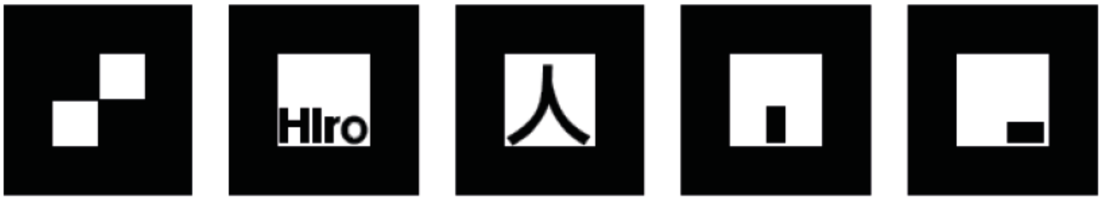


Figure 2.4: ARToolKit marker [74]

Ababsa et al. [5] developed a robust tracking system for Marker-based AR. The advantage of this system is that it works at a large range of distance and realibility under sever orientations.

Möhring et al. [45] developed a tracking solution for mobile phones, which tracks colour-coded 3D marker.

Steinbis et al. [61] presented a set of 3D *cone fiducials* for scalable indoor or outdoor tracking which are easy to *segment* and have a large working volume.

Maidi et al. [40] developed a system that combined extended *Kalman filter* and an analytical method with direct resolution of pose parameters computation. The advantage of this system is that it improves the stability, covergence and accuracy of those pose parameters.

Herout et al. [28] introduced an improved design of the Uniform Marker Fields and an algorithm for their fast and reliable detection. This marker field is designed to be detected and to be recognized for camera pose estimation: in various lighting conditions, under a severe perspective, while heavily occluded, and under a strong motion blur. This marker field detection harnesses the fact that the edges within the marker field meet at two vanishing points and that the projected planar grid of squares can be defined by a detectable mathematical formalism. The modules of the grid are gray-scale and the locations within the marker field are defined by the edges between the modules. The detection rates and accuracy are slightly better and faster compared with state-of-the-art marker-based solutions.

Rabbi et al. [53] extended the functionality of ARToolKit to a semi-controlled or uncontrolled environment using multiple files, which are recorded in different environmental conditions. This approach improved the marker tracking performance under different lighting conditions, brightness and contrast level. This approach may increase processing time, which is controlled by implementing a priority queue. This queue provides a priority to the pattern that is mostly used for tracking in the environment.

Yamauchi et al. [74] studied the position and pose error detected by an augmented reality system (using ARToolKit library). As shown in Figure 2.5, the marker is perpendicular to the line of sight of the camera. The characteristics of the marker detection are summarized as follows. The position of a marker is determined with sufficient accuracy for the directions perpendicular to the line of sight of the camera. The rotation angle of a marker around the line sight (of the camera) is also determined accurately. However, the position of the marker in the line of sight direction cannot be accurately determined. The detection error in this direction was revealed to be proportional to the square of the distance between the camera and the marker. The pose angles other than the rotation angle are also difficult to determine accurately.

Freeman et al. [24] proposed a method for predicting marker-tracking error in order to quantify the accuracy of the marker position. This statistical approach uses a modified Scaled Spherical Simplex Unscented Transform (SSSUT) algorithm in order to establish the maximum and minimum error of the marker-position estimations (estimated by the augmented reality system).

Wang et al. [70] presented a coarse-to-fine marker detection algorithm with

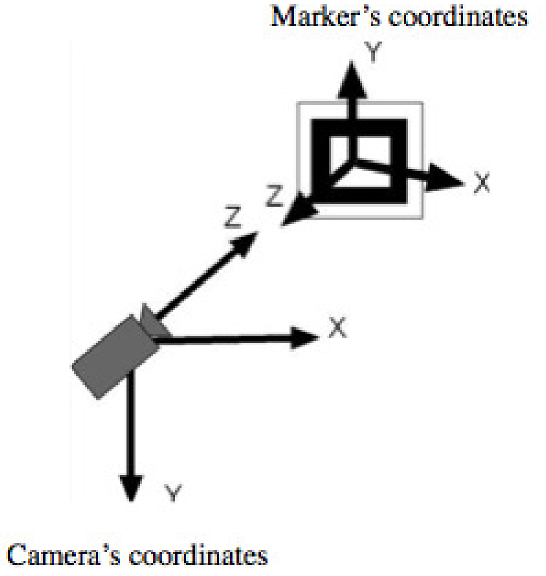


Figure 2.5: Marker's coordinates and camera coordinates [74]

sub-pixel edge localization. In this work, it is proposed a marker with a dot pattern, which is detected and matched to a predefined descriptor in a fast way using a simple threshold and hierarchical contour analysis. The algorithm yielded a feature detection error of less than 0.1 pixel (up to noise level $\sigma_n \leq 0.35$) with real-time performance.

Markerless-based tracking

Markerless tracking is when the marker-image detected is replaced with environment data; i.e. the marker is erased using image data that is near of the marker. Harris [27] described a markerless 3D visual tracking system called RAPiD (Real-time Attitude and Position Determination), which is the most popular and earlier markerless tracking system. The advantage of this technique is that it minimizes the amount of data that needs to be extracted from the video feed.

Park et al. [50] presented a method that allows *natural features* to be used for tracking instead of artificial features. This method uses known visual features, first the camera pose is estimated, then the system dynamically acquires additional natural features and uses them to a continuous update of the pose calculation. The advantage of this method is that it provides robust tracking even when the original fiducials are no longer in view.

Vacchetti et al. [65] combined edge and texture information to get a real-time 3D tracking. This system works as follows: first the interest points are found in the image for each frame. Second, these interest points are then matched with interest points of the reference frame, which are used for smooth camera trajectory.

Wuest et al. [73] presented a real-time model-based line tracking approach with adaptive learning of image edge features, which can handle partial occlusion and illumination changes. The advantage of this approach is that it uses a CAD model of the object for proper tracking in order to improve the robustness and efficiency of the system.

Reitmayr et al. [54] introduced a model-based tracking system for outdoor augmented reality in urban environments, which enabled accurate real-time overlays for a handheld device. This system combines several well-known approaches (i.e. an edge-based tracker for accurate localization, gyroscope measurements to deal with fast motions, measurements of gravity and magnetic field to avoid drift and a back store of reference frames with on-line frame selection to reinitialize automatically after dynamic occlusions or failures)

Ababsa et al. [3] introduced a markerless tracking for an outdoor augmented reality, which provided robust and reliable tracking using mobile handheld camera. The advantages that this system is efficient for partially known 3D scenes which combined edge-based tracker with a spare 3D reconstruction of the real-world.

Dame et al. [15] presented a direct tracking approach that uses *Mutual Information* as metric for proper alignment. The advantage of this approach is that it provides a robust, real-time and an accurate estimation of the displacement.

Sanchez et al. [57] presented a solution of real-time camera tracking and 3D reconstruction.

Park et al. [51] presented a method for improving the accuracy of 3DOF position and orientation for outdoor AR. The advantage of this method is that it uses corner points of buildings, detected as vertical edges in the image, and use it for refining GPS location and compass orientation.

Kim et al. [36] presented a real-time solution for modeling and tracking multiple 3D objects in unknown environments. The advantage of this system is that it can track 40 objects in 3D within 6 to 25 milliseconds.

Ababsa and Mallem [4] proposed particle filter framework with points and lines model-based tracking to achieve real-time camera pose estimation. The advantages of this implementation are simplicity and flexibility. It was showed that the algorithm can accurately track the camera pose successfully under sever occlusions and nonsmooth camera motions. A textureless object detection and 3D tracking with on-line training using a depth camera.

Park et al. [52] presented a textureless object detection and 3D tracking with on-line training using a depth camera. This method eliminates the requirement of prior object model, since any data for detection and tracking is obtained on the fly, which enhances the depth map.

Simon et al. [60] presented a promising method for markerless vision-based camera tracking for augmented reality applications tracking-by-synthesis. The advantage of this system is that it can run at high speed by combining fast corner detection and pyramidal blurring.

Donoser et al. [17] presented a real-time method to track weakly textured planar objects and to simultaneously estimate their 3D pose. The basic idea of this method is to adapt the classic tracking-by-detection approach, which seek for the object to be tracked independently in each frame, for tracking non-textured objects.

Ito et al. [32] presented a solution for the problem of the tracking performance deteriorated by viewing the plane to be tracked has a significantly oblique angle to the viewing direction or by moving object to a distant location from the camera. Ito et al. presented the solution of modeling the sampling and the reconstruction process of images. The main idea of this solution is that the template is corrected by applying a linear filter, which is generated by means of a tracked pose of the plane, and then using it for optimization, which tracks the plane in real time.

Lieberknecht et al. [38] presented a real-time method based on a consumer RGB-D camera that tracks the camera motion within an unknown environment. This system, While tracking, reconstructs a dense-textured mesh for it.

Uchiyama et al. [63] presented an approach for detection and tracking of different types of textures including colorful pictures, fiducial markers and hand writing.

2.2.3 Hybrid tracking

Each sensor-based tracking and vision based tracking has its own limitations or disadvantages. In order to have a robust tracking solution, hybrid methods have been developed. Hybrid tracking technique is the combination of both sensor-based tracking and vision-based tracking that attempts to compensate the shortcomings of each technique by using multiple measurements to produce robust results.

Hirota et al. [29] developed a hybrid tracking technique that combined vision based tracking (landmark tracking) and sensor based tracking (magnetic tracking).

Azuma et al. [10] described that not a single technology gives a complete solution for outdoor tracking, then a hybrid tracking technique was proposed for outdoor augmented reality system, which is based on GPS inertial and computer vision technologies.

Kanbara et al. [33] presented a hybrid system, which combines vision-based with inertial sensor. In this system, it is used vision-based approach for estimating the position and orientation of the camera by tracking markers in the real world environment, and inertial sensor is used to track stereo images and a camera orientation to produce a robust tracking system.

Foxlin et al. [23] developed a hybrid system, this system combines miniature MEMS (Micro Electro-Mechanical Systems) sensors to cockpit helmet-tracking for synthetic vision by inertial tracking between helmet-mounted and aircraft-mounted inertial sensors, and novel optical drift correction techniques. The advantage of this system is that it achieves a better position accuracy and angular accuracy with low latency.

Reitmayr et al. [54] developed a model-based hybrid tracking system for outdoor augmented reality in urban environments. The advantage of this hybrid tracking system is that it enables accurate and real-time overlays for hand-held devices. Hybrid tracking system combines edge-based tracker to track accurate localization, gyroscope measurements to deal with fast motions, measurements of gravity and magnetic field to avoid drift.

Bleser et al. [12] developed a hybrid tracking approach that combines SFM

Table 2.1: Tracking sensors

Sensors Tracking	Accuracy	Sensitivity	Cost	DOF	Advantages	Disadvantages
Optical Sensors	Accurate	Light	Cheaper	3/6 DOF	High update rate and better resolution	Effect with optical noise and occlusion
Magnetic Sensors	Less Accurate	Electronic Devices electromagnetic noise	Cheaper	6 DOF	No occlusion problem, high update rate	Small working volume, distance affect accuracy
Acoustic Sensors	Less Accurate	Temperature Humidity Pressure	Cheaper	3/6 DOF	Slow, Small, light, no distortion	Occlusion and ultrasonic noise
Inertial Sensors	Accurate	Friction	Cheaper	1/3 DOF	No reference needed, No prepared environment needed	Due to small friction conservation error
Hybrid	Accurate	Depend on sensors used	Costly	6 DOF	Compact, accurate stable	Depend on sensors used

Table 2.2: Tracking system approaches

Treatments	**Light changes**	**Method**	**Multiple cameras**
Rabbi's approach [53]	X	Multiple files	
Maidi's approach [41]		Statistical method	
Herout's approach [28]		Mathematical method	
Dhiman's approach [16]		Algebraic method	X
Yamauchi's approach [74]		Algebraic method	
Freeman's approach [24]		Statistical method	
Wang's approach [70]		Hierarchical contour analysis	
Our's approach	X	Statistical method	

(structure from motion), SLAM (Simultaneous Localization and Mapping) and model-based tracking.

Ababsa et al. [2] developed a hybrid tracker that combines optical sensor and vision based approach. This tracking system uses component-based framework that is designed for wide range tracker.

Schall et al. [58] introduced a 3DOF (Degrees Of Freedom) orientation tracking approach that combines the accuracy and stability of vision-based tracking with the correct orientation from inertial and magnetic sensors.

Waechter et al. [67] introduced a mobile multi-sensor platform to overcome the shortcomings of single sensor system. This platform combines an optical camera and mounted odometric measurement system that provides relative positions and orientations with respect to the ground plane.

Bleser et al. [11] presented a hybrid system that combines the egocentric vision with inertial sensors to track upper-body motion. In this hybrid system visual detectors of the wrists are used with the images of a chest-mounted camera to substitute the magnetometer measurements.

In Table 2.3, the differences between each approach described above are shown.

Most of these approaches used for marker detection do not take into account many factors as light intensity, brightness and contrast. Rabbi's approach takes into account light intensity, but this approach may increase processing time according to

Table 2.3: Tracking system approaches

Treatments	Light changes	Method	Multiple cameras
Ullah's approach [64]		Optical tracking system using infrared light	
Yang's approach [76]		Magnetic based tracking	
Ullah's approach [64]		Ultrasonic transmitters and acoustic sensors	
Ullah's approach [64]		Inertial tracking system using mechanical gyroscope and a accelerometer	
Auer's approach [6]		Hybrid tracking using optical and magnetic tracking.	
Maidi's approach [41]		Hybrid tracking using extended Kalman filter (EKF) and an analytical method with a direct resolution of pose parameters computation.	
Dhiman's approach [16]		Localization method	X
Zhou's approach [77]		Vision-based tracking using computer vision methods	X
Narzt's approach [47]		Marker's approach using	
Comport's approach [14]		Markerless approach	
Naimark's approach [46]		Marker approach using a new fiducial design	
Ababsa's approach [5]		Marker's approach for tracking	
Möhring's approach [45]		marker's tracking solution using a color-coded 3D marker	
Steinbis's approach [61]		Marker's approach using a novel 3D cone fiducial	
Herout's approach [28]		Algorithm for fast and reliable detection of markers using an improved design of the Uniform Marker Fields	
Harris's approach [27]		Markerless approach using 3D visual tracking called RAPiD	
Park's approach [27]		A method using natural features to be used for tracking	
Vacchetti's approach [65]		3D tracking using edge and texture information	
Wuest's approach [73]	X	Tracking system with adaptive learning of image edge features	
Reitmayr's approach [55]		Hybrid tracking using a edge-based tracker, gyroscope and measurements of gravity and a magnetic field	
Bleser's approach [12]		Hybrid system which combines SFM, SLAM and a model-based tracking	
Ababsa's approach[2]		Hybrid system that combines optical sensor and a vision based approach	
Schall's approach [58]		Hybrid system that combines 3DOF orientation tracking with vision-based tracking and magnetic sensors	
Waechter's approach [67]		Multi-sensor for mobile AR	
Bleser's approach [11]		Hybrid tracking that combines egocentric vision with inertial sensors	
Rabbi's approach [53]	X	Multiple files	
Maidi's approach [41]		Statistical method	
Herout's approach [28]		Mathematical method	
Dhiman's approach [16]		Algebraic method	X
Yamauchi's approach [74]		Algebraic method	
Freeman's approach [24]		Statistical method	
Wang's approach [70]		Hierarchical contour analysis	
Our's approach	X	Statistical method	

the number of pattern files. The main problem is the noise of the marker position estimation, which depends on the illumination and the rotation of the marker. In this paper, it is proposed to use machine learning in order to find the correlation between lighting conditions and position estimation of the marker.

Chapter 3

Machine Learning

For the past years, machine learning has been used for "learning" or analyze large amounts of data and draw conclusion. Using machine learning, it is possible to do better estimations and find patterns and tendencies, which could be used in different area as: finance, healthcare, industry, and more. This chapter explains what is machine learning and the different types of machine learning. In this chapter, it is also explained the multiple linear regression, which is a technique used in machine learning.

3.1 Machine learning definition

Machine learning is a subfield of Artificial Intelligence, where algorithms are used to learn in order to behave more intelligently rather than just storing and retrieving data like data systems do. Machine learning is a set of methods that can detect patterns in data, and then use these uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty.

As shown in figure 4.1, machine learning is divided in three main types: **supervised, unsupervised and reinforcement learning**.

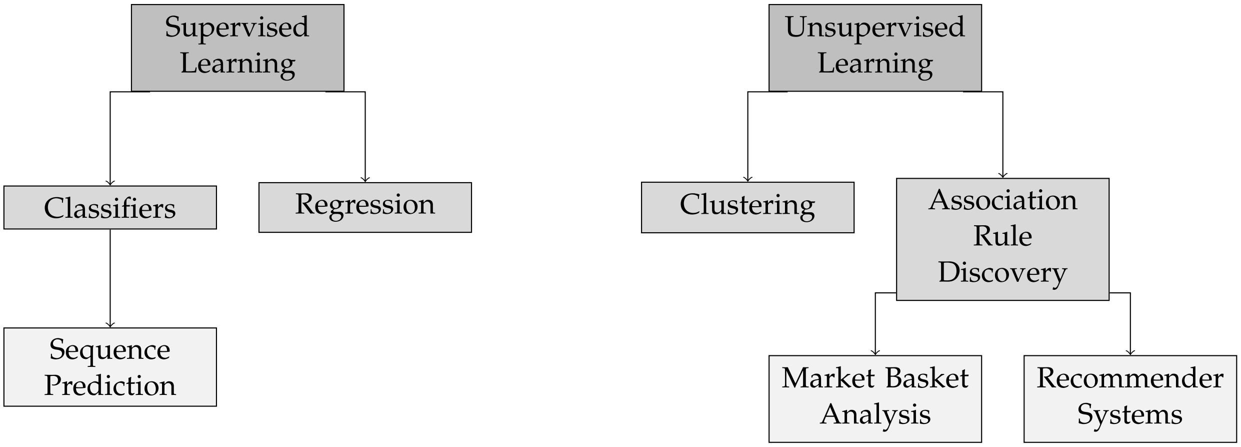


Figure 3.1: Machine learning types

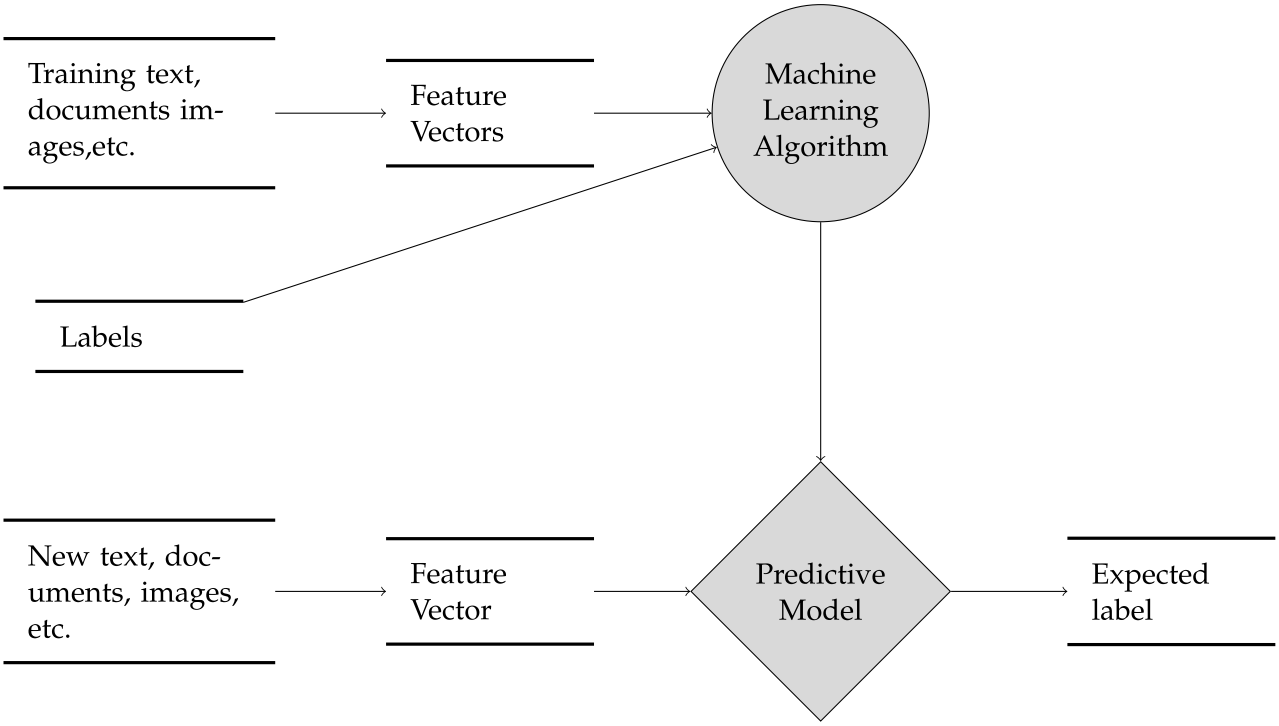

Training text, documents images,etc.
Feature Vectors
Machine Learning Algorithm
Labels
New text, documents, images, etc.
Feature Vector
Predictive Model
Expected label

Figure 3.2: Supervised learning

In **supervised or predictive** learning approach, the goal is to learn a mapping from a set of inputs (called **x**) to a set of outputs (called **y**), given a **labeled** set of input-output pairs $\mathbf{D} = (x_i, y_i)_{i=1}^{N}$, where **D** is the training set and **N** is the number of training examples.

The **training** input $\mathbf{x}_i$ is called **feature**, **attribute** or **covariate**, which could be a D-dimensional vector of numbers representing real-valued scalar variables as height and weight of a person, or it could be a complex structured object (such as image, a sentence, an email message, a time series, a molecular shape, a graph, etc.).

The **training** output $\mathbf{y}_i$ is called **response variable**, which could be **a real-valued scalar** (such as a vector of numbers) or a **categorical or nominal** variable from some finite set $\mathbf{y}_i \in 1, \ldots, C$ (such as male or female).

In **supervised** or **predictive** learning there are two main types of problems: **classification** (or **pattern recognition**) and **regression** problem. When the $\mathbf{y}_i$ is **categorical**, the problem is known as **classification** or **pattern recognition**. When the $\mathbf{y}_i$ is real-valued scalar, the problem is known as regression. Furthermore, another variant of **regression** problem, called **ordinal regression**, occurs when the training output $\mathbf{y}_i$ has some natural ordering such as grade A-F.

As shown in figure 3.2, the training text, documents, images, etc., and the feature vectors are the input for the machine learning system. This system generates a **predictive model** or a **predictor function**. This function receives new text, documents, images, etc., and feature vectors in order to generate a likelihood of cluster ID or a better representation.

In **unsupervised** or **descriptive** learning approach there are not a **labeled** set of

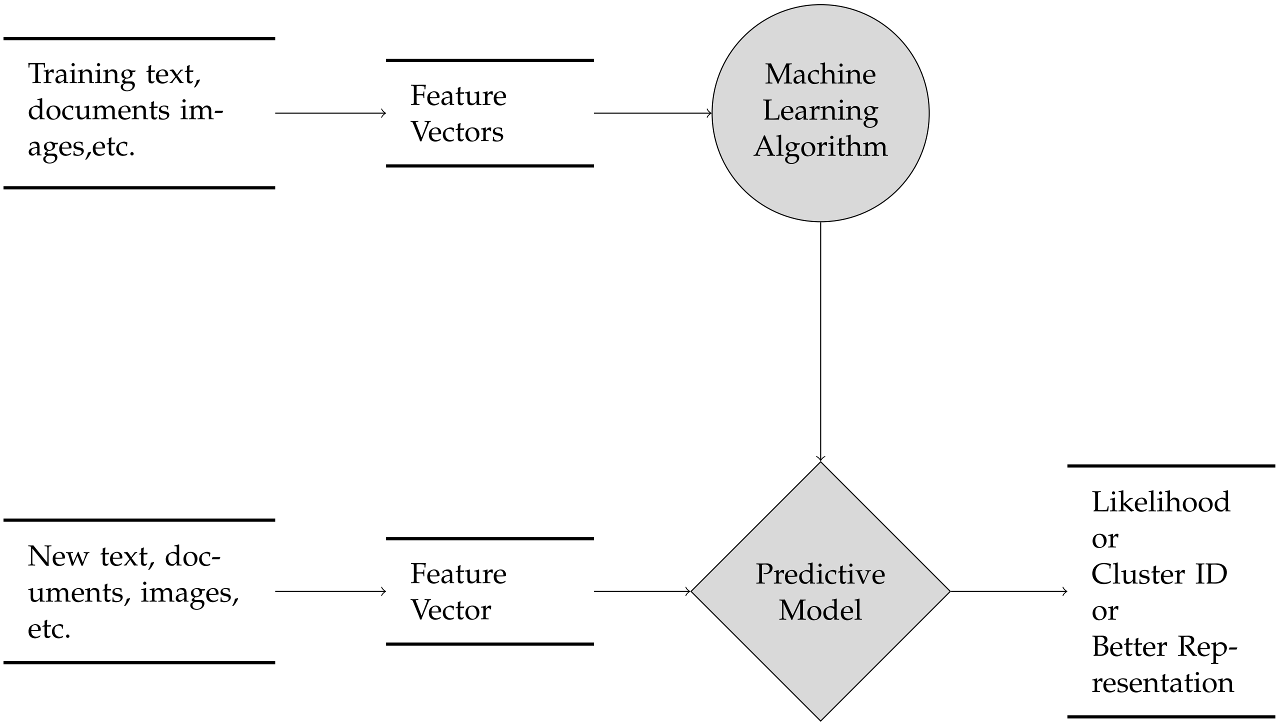

Training text, documents images,etc.
Feature Vectors
Machine Learning Algorithm
New text, documents, images, etc.
Feature Vector
Predictive Model
Likelihood or Cluster ID or Better Representation

Figure 3.3: Unsupervised learning

input-output pairs $\mathbf{D} = (x_i, y_i)_{i=1}^{N}$. In this approach, the goal is to find "interesting patterns" in the data, given a set of training inputs ($\mathbf{D} = (x_i)_{i=1}^{N}$). This problem is called knowledge discovery. Without knowing what patterns it is looking for and without an obvious error metric to use, this problem is much less well-defined problem than supervised problem.

As shown in figure 3.3, the training text, documents, images, the feature vectors, and the *labels* are the input for the machine learning system. This system generates a **predictive model** or a **predictor function**. This function receives new text, documents, images, etc., and feature vectors in order to generate an expected *label*.

In **reinforcement learning**, the goal is to learn how to act or behave when given occasional reward or punishment signals.

For example, ML can be used to detect the patterns of some images given. If we have two types of image (star and diamond), then the ML system will detect the two patterns.

3.2 Supervised Learning: Regression

Regression analysis approach is used when one or more predictor variables have a correlation with the response variable. The optimal regression coefficients depend on both the marginal distribution of the predictors and the joint distribution (covariances) of the response and the predictors.

The regression coefficient is interpreted as the estimation of how much the response would change, if the independent variable were increased by one unit, holding the other predictor variables constant.

Regression analysis is used for probabilistic prediction. It is selected a sub-set of the observations in order to make estimation of the whole population. A casual or counter-factual prediction is an estimation done when the predictor variables are on fixed values. Regression in machine learning works as a tool for causal interference.

3.2.1 Multiple linear regression

In machine learning, the most common technique used, depending of the problem, is the multiple linear regression. Multiple linear regression is used to predict the value of a variable **y** (called criterion variable) using multiple variables $(\mathbf{x_i}, ..., \mathbf{x_m})$ (called predictor variables). These predictor variables could be known or unknown. When the predictor variables are known, it is called *supervised machine learning* and the predictor variables are called *labeled* variables. When the predictor variables are unknown, it is called *unsupervised machine learning*. This leads to the following multiple regression function:

$$h(x_1, ..., x_m) = \Theta_0 + \textstyle\sum_{i=1}^{m} \Theta_i x_i \tag{3.1}$$

where Θ_0 is called the intercept and the Θ_i are called coefficients.

The process of learning consists on using mathematical algorithm in order to optimize the predictor function $\mathbf{h}(\mathbf{x_i})$, which is an estimation of the criterion variable **y**. For the optimization process, it is used training examples, which are input data of both predictor variables $(\mathbf{x_1}, ..., \mathbf{x_m})$ and criterion variable **y**, which is known in advance.

The loss function, as shown in Equation 3.2, is used to measure the improvement of the predictor function $h(x_{t,i})$. The input Θ represents all of the coefficients that we are using in the predictor function. In this case, it is used only two coefficients Θ_0 and Θ_1.

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_{t,i}) - y)^2 \tag{3.2}$$

The loss function and training examples are used to know the difference between the criterion variable **y** and the estimated value $\mathbf{h}(\mathbf{x_1}, ..., \mathbf{x_m})$, which is called *error*, in order to measure the improvement of the predictor function. In the optimization process, the intercept Θ_0 and the coefficients Θ_i are changed constantly in order to converge on the best values that minimize the *error*.

Chapter 4

Augmented Reality

Augmented Reality (AR) has been used in the industry for many usage types. The definition of augmented reality and the differences between real environment, augmented virtuality, virtual reality and augmented reality are given in this chapter. The computer vision methods used in augmented reality for rendering and compute the pose for the virtual object are explained in this chapter. The marker's tracking using ARToolKit it is also explained in this chapter. Furthermore, the different types of augmented reality devices and interfaces are also explained in this chapter.

4.1 Augmented Reality: An overview

Augmented Reality (AR) is defined as a view of real-word environment that has been *augmented* by adding virtual computer-generated models in real-time. AR combines real and virtual objects.

Miligrams et al. [43] explains the difference between real environment, augmented reality, augmented virtuality and virtual environment. Milgram's Reality-Virtuality Continuum is defined by Paul Milgram et al. as a continuum that have the real environment and the virtual environment and the combination of these two, which are Augmented Reality (AR) and Augmented Virtuality (AV). AR is closer to the real world and AV is closer to a pure virtual environment, as seen in figure 4.1.

Augmented Reality brings virtual information to immediate surroundings and indirect view of the real-world environment, such as live-video stream. AR technology *augments* the sense of reality by superimposing virtual objects on the real environment in real time.

Azuma et al. [7] did not consider AR to be restricted to a particular type of display technologies. AR can potentially apply to all senses, augmenting smell, touch and hearing as well.

Azuma et al. [7] considered that AR applications require removing real objects from the environment, which are more commonly called *mediated* or *diminished reality*.

Indeed, removing objects from the real world corresponds to covering the object with virtual information that matches the background in order to give the user the

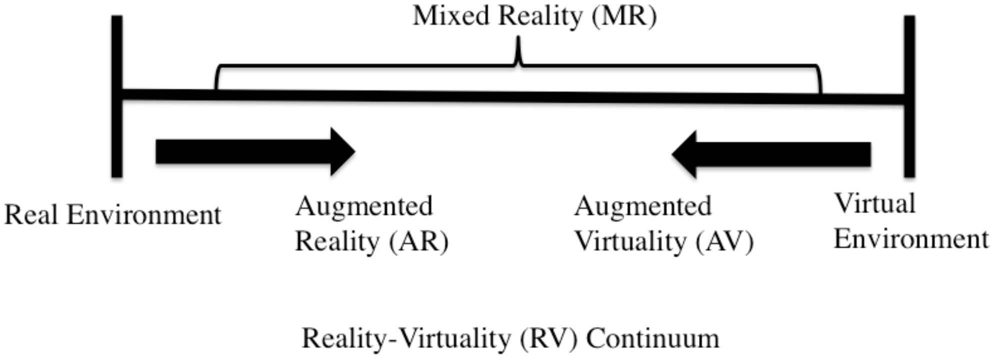


Figure 4.1: Milgram's reality-virtuaity continuum

impression that the object is not there. Virtual objects added to the real environment show information to the user that cannot directly detect.

4.2 Computer Vision Methods in Augmented Reality

The 3D virtual objects are rendered using computer vision methods. The orientation and position of these virtual objects depend on the pose of the fiducial marker, optical image or interest point where the virtual object is going to be placed. These computer vision methods are called **tracking** and **reconstructing/recognizing**.

In **tracking method**, the fiducial markers, optical images, or interest points are detected in the camera images. This method can make use of feature detection, edge detection, or other image processing methods to interpret the camera images. There are two types of tracking: feature-based and model-based. **Feature-based methods** consist of estimating the position and orientation of the 2D image features in the 3D world frame coordinates. **Model-based methods** make use of model of the tracked objects' features such as CAD models or 2D templates of the item based on distinguishable features.

Using the position and orientation of the 2D image in the 3D world frame, it is possible to find the camera pose (camera's position and orientation) for projecting the 3D coordinates of the feature into the observed 2D image coordinates and by minimizing the distance to their corresponding 2D features. The constraints for camera pose estimation are most often determined using point features. The **reconstructing/recognizing** stage uses both the orientation data and position data obtained from the first stage to reconstruct a real world coordinate system. Assuming a calibrated camera and a perspective projection model, if a point has the coordinates $(x, y, z)^T$ in the coordinate frame of the camera, its projection onto the image plane is $(x/z, y/z, 1)^T$.

In point constraints, we have two principal coordinate systems, as illustrated in Figure 4.2, the world coordinate system **W** and the 2D image coordinate system. Let

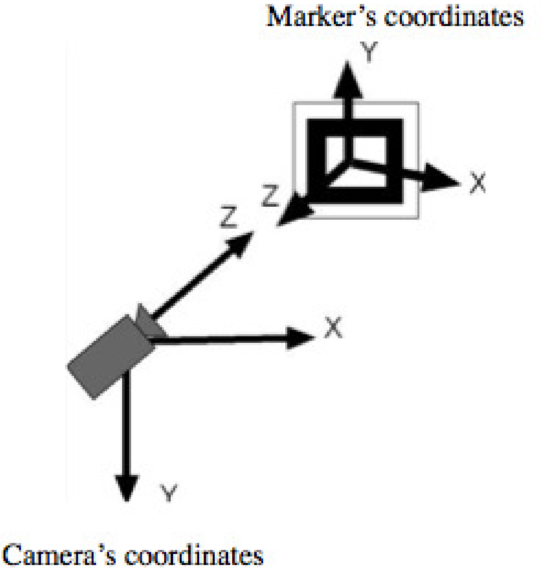


Figure 4.2: Coordinate system

$p_i(x_i, y_i, z_i)^T$, where $i = 1, ..., n$, with $n \geq 3$, be a set of 3D non-collinear reference points in the world frame coordinate and $q_i(x_i^{'}, y_i^{'}, z_i^{'})^T$ be the corresponding camera-space coordinates, p_i and q_i are related by the following transformation:

$$q_i = Rp_i + T \tag{4.1}$$

where

$$R = \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \ and \ T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{4.2}$$

are rotation matrix and a translation vector, respectively.

Let the image point $\mathbf{h}_i(u_i, v_i.1)^T$ be the projection of $\mathbf{p}_i$ on the normalized image plane. The *collinearity equation* establishing the relationship between h_i and p_i using the camera pinhole is given by:

$$h_i = \frac{1}{r_3^T p_i + t_z}(Rp_i + T) \tag{4.3}$$

The **image space error** gives a relationship between 3D reference points, their corresponding 2D extracted image points, and the camera pose parameters, and corresponds, to the point constraints. The image space error is given as follow:

$$E_i^P = \sqrt{\left(\hat{u}_i - \frac{r_1^T p_i + t_x}{r_3^T p_i + t_z}\right)^2 + \left(\hat{v}_i - \frac{r_2^T p_i + t_y}{r_3^T p_i + t_z}\right)^2} \tag{4.4}$$

where $\hat{m}_i(\hat{u}_i, \hat{v}_i, 1)^T$ are observed image points.

Huang et al. [39] use the scene 3D structure pre-calculated beforehand in order to know where to place a virtual object, where the device will have to be stationary and its position known. In case the entire scene is not known beforehand, Simultaneous Localization And Mapping (SLAM) technique is used for mapping fiducial markers or 3D models relative positions. In the case when no assumptions about the 3D geometry of the scene can be made, Structure from Motion (SfM) method is used. SfM method can be divided into two parts: **feature point tracking** and **camera parameter estimation**.

In the case the AR device is mobile and designed for an outdoor environment, the tracking techniques used are different from the techniques explained above. Nilsson et al. [48] built a pedestrian detection system for automotive collision avoidance using AR. This system is mobile and outdoor. For a camera moving in an unknown environment, the problem for computer vision is to reconstruct both the motion of the camera and the structure of the scene using the image and additional sensor data sequences. In this case, since no assumption about the 3D geometry of the scene can be made, SfM method is used for reconstructing the scene.

ARToolKit [34] is a set of AR libraries, which was developed in 1999 by Hirokazu Kato from the Nara institute of Science and Technology and was released by the University of Washigton HIT Lab. ARToolKit is a computer vision tracking library that allows the user to create augmented reality applications. It uses video tracking capabilities to calculate in real time the real camera position and orientation relative to physical markers. Once the real camera position is known, a virtual camera can be placed at the exact same position and 3D computer graphics model can be drawn to overlay the markers. The extended version of ARToolKit is ARToolKitPlus, which added many features over the ARToolKit, notably class-based APLIs; however, it is no longer being developed and already has a successor: Studierstube Tracker [69].

Studierstube Tracker's [69] is a set of AR libraries. It supports mobile phone, with Studierstube ES, as well as PCs, making its memory requirements very low (100KB or 5-10% of ARToolKitPlus) and processing very fast (about twice as fast as ARToolKitPlus on mobile phones and about 1 ms per frame on a PC).

4.3 Marker's Tracking using ARToolKit

The marker's tracking process is as follows: first, the marker image is converted to a black and white binary image, which is called *threshold*, using a threshold value (in order to know if the pixel is going to be black or white). Second, the black square of the marker is detected. Third, the position of the four corners of the marker are estimated, which are used to estimate the center position of the marker as well marker pose.

ARToolKit uses both functions *arDetectMarker* and *arGetTransMat* to detect and to estimate the position and pose of a marker. The *arDetectMarker* function detects the black edges of the marker as well as the pattern inside it, which is registered in the app. The *arGetTransMat* function identifies the position and pose of a marker, which is represented by a matrix, as shown in Equation (4.5).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = p \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} \tag{4.5}$$

The $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$ coordinates represent the transform matrix of the camera relative to the marker frame coordinates. The transform matrix **p** is multiply by the vector of the marker coordinates relative to the camera coordinates $(\mathbf{x_m}, \mathbf{y_m}, \mathbf{z_m})$.

The marker's pose is represented by three rotation angles called pitch Θ, yaw ϕ and roll ω, which denote rotations around the *x,y,z* axes, respectively. The rotation angles are calculated from the components of the transform matrix in equation (4.5) as follows:

$$\Theta = tan^{-1}\left(\frac{p_{21}}{p_{11}}\right) \tag{4.6}$$

$$\phi = tan^{-1}\left(\frac{p_{32}}{p_{33}}\right) \tag{4.7}$$

$$\omega = sin^{-1}(p_{31}) \tag{4.8}$$

4.4 Augmented Reality Devices

The main devices for augmented reality are displays, input devices, tracking, and computers.

4.4.1 Displays

There are three major types of displays used in Augmented Reality: **head mounted displays (HMD)**, **handheld displays** and **spatial displays**.

HMD is a display device worn on the head that places both images of the real and virtual environment over the user's view of the world. HMD can either be video-see-through or optical-see-through and can have a monocular or binocular display optic. Video-see-through systems are more demanding than optical-see-through systems as they require the user to wear two cameras on his head and require the processing of both cameras to provide both the "real part" of the augmented scene and the virtual objects with unmatched resolution, while the optical-see-through employs a half-silver mirror technology to allow views of physical world to pass through the lens and graphically overlay information to be reflected in the user's eyes.

Handheld displays employ small computing devices with a display that user can hold in their hands, such as smart-phones, PDSs and Tablet PCs. They use video-see-through techniques to overlay graphics onto the real environment and employ sensors, such as digital compasses and GPS units for their six degrees of freedom tracking sensors, fiducial marker systems, such as ARToolKit, and/or computer vision methods, such as SLAM.

Spatial Augmented Reality (SAR) make use of video-projectors, optical elements, holograms, radio frequency tags, and other tracking technologies to display graphical information directly onto physical objects without requiring the user to wear or carry the display. Spatial displays separate most of the technology from the user and integrate it into the environment. This permits SAR to naturally scale up to groups of users, thus allowing collaboration between users, increasing the interest for such augmented reality systems in universities, labs, museums, and in the art community. There exist three different approaches to SAR which mainly differ in the way they augment the environment: video-see-through, optical-see-through and direct augmentation. In SAR, video-see-through displays are screen based; they are a common technique used if the system does not have to be mobile as they are cost efficient since only off. The shelf hardware components and standard PC equipment are required. Spatial optical-see-through displays generate images that are aligned within the physical environment. Spatial optical combiners, such as planar or curved mirror beam splitters, transparent screens, or optical holograms are essential components of such displays. However, much like screen-based video see-through does not support mobile applications due to spatially aligned optics and display technology. Finally, projector-based spatial displays apply front-projection to seamlessly project images directly onto physical object surfaces.

4.5 Input Devices

There are many types of input devices for AR systems. Reitmayr et al.'s mobile augmented system [55] used gloves as input device. ReachMedia [20] use a wireless wristband as input device. Smart-phones can be used as pointing device; for example, Google Sky Map on Android phone requires the user to point his/her phone in the direction of the stars or planets the user wishes to know the name of. The input devices chosen depend greatly upon the type of application the system is being developed for and/or the display chosen. For instance, if an application requires the user to be hands free, the input device chosen will be one the enables the user to use his/her hands for the application without requiring extra unnatural gestures or to be held by the user, examples of such input devices include gaze interaction or the wireless wristband. Similarly, if a system makes use of a handheld display, the developers can use a touch screen input device.

4.5.1 Tracking

Tracking devices consist of digital cameras and/or other optical sensors, GPS, accelerometers, solid state compasses, wireless sensors, etc. Each of these technologies has different level of accuracy and depends greatly on the type of system being developed. General tracking technology for augmented reality: mechanical, magnetic sensing, GPS, ultrasonic, inertia, and optics.

4.5.2 Computers

AR systems require powerful CPU and considerable amount of RAM to process camera images. So fat, mobile computing systems employ a laptop in a backpack configuration, but with the rise of smart-phones technology and iPad, we can hope to see this backpack configuration replaced by a lighter and more sophisticated looking system. Stationary systems can use a traditional workstation with a powerful graphics card.

4.6 Augmented Reality Interfaces

In Augmented Reality, tangible interfaces use real-world objects and tools in order to have a direct interaction with the real environment. Kato et al. [35] developed the VOMAR, which enables a person to select and rearrange the furniture in an AR living room design application by using a real, physical paddle. Paddle motions are mapped to intuitive gesture based commands, such as "cooping up" an object to select it for movement or hitting an item to make it disappear in order to provide the user with an intuitive experience.

Another example is TaPuMa [44], which is a table-top tangible interface that use physical objects to interact with digital projected maps. In this tangible interface, the user (using real-life objects) carries with him as queries to find locations or information on the map. The advantage of this application is that the use of objects as keywords eliminates the language barrier of conventional graphical interfaces. On the other hand, keywords using objects can also be ambiguous because it can have different meaning for each person. White et al. [72] gave a solution for TaPuMa problem, which was to provide virtual visual hints on the real object showing how it should be moved.

Another example of tangible AR interactions include the use of gloves or wristband.

4.7 Collaborative Augmented Reality Interfaces

Collaborative AR interfaces include the use of multiple displays to support remote and co-located activities. Co-located sharing uses 3D interfaces to improve physical collaborative workspace. In remote sharing, AR integrates multiple devices with multiple locations to enhance teleconferences.

Studierstube [59] is an example of co-located collaboration. The designers of Studierstube had in mind a user interface that "uses collaborative augmented reality to bridge multiple user interface dimensions: Multiple users and contexts as well as applications, 3D-Windows, hosts, display platforms, and operating systems."

Remote sharing can be integrated with medical applications for performing diagnostics, surgery, or even maintenance routine in order for enhancing teleconference.

4.7.1 Hybrid Augmented Reality Interfaces

Hybrid interfaces combine different interfaces that are used to interact through a wide range of interaction devices. They provide a flexible platform, which is helpful when the type of interaction display or devices will be used are unknown. Sandor et al. (sandor2005immersive) developed a hybrid user interface using head-tracked, see through, head-worn display to overlay augmented reality and provide both visual and auditory feedbacks. This AR system is then implemented to support end users in assigning physical interaction devices to operations as well as virtual objects in which to perform those procedures, and in reconfiguring the mappings between devices, objects and operations as the user interacts with the system.

4.7.2 Multimodal Augmented Reality Interfaces

Multimodal interfaces combine real objects input with naturally occurring forms of language and behaviors such as speech, touch, natural hand gestures, or gaze. These types of interfaces are more recently emerging.

Wear ur World (WUW) (mistry2009wuw) is the MIT's sixth sense wearable gestural interface, which brings the user with information projected onto surfaces, walls, and physical objects through natural hand gestures, arms movement, and/or interaction with the object itself.

Lee et al. (lee2010design) developed a multimodal interface that makes use of gaze and blink to interact with objects. This type of interaction is now being largely developed and is sure to be one of the preferred type of interaction for future augmented reality application as they offer a relatively robust, efficient, expressive and highly mobile form of human-computer interaction. They have the capability to support users' ability to flexibility combine modalities or to switch form one input model to another depending on the task or setting. In addition, multimodal interfaces offer the freedom to choose which mode of interaction the user prefers to use depending on the context; i.e. public place, museum, library, etc. This freedom to choose the mode of interaction is crucial to wider acceptance of pervasive system in public places.

4.7.3 Augmented Reality Systems

Augmented reality systems can be divided into five categories: fixed indoor systems, fixed outdoor systems, mobile indoor systems, and mobile indoor and outdoor systems. Mobile system allows the user for movement that is not constrained to one room and allows the user to move through the use of a wireless system. Fixed system cannot be moved around and the user must use these systems wherever they are set up without having the flexibility to move unless they are relocating the whole system setup. The selection of the type of system to be built must be made as it will help them in deciding which type of tracking system, display choice and possibly interface they should use. For instance, fixed systems will not make use of GPS tracking, while outdoor mobile system will.

4.7.4 ARToolKit

Artoolkit is a set of libraries for augmented reality developing. It was developed by Hirokazu Kato, the team of HIT Lab of the University of Washington and the team of HIT Lab NZ of the University of Canterbury (New Zealand).

ARToolKit gets the camera position relative to 6 DoF using computer vision methods for marker's tracking in real time. The characteristics are:

- **Camera tracking**: it is used a basic version for camera tracking.
- **Black squares markers**: it is used tracking methods of flat black squares with a pattern, which must not be symmetrical.
- **Fast and cross-platform**: it works with many operative systems such as Linux, Mac, Windows, IRIX, SGI, and so on, and it works with mobile devices and smartphones such as Andriod, iPhone, PDAs, and so on.
- **Active community**: the community helps and resolves any doubt about **ARToolKit**.
- **Open source**: the applications made using ARToolKit can be used, modified and distributed (with the license GPL v2).

4.7.5 ARToolKit algorithm to superpose an object using a marker

Algorithm 1 shows how ARToolKit works. In this algorithm, it is created an *OpenGL* window with Glut library, the initiation function is called, then it is created a thread for video and then main-loop function is called.

Algorithm 1 Main algorithm of ARToolKit

```
// Create an OpenGL window with Glut library
$glutInit(\&argc, argv)$
//Call the initiation function
$init()$
//Create a thread for video
$arVideoCapStart()$
//Associate callbacks
$argMainLoop(NULL, NULL, mainLoop)$
//End of the program
$return(0)$
```

Algorithm 2 Initiation algorithm of ARToolKit

```
//Intrinsic parameters of the camera
ARParam wparam, cparam
//Size of the video camera (pixels)
//int xsize, ysize
//Open video device
if $arVideoOpen(””) < 0$ then exit(0)
if $arVideoInqSize(\&xsize, \&ysize) < 0$ then exit(0)
//Put the intrinsic parameters of the camera
if $arParamLoad(”data/camera_{p}aram.dat”, 1, \&wparam) < 0$ then
    print_error(Error putting the camera parameters)
arParamChangeSize(&wparam,xsize,ysize,&cparam)
//Initialize the camera
arInitCparam(&cparam)
//Add the marker
if patt_id=arLoadPatt("data/simple.patt")<0 then
    print_error("Error adding the marker")
//Open the window
argInit(&cparam,1.0,0,0,0,0)
// The mainLoop function it is called automatically (a callback it is registered).
This algorithm grabs a image or frame, detects a marker and calls a function to
draw a 3D object.
```

The *Initialization* algorithm (Algorithm 2) consists on *read* the camera parameters and the marker pattern to be used.

Algorithm 3 Main Loop algorithm of ARToolKit

```
//Capture an image from the video stream
if dataPtr = (ARUint8 *)arVideoGetImage()==NULL then
    $arUtilSleep(2)$
    return
//Draw the image grabbed from the video stream
$argDrawMode2D()$
$argDispImage(dataPtr, 0, 0)$
//Detect the marker in the grabbed image (if it returns -1,
//it means error)
if $arDetectMarker(dataPtr, 100, \&marker_{i}nfo, \&marker_{n}um)$<0 then
    cleanup()
    exit(0)
//Get another image from video stream
$arVideoCapNext()$
//Get the marker's position and orientation with the best reliability after the
marker-detection algorithm has been run several times.
for $j = 0, k = -1; j < marker\_num; j++$ do
    if $patt\_id == marker\_info[j].id$ then
        if $(k == -1)$ then $k = j$
        else
            if $marker\_info[k].cf < marker\_info[j].cf$ then $k = j$
//If the marker has been detected, it is obtained the transformation between the
marker and the camera
if $k! = -1$ then
    $arGetTransMat(\&marker\_info[k], p\_center, p\_width, patt\_trans)$
    $draw()$
//Changing the buffer where it is draw the model
$argSwapBuffers()$
```

The *main loop* algorithm (Algorithm 3) is the main phase and it consists of: grabbing an image of the video stream, detect a marker on the image, detecting the marker's position and orientation, computing the relative position between the marker and the camera, and drawing a 3D object over the marker's pattern (if it is detected any).

Algorithm 4 Draw algorithm

```
//Matrix 4x4 used by OpenGL
double gl_para[16]
GLfloat mat_ambient[]=0.0,0.0,1.0,1.0
GLfloat light_position[]=100.0,-200.0,200.0,0.0
//Change the context to 3D
argDrawMode3D()
//Change the camera view to 3D
argDraw3dCamera(0,0)
//Clean the buffer
glClear(GL_DEPTH_BUFFER_BIT)
glEnable(GL_DEPTH_TEST)
glDepthFunc(GL_LEQUAL)
//Convert the matrix of the marker to be used by OpenGL
argConvGlparam(patt_trans,gl_para)
glMatrixMode(GL_MODELVIEW)
glLoadMatrixd(gl_para)
//Draw the 3D object
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glLightfv(GL_LIGHT0,GL_POSITION,light_position)
glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient)
glTranslatef(0.0,0.0,60.0)
glRotatef(90.0,1.0,0.0,0.0)
glutSolidTeapot(80.0)
glDiable(GL_DEPTH_TEST)
```

Algorithm 4 shows how the draw algorithm of ARToolKit works. Firstly, a matrix 4x4 is created. Secondly, the context is changed to 3D. Thirdly, the camera view is changed to 3D. Fourthly, the buffer is cleaned. Fifthly, the marker's pose matrix is converted to be used by *OpenGL*. Sixthly, a 3D object is drawn.

The proposed methodology is explained in the next section.

Chapter 5

Methodology

Marker′s tracking research for augmented reality is a challenge that needs a practical solution in real-time. In this chapter, it is explained the criteria of decision for the proposed solution. Furthermore, it is explained the proposed solution, the light range setting and the machine learning system.

5.1 Criteria of decision of evaluation of different algorithms to reduce marker's position error

In the past section, it was described the definition of the problem of how to have an accurate marker′s position estimation under uncontrolled environment. In this section, it is described the criteria of evaluation of different algorithms to reduce marker′s position error.

The heuristics and evolutionary optimization approaches uses an intuitive sense, which generally gives fast results, but these results are not necessarily the best solution. In the case of the statistical approach, it can be helpful to understand the outcome of the problem, the effect of some variables, the relationship between two variables, the differences among groups of observations are the same or different, and so on. Statistics are used to substantiate the findings and to help to establish objectively when the results are significant. In Table 5.1 are defined and evaluated different algorithm of artificial intelligence.

Therefore, the statistical approach is the only solution that can generate the best solution. It is proposed to use an algorithm that uses a statistical as the machine learning. Machine learning uses regression tool in optimization problems. These type of problems have variables that affect a response variable, and the regression tool is used to understand the relationship between the variables and the response variable.

Table 5.1: Evaluation of different algorithms

Algorithm	**Type of algorithm**	**Brief description**	**Evaluation of the algorithm for marker's position error reduction**
Unsupervised learning [42]	Machine learning	It is used to find patterns of input	In wasn't needed to find a pattern in a stream in this study case. The labels of the data was already known and the classification of the marker's pattern wasn't in the scope of the problem.
Supervised learning [42]	Machine learning	Includes both classification and numerical regression. Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs outputs should change as the inputs change.	The classification problem wasn't in the scope of the problem. Regression is a good option to reduce the marker's error position.
Reinforcement learning [42]	Machine learning	The agent is rewarded for good responses and punished for bad ones. The agent uses this sequence of rewards and punishments to form a strategy for operating in its problem space.	It is an heuristic option and it doesn't guarantee an optical solution.
Heuristics [49]	Search and optimization	Heuristics supply with a "best guess" for the path on which the solution lies. Heuristics limit the search for solutions into a smaller sample size.	The heuristics is an option for search a solution in optimization in a fast way.
Hill climbing [49]	Search and optimization	The algorithm begin the search at a random point on the landscape, and then, by jumps or steps, we keep moving our guess uphill, until we reach the top.	The heuristics is an option for search a solution in optimization in a fast way.
Evolutionary optimization [13]	Evolutionary computation	It uses a form of optimization search. For example, it begins with a population of organisms(the guesses) and then allow them to mutate and recombine, selecting only the fittest to survive each generation (refining the guesses). Forms of evolution computation include swarm intelligence algorithms (such as ant colony or particle swarm optimization) and evolutionary algorithms (such as genetic algorithms, gene expression programming, and genetic programming).	Evolutionary optimization searches in limited population. It generates fast results but the search is limited.

5.2 Proposed solution

In the past section, it was described the definition of the problem of how to have an accurate marker's position estimation under uncontrolled environment. In this section, it is described the proposed solution for this problem, which consists on a proposed methodology for estimating marker's position based on machine learning techniques using statistical uncertainty of marker's position under different light conditions [66].

Based on a pre-statistical analysis (Anderson-Darling test) of the samples taken, the data had a normal distribution. Using this information, it is proposed to use a regression technique, which is used for fitting the data, which means that the noise could be reduced. In addition, the pre-statistical analysis shows that the marker's position has different error depending on the light.

Based on these facts, it is proposed a methodology that consists on machine learning using multiple regression for each type of regression range. This proposed methodology consists of the following steps:

1. Take samples (initial preparation offline)
2. Apply machine learning to the samples taken (training phase)
3. Prediction of the marker's position estimation
4. Optimization process of the prediction functions (which consists on the repetition of stage 1 and stage 3)

The proposed methodology needs a preparation offline before the usage of the predictor function for fitting the data in order to have better marker's position estimations. In addition, this methodology like machine learning gets accurate results when the training examples increase. Furthermore, the predictor function generated must be used under the same light conditions that have been used in offline preparation.

5.2.1 Initial preparation offline

Before starting using the proposed methodology, an initial preparation offline is made, which consists on the preparation of the experimental setup for taking samples, which are the input to the machine learning process for fitting marker's position data.

The experimental setup (as shown in Figure 6.1) for taking samples of the marker's position in different positions under different light conditions consist of the following:

1. A luxmeter tool (to take light measurements)
2. An incandescent light bulb
3. A camera (Logitech C920) (in front the marker or fiducial)

4. A marker (with QR-code pattern)
5. A computer (2.8 GHz Intel Core i7 MacBook Pro) (to take marker′s positions)
6. A device for getting the Ground truth (in this experimental setup was used a CNC machine tool guide)
7. Stages (to fix these tools).

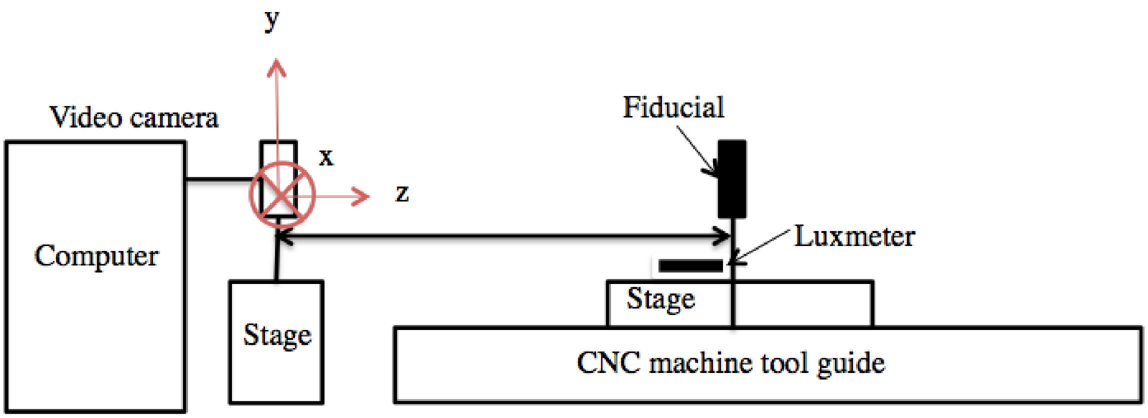


Figure 5.1: Experimental setup

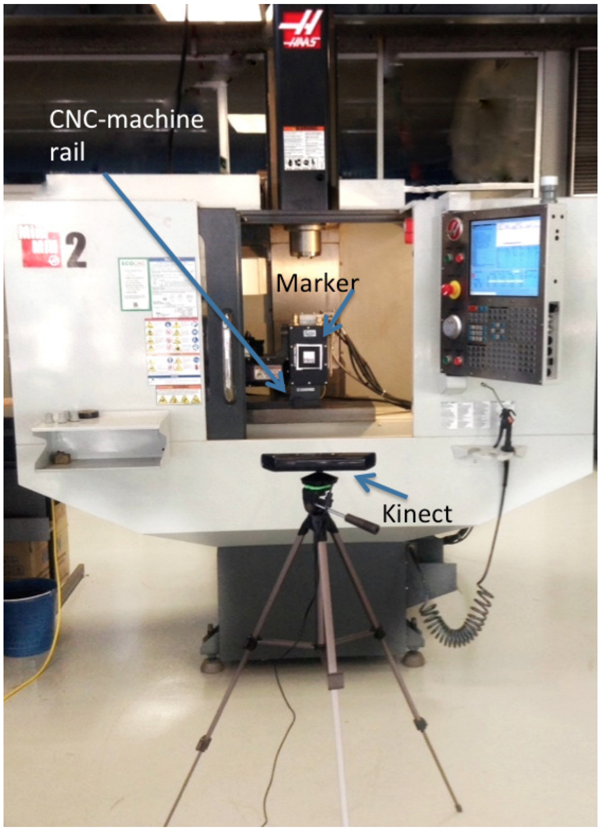


Figure 5.2: Experimental setup: CNC machine tool guide

The purpose of this step is to have a better understanding of the effect of the light under the marker's position estimation. For accomplish this purpose, the samples (of estimated marker's positions under different light conditions and positions) are taken for quantifying the statistical uncertainty. The process of taking samples consists on taking and saving the data in a text file (using ARToolKit application) when the fiducial along the CNC machine tool guide is moving and the light measurement is taken with a luxmeter tool. Every time the marker reach the distance wanted (which is moved with a CNC machine tool guide), the light measurement is taken with a luxmeter near the marker. The ground truth of the marker's position is taken with a CNC machine tool guide.

As it was explained before, the pre-statistical analysis of the data shows that the marker's position estimation has different error when the light changes. In order to classify the data and have accurate data fitting of the marker's position it is proposed a method to classify light by range.

5.3 Light range setting

The accuracy of the marker's position estimation depends on light conditions. In order to minimize more the position estimation error, it is proposed to divide the data into ranges of light conditions. Just because the marker's position estimation is

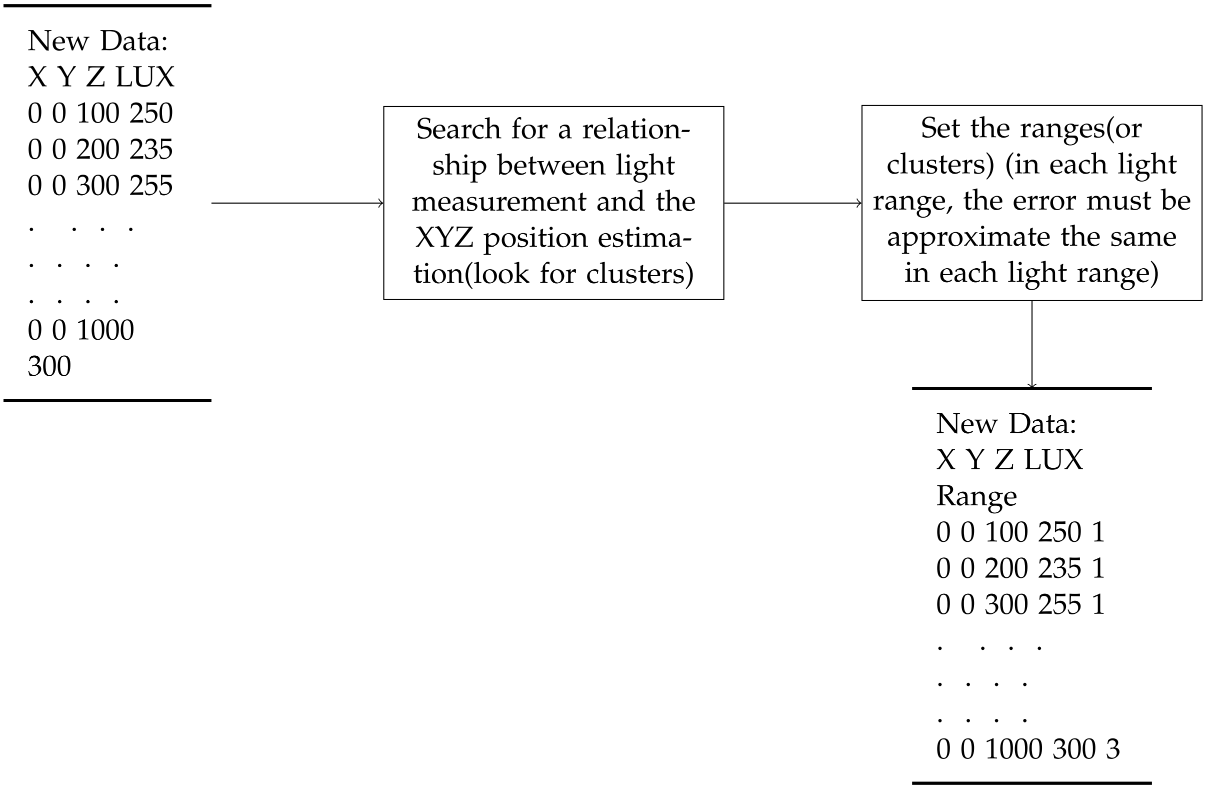

New Data:
X Y Z LUX
0 0 100 250
0 0 200 235
0 0 300 255
. . . .
. . . .
. . . .
0 0 1000 300
Search for a relationship between light measurement and the XYZ position estimation(look for clusters)
Set the ranges(or clusters) (in each light range, the error must be approximate the same in each light range)
New Data:
X Y Z LUX Range
0 0 100 250 1
0 0 200 235 1
0 0 300 255 1
. . . .
. . . .
. . . .
0 0 1000 300 3

Figure 5.3: Process for setting light range

so sensitive to the light change, it is proposed treat the error by light range separately in order to minimize the error.

Using the data taken from the samples (marker's position estimation using ARToolKit under different light conditions and positions), it is used to choose the light ranges in order to classify the data. As shown in Figure 5.3, the process for setting *light range* is the following: first, a relationship between the light measurement and the XYZ position estimation must be searched. Second, an analysis of the relationship of the position error and the light measurement must be done in order to set the light ranges. In each light range chosen, the approximate error must be the same. Third, the light range information must be added to the data input for the machine learning system.

The process of ARToolKit for estimating the marker's position is explained in Algorithm 5, which shows the pseudo code of the algorithm for grabbing an image, detect a marker and estimate its XYZ positions. This algorithm starts creating a file for saving the data and detecting the camera. Then, if there isn't any problem, a loop starts for grabbing an image, detecting the marker in it, and saving the data in the file. The data is saved in a file in the form of lists of the XYZ position. Then, the light measurements are added into the file. At the end, the data is divided by light range into three different files (one file for each light range).

Using the light range and the samples (taken in the initial preparation offline),

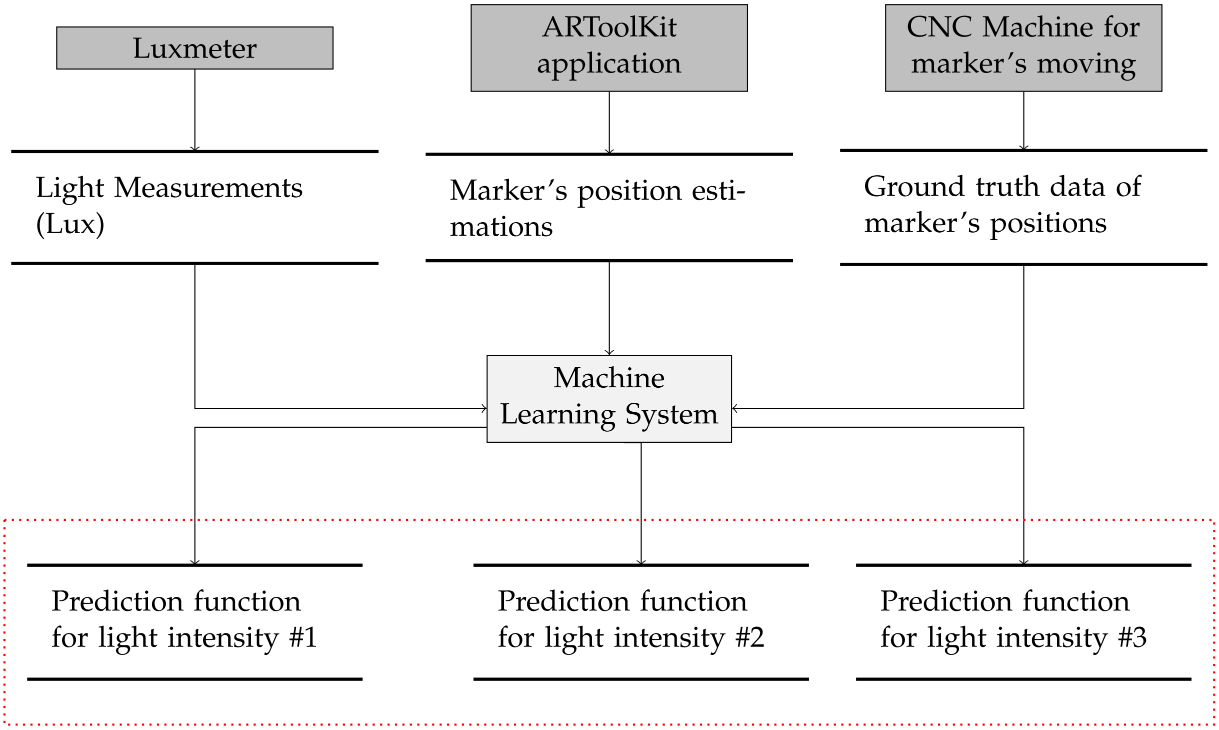

Luxmeter
ARToolKit application
CNC Machine for marker's moving
Light Measurements (Lux)
Marker's position estimations
Ground truth data of marker's positions
Machine Learning System
Prediction function for light intensity #1
Prediction function for light intensity #2
Prediction function for light intensity #3

Figure 5.4: Machine learning system for AR marker position estimation

the sample data is classified by light range, which is the input for the machine learning process in order to generate models for fitting the marker's position by light range.

5.4 Machine learning

As it was explained in the past section, it is used a machine learning based method for fitting the data of marker's position estimation by light range in order to minimize the error. As it was explained before, the data has a normal distribution, so it was applied multiple linear regression in order to generate models.

As shown in Figure 5.4, the proposed method for marker's position estimation under controlled environment works as following: first, the system receives as input the sample data under different position and light conditions (as it was explained in the past section). Then, the light measurements are taken with a luxmeter tool. Initial marker position estimations (or sample data) are taken using ARToolKit application (as it was shown in Algorithm 5). Second, the machine learning system (Algorithm 6) builds a model to fit the marker position data using the initial position estimations and light condition data. The training is finished when the system stops receiving training examples (or samples). Third, the system finds the best predictor function for marker position estimation.

In order to get the maximum reduction of error of the marker's position estimation, the data is classified by light range. In the statistical analysis in each light range

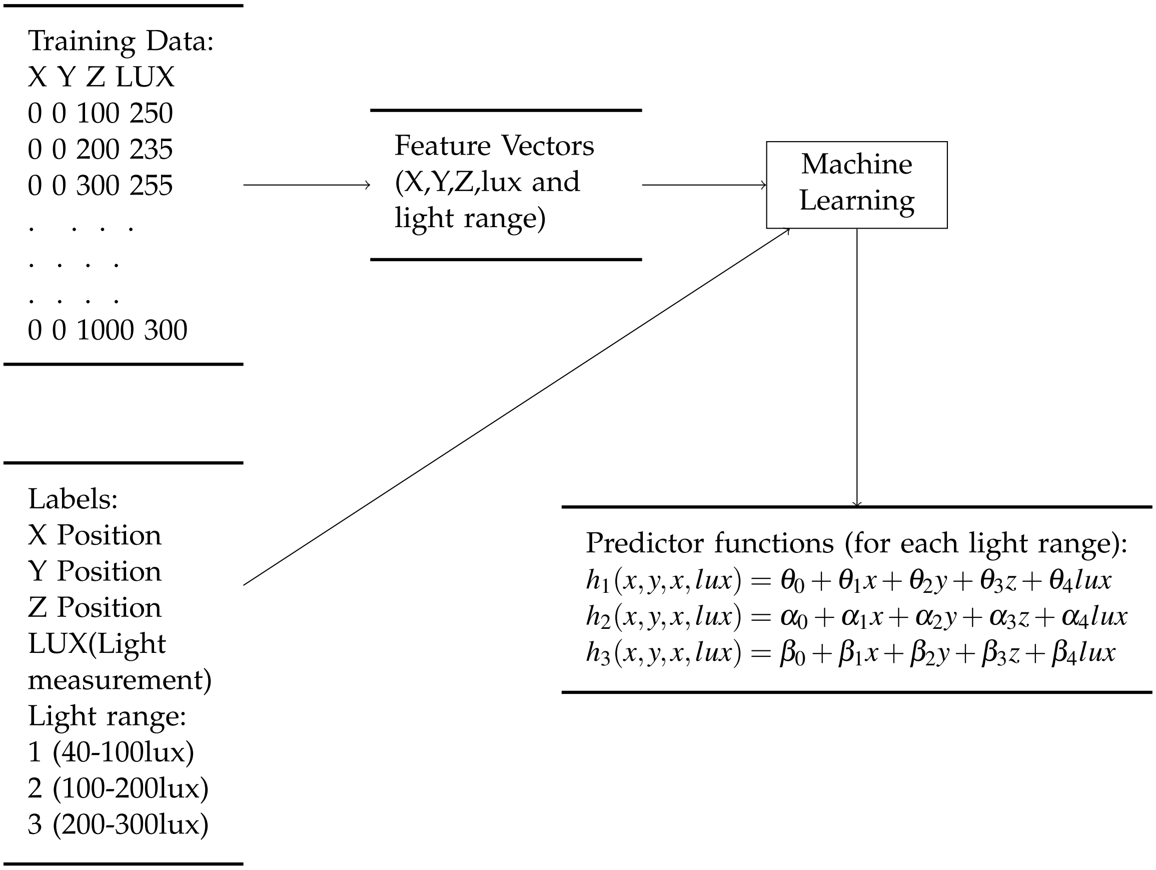

Training Data:
X Y Z LUX
0 0 100 250
0 0 200 235
0 0 300 255
. . . .
. . . .
. . . .
0 0 1000 300
Feature Vectors (X,Y,Z,lux and light range)
Machine Learning
Labels:
X Position
Y Position
Z Position
LUX(Light measurement)
Light range:
1 (40-100lux)
2 (100-200lux)
3 (200-300lux)
Predictor functions (for each light range):
$h_1(x,y,x,lux) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 z + \theta_4 lux$
$h_2(x,y,x,lux) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z + \alpha_4 lux$
$h_3(x,y,x,lux) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z + \beta_4 lux$

Figure 5.5: Machine learning process

the data has different mean error. The machine learning process is made using this light range classification.

As shown in Figure 5.5, the data (i.e. XYZ marker's position and light measurement) is divided into n categories of light range and saved into three different files. This data is the input of the machine learning system, which uses a multiple linear regression algorithm. As shown in the Algorithm 6, the system grabs the information and save it in a matrix in order to apply the *gaussian elimination* to it and obtain a model. Then, using the *loss function* (Equation 3.2), the model is optimized with the best coefficients for it. This optimization process with loss function consists on changing the coefficients dynamically in a random way and find the best solution. The system learns from previous computations and new training examples in order to produce new models for AR marker position estimation. As shown in Figure 5.8, the training process ends when the system stops receiving training data.

The distribution of the data is a very important step before made any predictor function. A statistical analysis is needed in order to know the distribution of a population. Knowing the distribution of the data, it is possible to generate more accurate models (predictor functions).

As shown in Figure 5.6, the first step to fit data, it's finding the data distribution. First, it's done a statistical analysis, in which it's tested the data for the typical

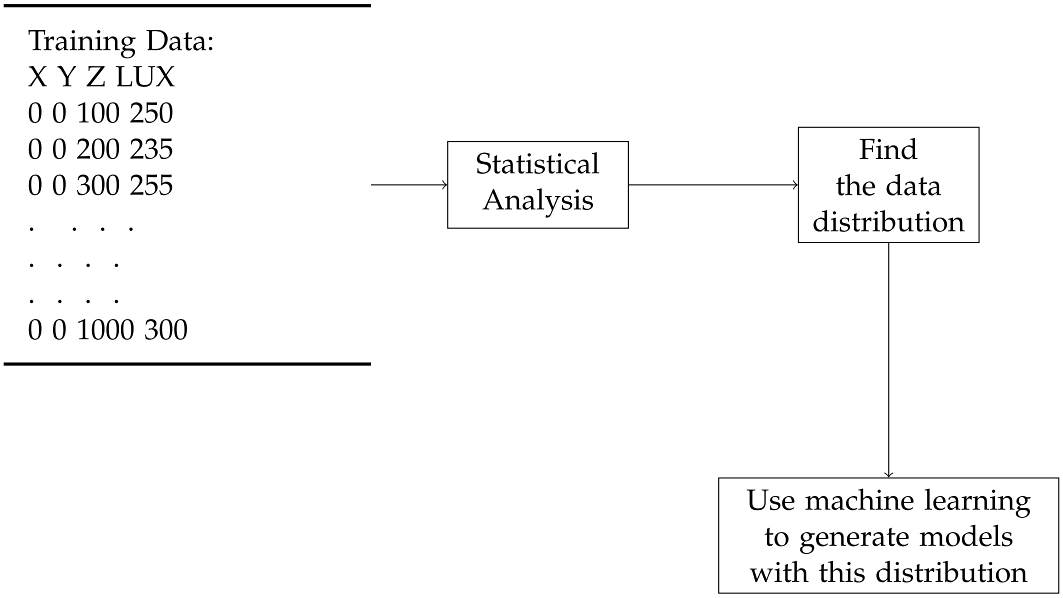


Figure 5.6: Process for finding the data distribution

distributions like normal, uniform, logistic, and so on. Second, the distribution that fits better the data must be chosen. Third, the distribution chosen must be used in the machine learning system in order to find the best model that fit the data.

Using this machine learning based method, the predictor functions are generated and used to make estimations for marker's position with more accuracy. In the next section, it is going to be explained this process.

5.4.1 Prediction using machine learning

In this section, it is explained the use of the predictor functions in order to make accurate estimations of marker's position. A better accuracy in marker's position estimation is needed in Augmented reality in order to get rid errors like *jittering* (a virtual object appears and disappears) or to get more accuracy for detecting objects with a marker over them.

As shown in Figure 5.5, in order to make a prediction, new data (XYZ marker's positions and light measurement) must be given to the system. Using the models generated using machine learning and the data to estimate the marker's position with more accuracy.

The accuracy obtained with these predictor functions is improved when the machine learning based system gets more training examples, this process is called *optimization*. This process is explained in the next section.

5.4.2 Optimization of the models

In industrial applications as detecting the position of objects with markers over them, it is very important to have accuracy in order to have a full or semi automatic system.

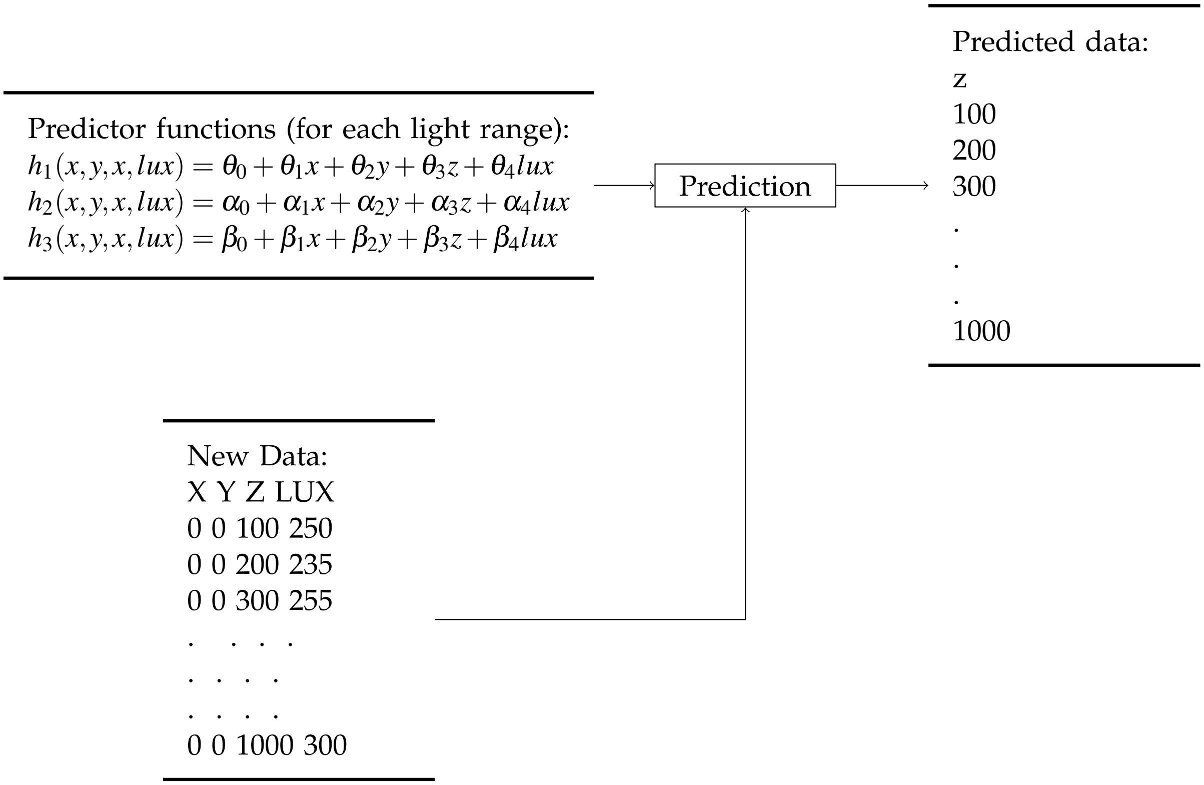

Predictor functions (for each light range):
$h_1(x,y,x,lux) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 z + \theta_4 lux$
$h_2(x,y,x,lux) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z + \alpha_4 lux$
$h_3(x,y,x,lux) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z + \beta_4 lux$
Prediction
Predicted data:
z
100
200
300
.
.
.
1000
New Data:
X Y Z LUX
0 0 100 250
0 0 200 235
0 0 300 255
. . . .
. . . .
. . . .
0 0 1000 300

Figure 5.7: Prediction process

The system learns from previous training models and new data in order to get new models for AR marker position estimation. As shown in Figure 5.8, the input are old models (it also could be the old data with ground truth) in order to generate new models. The training process ends when the system stops receiving training data.

The models become better and more accurate when it is used more training examples. The training examples must have been taken in many conditions possible. For example, take samples in the same positions, but it was changed the light conditions every time the sample is taken. The samples taken must be at the same distances, but it was used a different light condition.

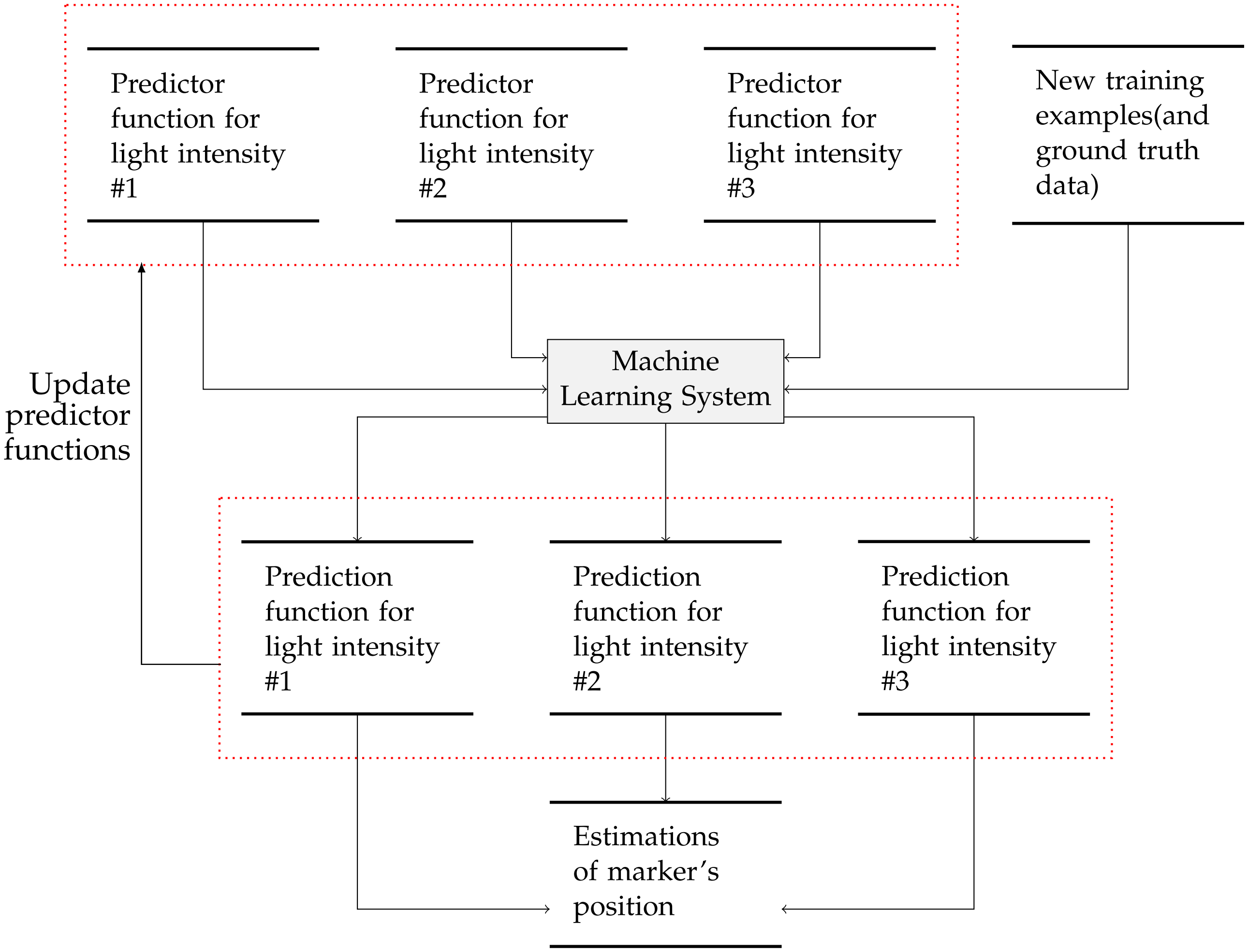

Predictor function for light intensity #1
Predictor function for light intensity #2
Predictor function for light intensity #3
New training examples(and ground truth data)
Machine Learning System
Update predictor functions
Prediction function for light intensity #1
Prediction function for light intensity #2
Prediction function for light intensity #3
Estimations of marker's position

Figure 5.8: Optimization process of predictor function

In the next chapter, it is explained how to set the experimental setup and the conditions of the experiment. It is also explained how the light range selection was made.

Chapter 6

Experimental setup and results

In the previous chapter, the problem and proposed solution of how to make the marker's detection more accurate in an uncontrolled environment was discussed. It was proposed a machine learning based method in order to fit the data generated by the ARToolKit application. It was also proposed to generate a predictor function for each light range. It was also proposed a method for choosing a light range. This light range was chosen by similar average error of marker's position. This approach reduces the error of the marker's position.

In this section, it is described the light range selection based in the samples taken. Indeed, in this section it is described the experimental setup and the experiments carried out. The same experiment (with different parameters) was repeated three times.

6.1 Experimental setup

The proposed method needs an initial preparation offline before using this method properly in order to make any marker position estimations. Firstly, before taking any sample, it is needed to prepare the experimental setup (as shown in Figure 6.1) with the luxmeter tool, the camera (Logitech C920, camera web ACTECK CW-760 standard and kinect Xbox 360), the marker (ARToolKit fiducial), the computers (2.8 GHz Intel Core i7 MacBook Pro for machine learning system and a 2.00GHz Intel(R) Pentium(R) Dual CPU T3200 Toshiba Satellite for getting samples), CNC machine tool guide (for training), and stages (to fix these tools). Secondly, the samples (of estimated marker positions under different light conditions) are taken and saved in a text file when the fiducial along the CNC machine tool guide is moving and the light measurement is taken with a luxmeter tool. It was used CNC machine tool guide in order to get the ground-truth position of the marker. The data is divided into three categories of light range and saved into three different files. Thirdly, the data saved in a text file by the ARToolKit application is the input of the machine learning application written in C++, which made the regression analysis with the data. Algorithm 6 shows the training process using multiple linear regression to generate a model for each range light. The machine learning application returns predictor functions for each light range. These predictor functions are used to estimate the marker position.

In each sample, it is going to take into account the last marker's position estimation in each position where it is taken. Every camera or kinect has an auto-focus, which generates noise to the data at the start of the marker's detection. Therefore, it is going to take into account the sample #400 in each position.

The cameras or kinect used are the following:

1. Logitech C920 with a resolution of 1920 x 1080 RGB @ 30fps
2. Camera web ACTECK CW-760 standard with a resolution 640 x 480 RGB @ 30fps
3. Kinect Xbox 360 with a resolution 1280 x 960 RGB @ 12fps

In this experiment, it is studied the effect of underfitting in each sample. Each sample has 41 marker's position estimations that were taken with 10mm of difference between each position estimation. Then, every sample is reduced in order to establish the underfitting functions. Therefore, in each sample set of 41 position estimations are eliminated 20 position estimations. Then, each underfit sample consists on position estimations that were taken every 20mm through 400mm.

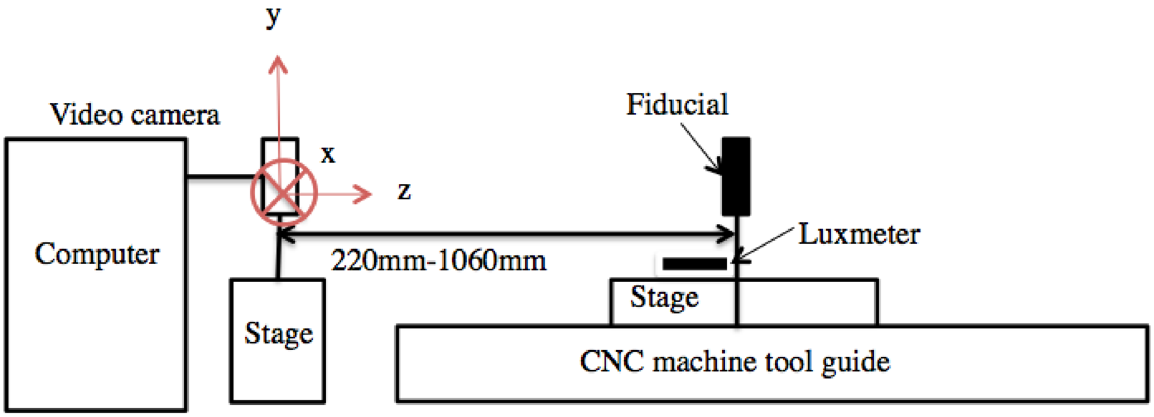


Figure 6.1: Experimental setup

6.2 Light range selection

The light range selection is an important step in order to divide the data into groups. The main idea of this process is to reduce the light error that makes noise. If the data is divided into groups, then the noise is not going to affect all the estimations.

The light range used in this work was chosen through the statistical analysis of the sample data taken. In the analysis of the data, it was found that three groups have the same mean error. In the light range from 40 to 100 lux, the mean marker's position error was of 40.69 mm. In the light range from 100 to 200 lux, it was found that the mean marker error was of 40.13 mm. In the light range from 200 to 310 lux, it was found that the mean marker's position error was of 69.76 mm. Then, these three groups of light measurement are used in our experiment as three range light, which are shown in Table 6.1.

Light Ranges		
Type of range	Lux range	Average error of marker's position
Range #1	40-99	40.69
Range #2	100-199	40.13
Range #3	200-310	69.76

Table 6.1: Light ranges

6.3 Standard distance for each operations

In automated assembly it is used robotics arms. In this study case, where used two types of robots: a pick-and-place robot and a welding robot. The operations in this case are pick and place a piece and weld the joints of the assembly. The ISO 10218-2: 2011:Robots and robotic devices [31] establish the security distances for stopping the robots. It was taken into account the safe distance in order to establish the scope of the experiment.

The formula for the safe stopping distance is in Table 6.2. In many cases the safe stopping distance is taken into account in the design of a work cell. In this study case, it is used a pick-and-place robot with a reach distance of 710mm. It is also used a welding robot with a reach distance of 1437mm. In this case, the reach distance is taken as a minimum distance from the camera and the respective robot.

6.4 Experiments using a Logitech C920 camera

In sample 1, Figure 6.2, the light range used in this sample goes from 41 to 303 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker's position error using ARToolKit approach is 49.8645 mm.

In sample 2, Figure 6.3, the light range used in this sample goes from 124 to 279 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker's position error using ARToolKit approach is 32.7584 mm.

In sample 3, Figure 6.4, the light range used in this sample goes from 116 to 256 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker's position error using ARToolKit approach is 20.5810 mm.

Standard distance for operation			
Type of operation	Description	Distance	Formula
Pick-and-place	Robot CRS F3	710mm	Robot's reach=710mm
Welding	Fanuc ARC Mate 0iBt	1437mm	Robot's reach=1437mm
Safe distance	Safe distance to stop a robot	DS	$DS = 63(in/s) * (TS + TC + TR) + DPF$ Where: DPF= 1.2 m (48 in.) DS= minimum safe distance TS= stopping time of device TC= worst stopping time of control system TR= response time of safeguarding device including interface DPF= maximum travel distance toward a hazard once someone has entered the field [62]

Table 6.2: Safe distances for each operation

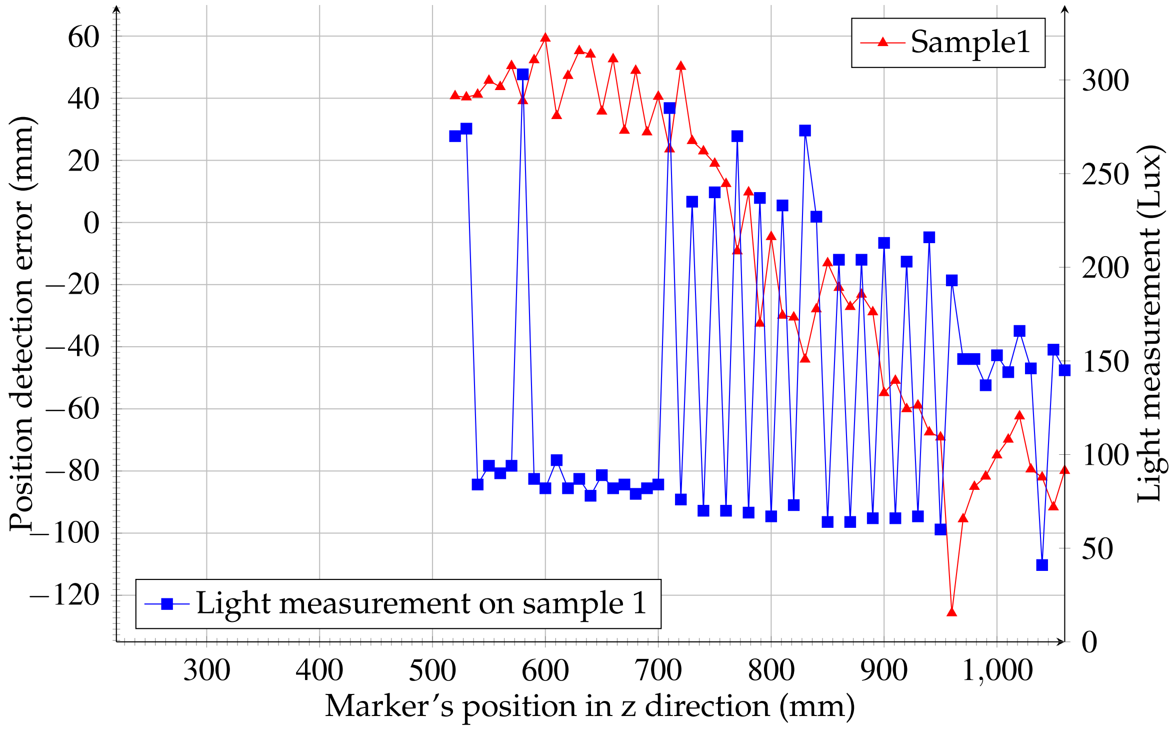


Figure 6.2: Experimental results of position detection error on Sample 1

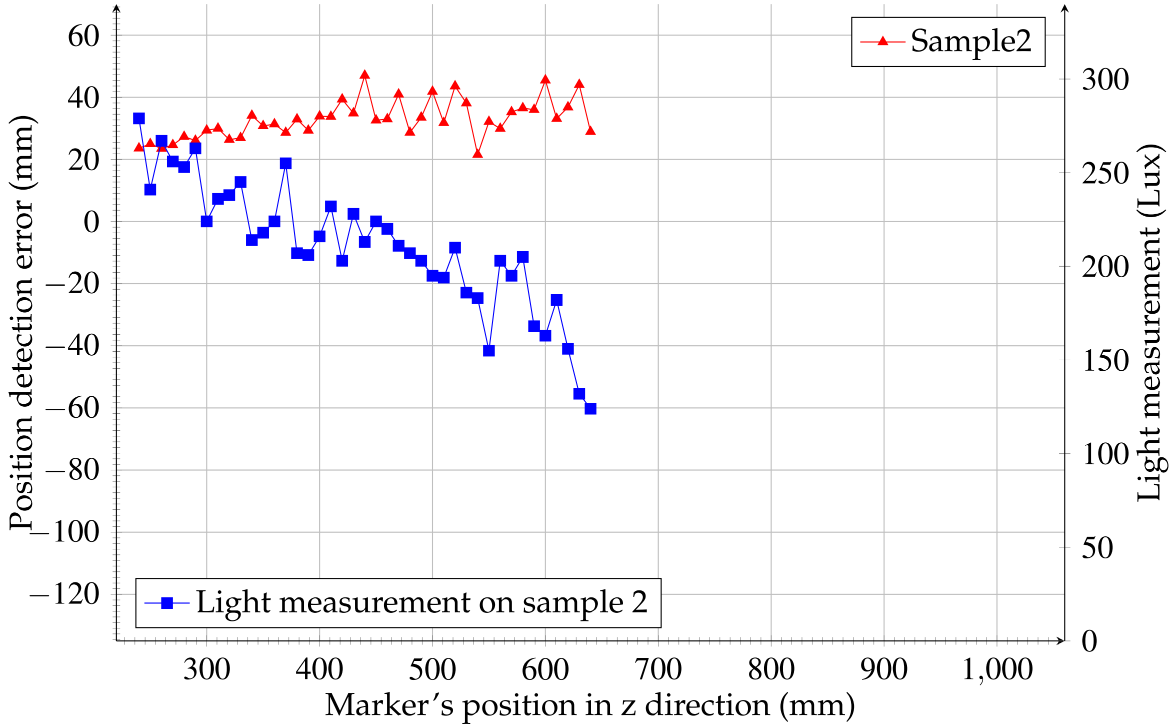


Figure 6.3: Experimental results of position detection error on Sample 2

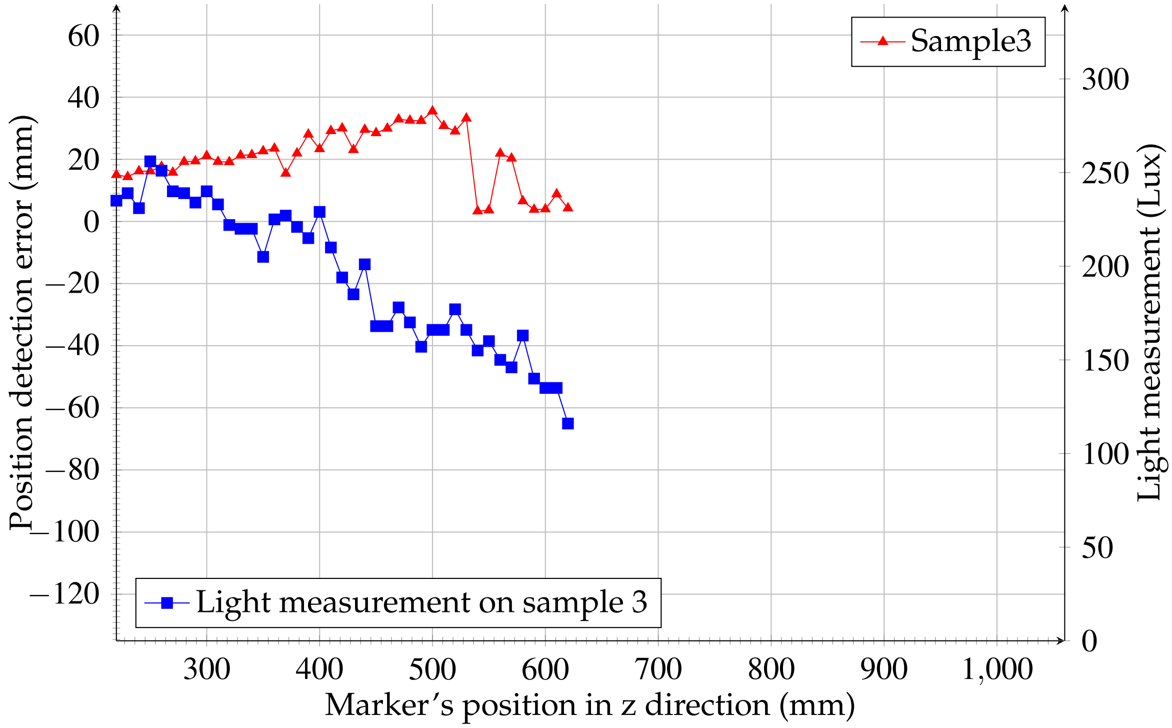


Figure 6.4: Experimental results of position detection error on Sample 3

In these three samples, it was found that the light conditions change the marker position estimation using ARToolKit libraries. The mean of the marker's position error was not constant in these samples because of the light conditions were different in each sample. In our proposed approach, the *light condition variable* is taken into account in order to have better marker position estimations.

In the statistical analysis of the samples, it was found that the data has a normal distribution. It was used the Anderson-Darling normality test on the three samples in order to know if the samples have a normal distribution. It is possible to use multiple linear regression in order to generate models to estimate marker's position because the data came from a normal distribution.

In Table F.1 are shown the fit models generated for the proposed method and the simple linear regression. Also, in Table F.2 are shown the underfit models generated for the proposed method and the simple linear regression.

The Figure 6.5 shows the cumulative improvement of different approaches for the marker's position estimation in z-direction on sample 3. The improvement of the proposed approach and the simple regression approach are compared to ARToolKit. It was compared the cumulative error in each position of the proposed method and the simple linear regression method with the ARToolKit position estimation error. The ARToolKit approach refers to get estimations of marker position using ARToolKit libraries. The simple linear regression analysis approach refers to generate predictor functions of the data without be classified by light range. The proposed approach consists in classify the data by light range and then do a multiple regression analysis to the data. In this experiment, the three light ranges used are from 40 to 100 lux, 100 to 200 lux and 200 to 310 lux.

As shown in Figure 6.5, it was found that using the proposed method and taking into account light conditions, the error of position estimation can be reduced. The mean of the position error using the proposed approach is of 1.258 mm, additionally, the mean of the position error using ARToolKit approach is 20.58 mm. Using the proposed approach, it is obtained better results than using only ARToolKit approach. In the case of underfitting, the mean of the marker's position error using ARToolKit approach is 20.6142mm, and the mean of the position error using the proposed approach is of 7.8325mm.

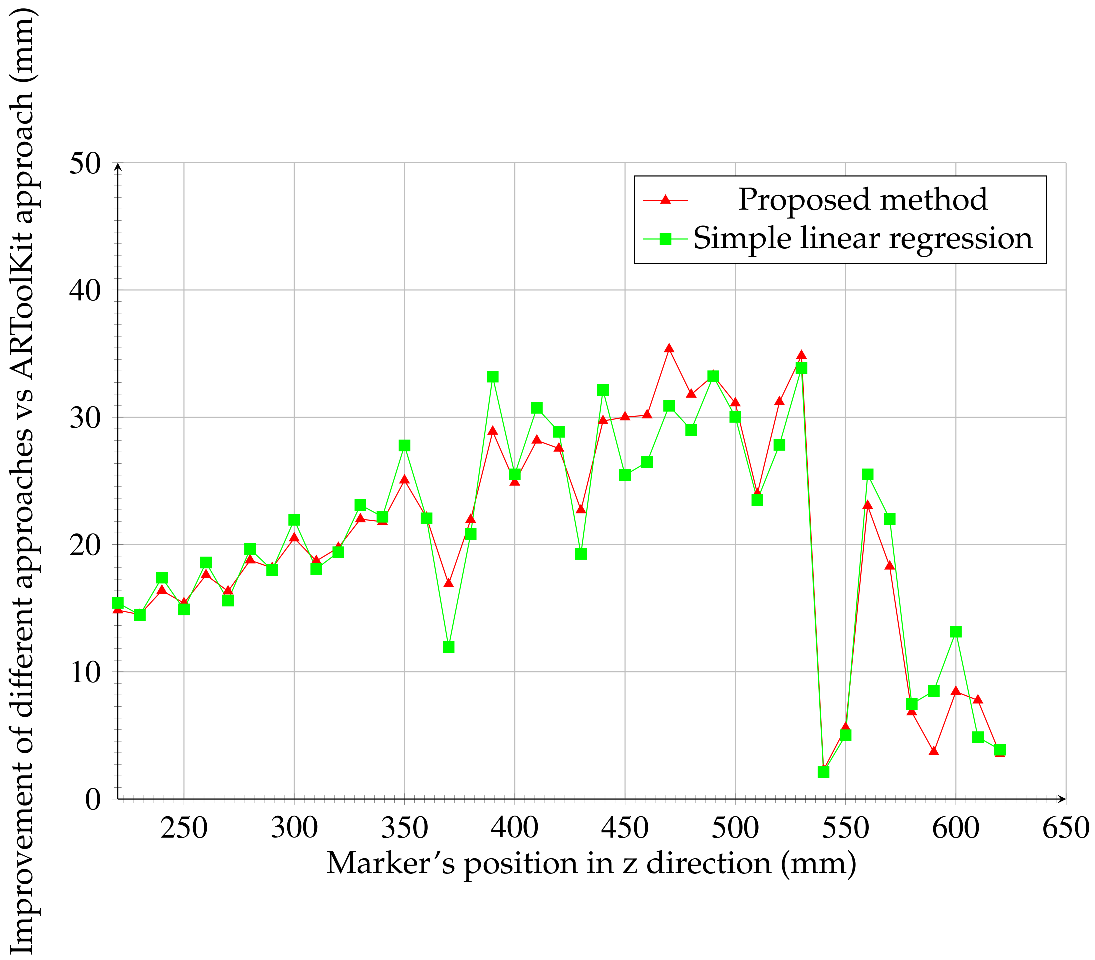


Figure 6.5: Experimental results of improvement of proposed approach and simple regression approach on Sample 3

Furthermore, the proposed approach has better results if the training set is increased. The position error was reduced 87% in this scenario. Furthermore, nearly 99% of the total variability in the response variable (marker position) is accounted for by the estimated marker position by ARToolKit and the light measurement. In this experiment, the regression line models have a $R^2 = 0.99$ that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker position and the light conditions).

In the case of underfitting proposed method, the marker's position error was reduced 62.00% in this scenario. Furthermore, nearly 98.18% of the total variability in the response variable (marker's position) is accounted for by the estimated marker's position by ARToolkit and the light measurement. In this experiment the regression line models have a $R^2 = 0.9641$ and $R^2 = 0.9995$ respectively that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker's position and the light conditions).

In Figure 6.5 is shown the maximum improvement of the proposed method vs the ARToolKit approach is of 35.3535144mm and the minimum improvement is of

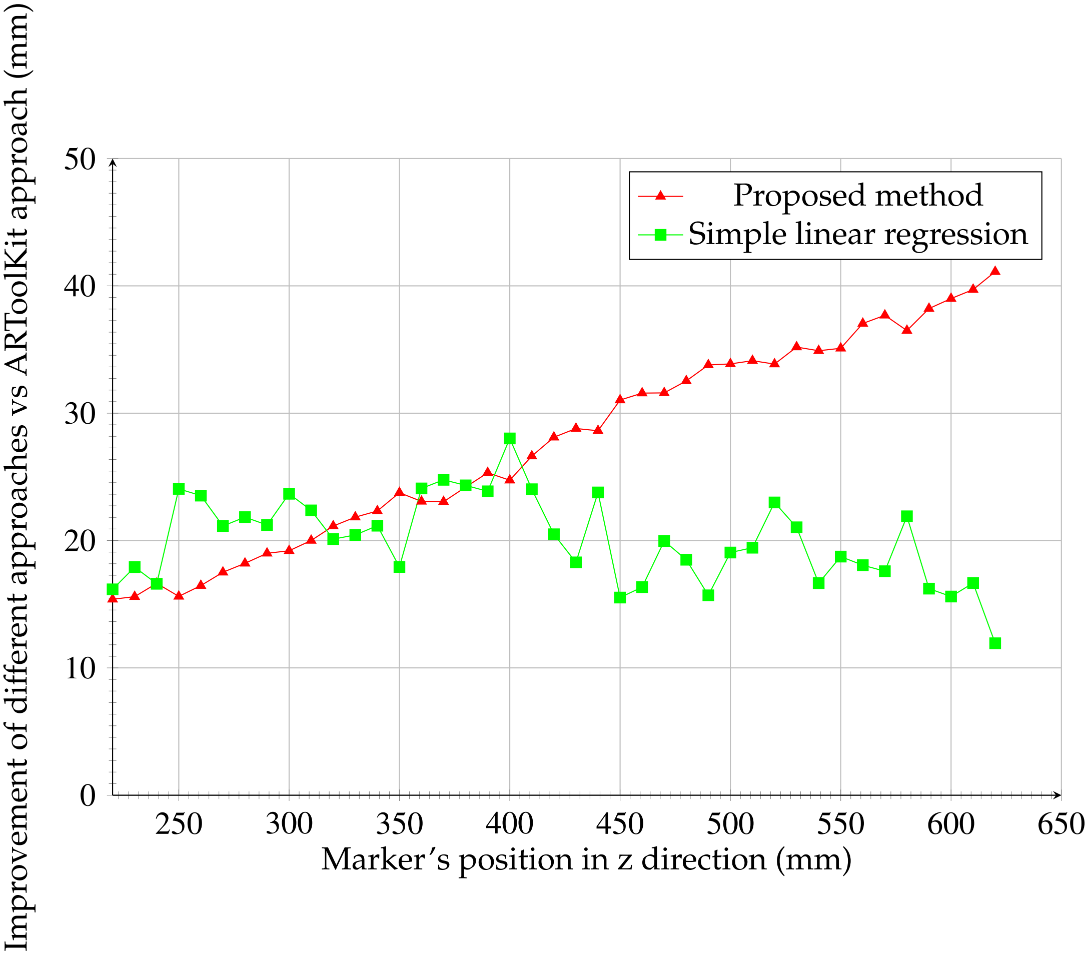


Figure 6.6: Experimental results of improvement of proposed approach and simple regression approach on Sample 3 with underfitting

2.2849mm. In Figure 6.5 is shown the maximum improvement of the simple linear regression vs ARToolKit approach is of 33.8724mm and the minimum improvement is of 2.1167mm. In this case, there is not much improvement between the proposed method and the simple linear regression method in each marker's position estimation.

In Figure 6.6 is shown the maximum improvement (with underfitting) of the proposed method vs the ARToolKit approach is of 41.1117mm and the minimum improvement is of 15.38mm. In Figure6.6 is shown the maximum improvement (with underfitting) of the simple linear regression vs ARToolKit approach is of 28.01862mm and the minimum improvement is of 11.9335mm. In this case, there is a significant improvement from the proposed method and the simple linear regression.

In Figure 6.7 is shown the cumulative improvement (which is the sum of all the improvements in each position) of the proposed method vs simple linear regression in this sample is of 20.85%. In Figure 6.8 is shown the improvement of the underfitting (using less data) of sample 2, the cumulative improvement is of 1863.63%. In this case, the proposed method have more improvement when it is used less data.

Therefore, using this camera, it is clearly that the proposed method have better results than the simple linear regression, and that the proposed method much have better performance when it is used less data than the simple linear regression.

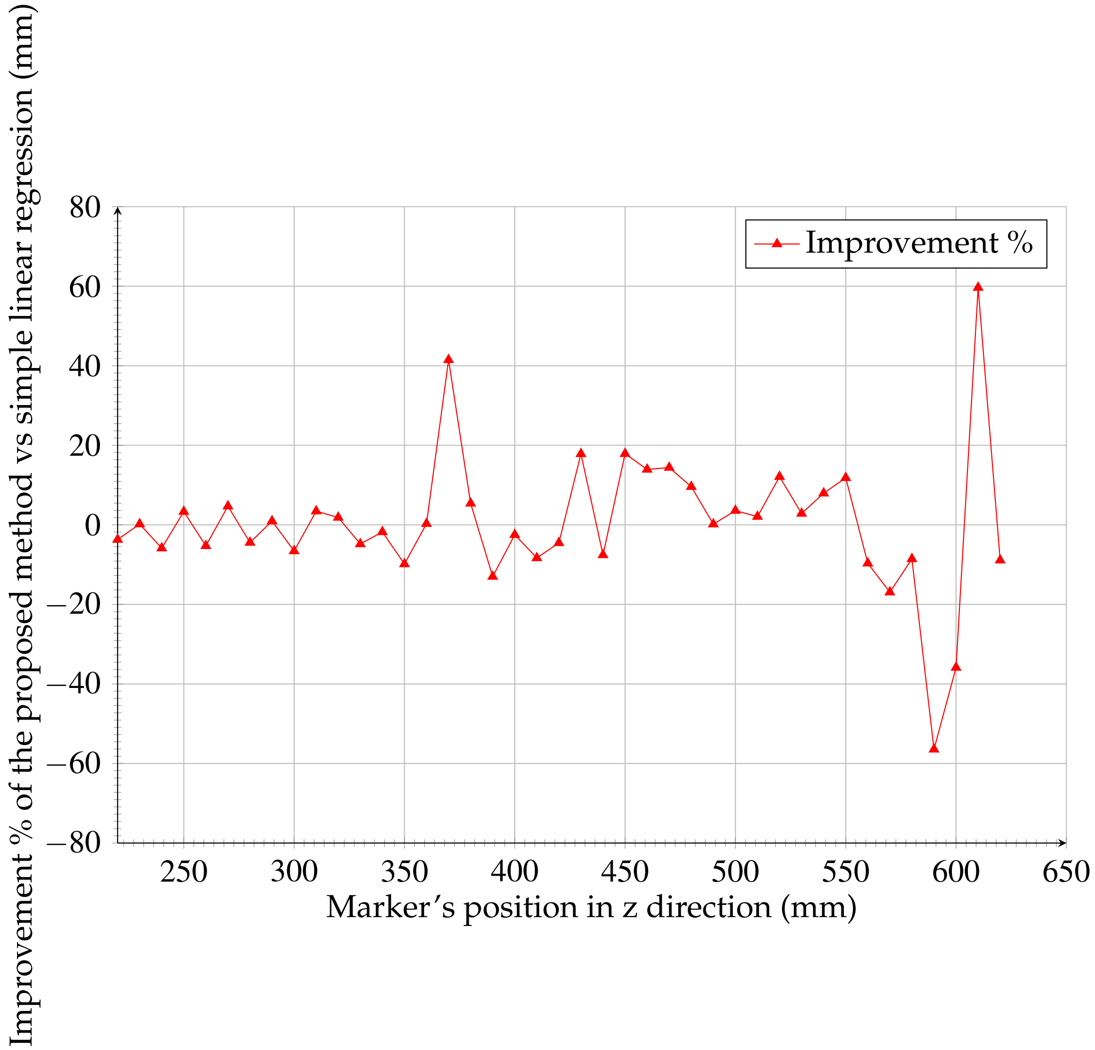


Figure 6.7: Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample 3

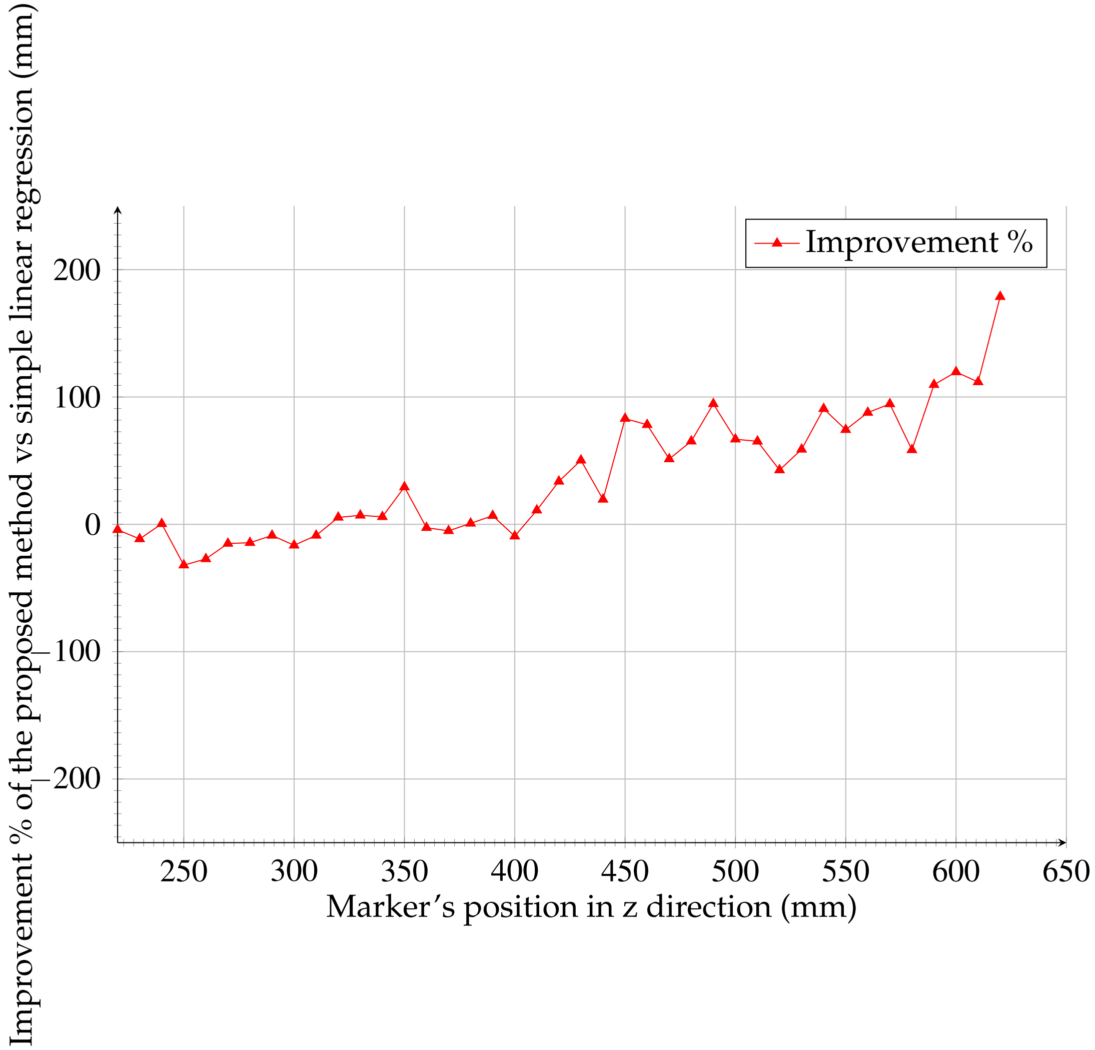


Figure 6.8: Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample 3 with underfitting

6.5 Experiments using a camera web Acteck CW-760 standard

In this experiment, it was used a camera web Acteck CW-760 standard for taking samples of marker's position estimation. Figure 6.9 shows that the light range used in this samples goes from 26 to 250 lux, where the light changes dynamically. In this sample, the mean of the marker's position error using ARToolKit approach is 133.3852mm. In the case of underfitting, the mean of the marker's position error using ARToolKit approach is 134.0064mm.

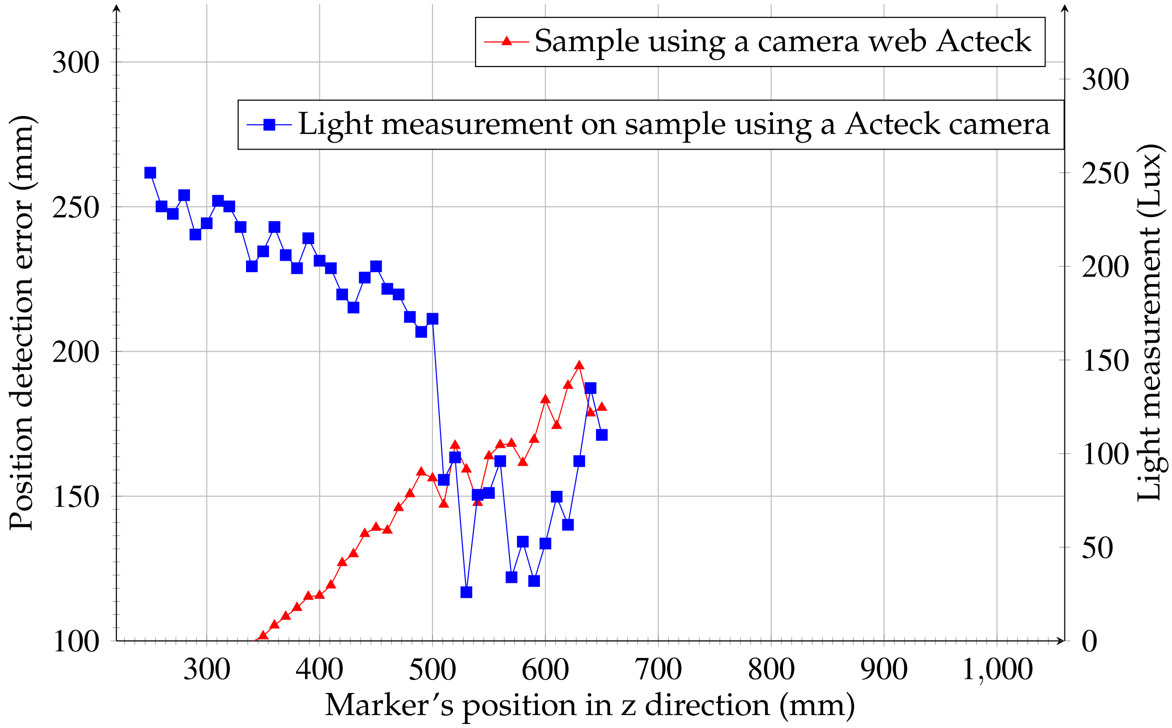


Figure 6.9: Experimental results of position detection error on the sample using a camera web Acteck

In Table F.1 are shown the fit models generated for the proposed method and the simple linear regression. Also, in Table F.2 are shown the underfit models generated for the proposed method and the simple linear regression.

The Figure 6.10 shows the cumulative improvement of different approaches for the marker's position estimation in z-direction using a camera web Acteck CW-760 standard. The improvement of the proposed approach and the simple linear regression approach are compared to ARToolKit. In this sample using a camera web Acteck CW-760 standard, it was found that using the proposed method the marker's position estimation error can be reduced. The mean of the position error using the proposed approach is of 2.227mm. Using this approach, it's also obtained better results than using only ARToolKit approach. In the case of underfitting, the mean of the position error using the proposed approach is of 2.5759mm.

Furthermore,the proposed approach has better results if the training set is increased. The marker's position error was reduced 98.33% in this scenario. Furthermore, nearly 99.08% of the total variability in the response variable (marker's position) is accounted for by the estimated marker's position by ARToolkit and the light measurement. In this experiment, the regression line models have a $R^2 = 0.9757$,$R^2 = 0.9972$ and $R^2 = 0.9997$ respectively that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker's position and the light conditions).

In the case of underfit proposed method, the marker's position error was reduced 97.84% in this scenario. Furthermore, nearly 99.41% of the total variability in the response variable (marker's position) is accounted for by the estimated marker's position by ARToolkit and the light measurement. In this experiment the regression line models have a $R^2 = 0.9827$,$R^2 = 1.00$ and $R^2 = 0.9998$ respectively that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker's position and the light conditions).

In Figure 6.10 is shown the maximum improvement of the proposed method vs the ARToolKit approach is of 187.6010mm and the minimum improvement is of 71.2973mm. In Figure 6.10 is shown the maximum improvement of the simple linear regression vs ARToolKit approach is of 188.5919mm and the minimum improvement is of 71.4334mm. In this case, both approaches have almost the same performance.

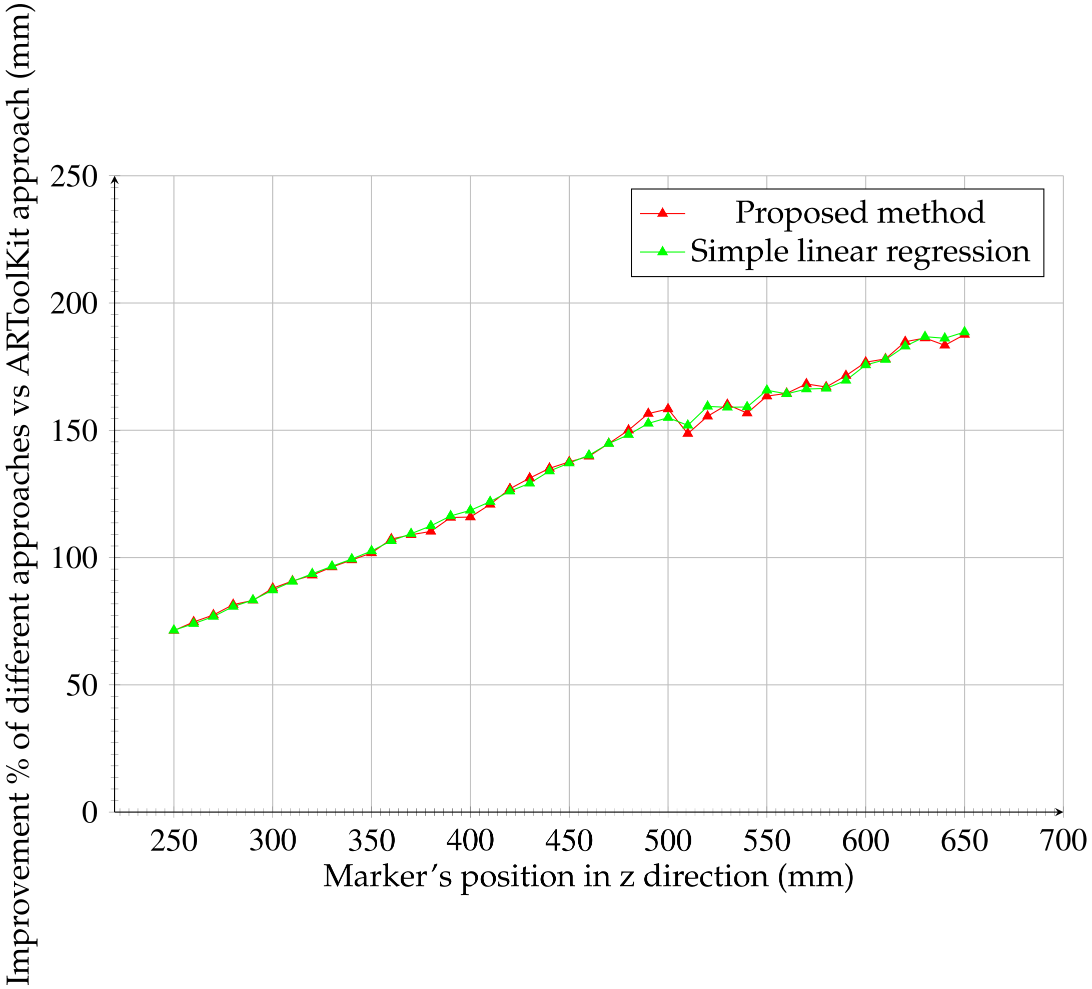


Figure 6.10: Experimental results of improvement of proposed approach and simple regression approach on Sample using a camera web Acteck

In Figure 6.11 is shown the maximum improvement of the underfit proposed method vs the ARToolKit approach is of 186.5510mm and the minimum improvement is of 71.9584mm. In Figure6.11 is shown the maximum improvement of the underfit simple linear regression vs ARToolKit approach is of 188.3479mm and the minimum improvement is of 72.03581mm. In this case, the simple linear regression have a slightly better performance than the proposed method.

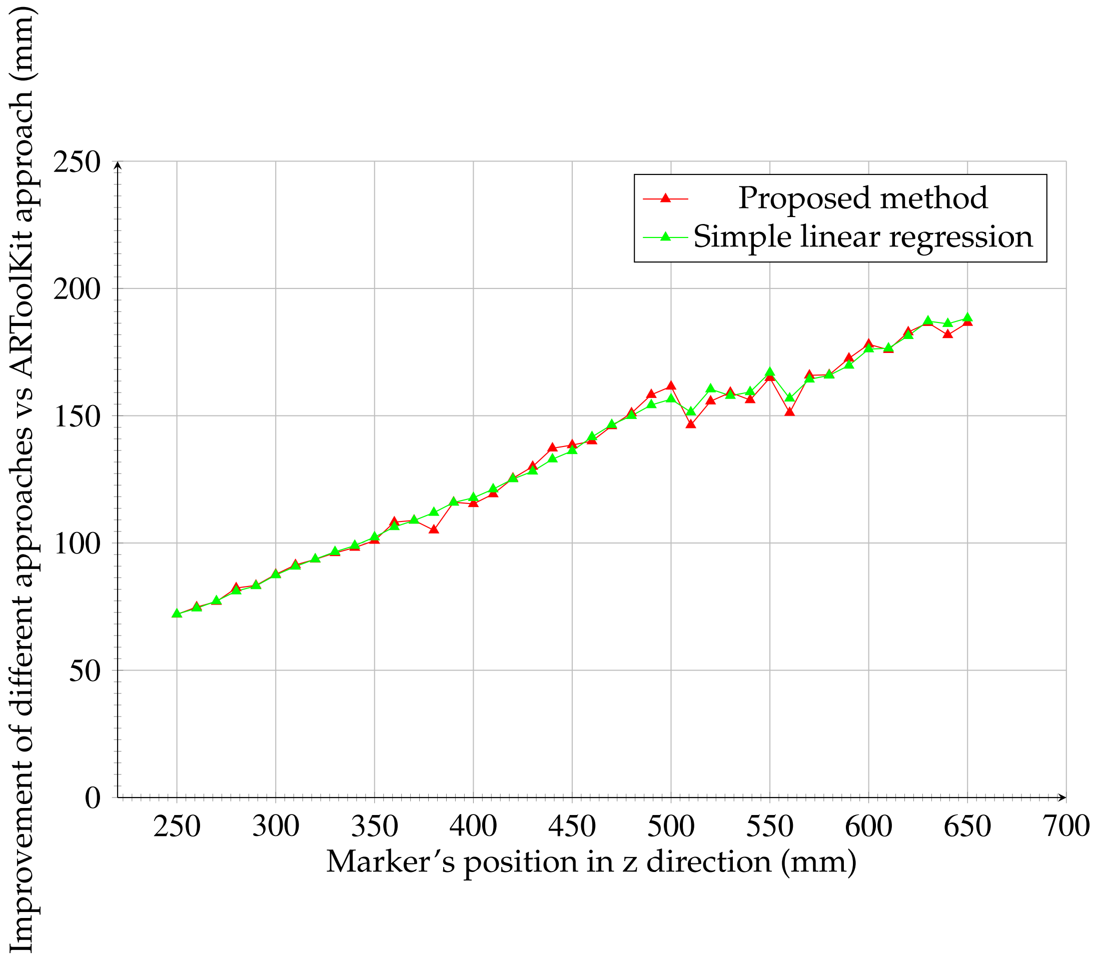


Figure 6.11: Experimental results of improvement of proposed approach and simple regression approach on Sample using a camera web Acteck with underfitting

In Figure 6.12 is shown the cumulative improvement (which is the sum of all the improvements in each position) of the proposed method vs simple linear regression in this sample is of 0.5028%. In Figure 6.13 is shown the improvement of the underfit sample 2, the cumulative improvement is of -7.6209%. In this case, only in when it is used enough data the proposed method have a slightly better performance, and when it is used less data the simple linear regression have a slightly better performance.

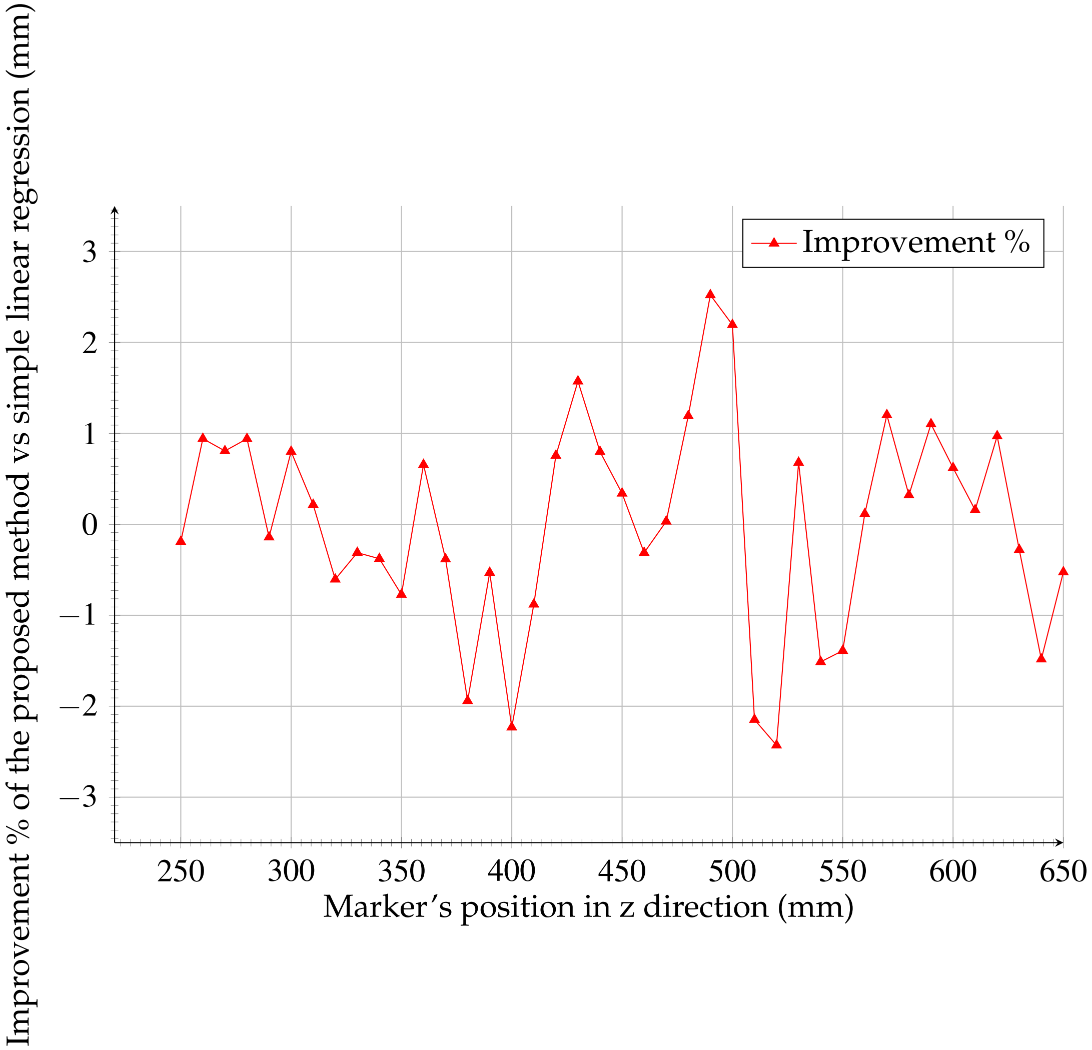


Figure 6.12: Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample using a camera web Acteck

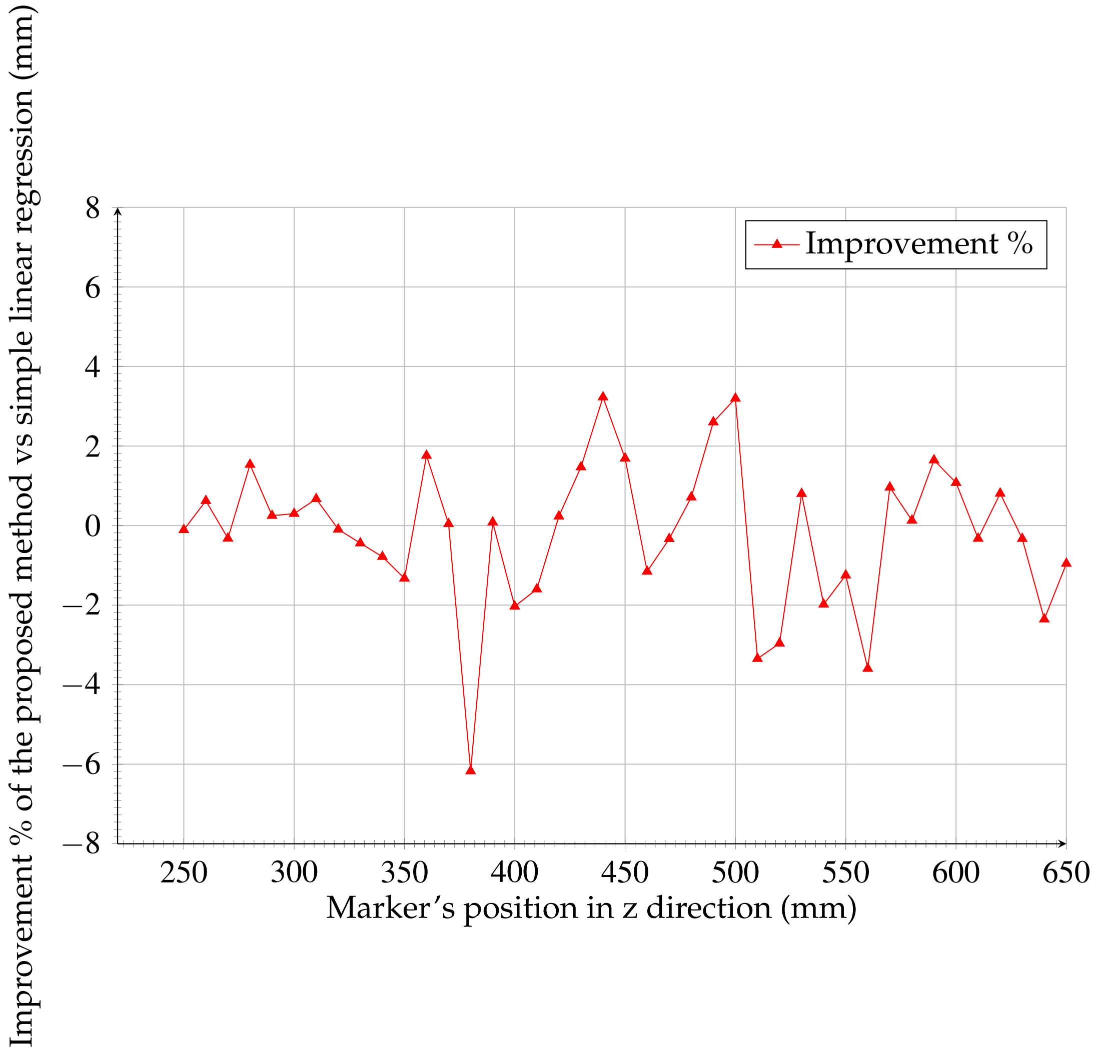


Figure 6.13: Experimental results of improvement % of proposed approach vs simple linear regression approach on the Sample with underfitting using a camera web Acteck

6.6 Experiments using a Kinect 360 Xbox

In this experiment, it was used a Kinect 360 Xbox for taking samples of marker's position estimation. Figure 6.14 shows that the light range used in this samples goes from 130 to 249 lux, where the light changes dynamically. In this sample, the mean of the marker's position error using ARToolKit approach is 209.9307mm. In the case of underfitting, the mean of the marker's position error using ARToolKit approach is 211.9582mm.

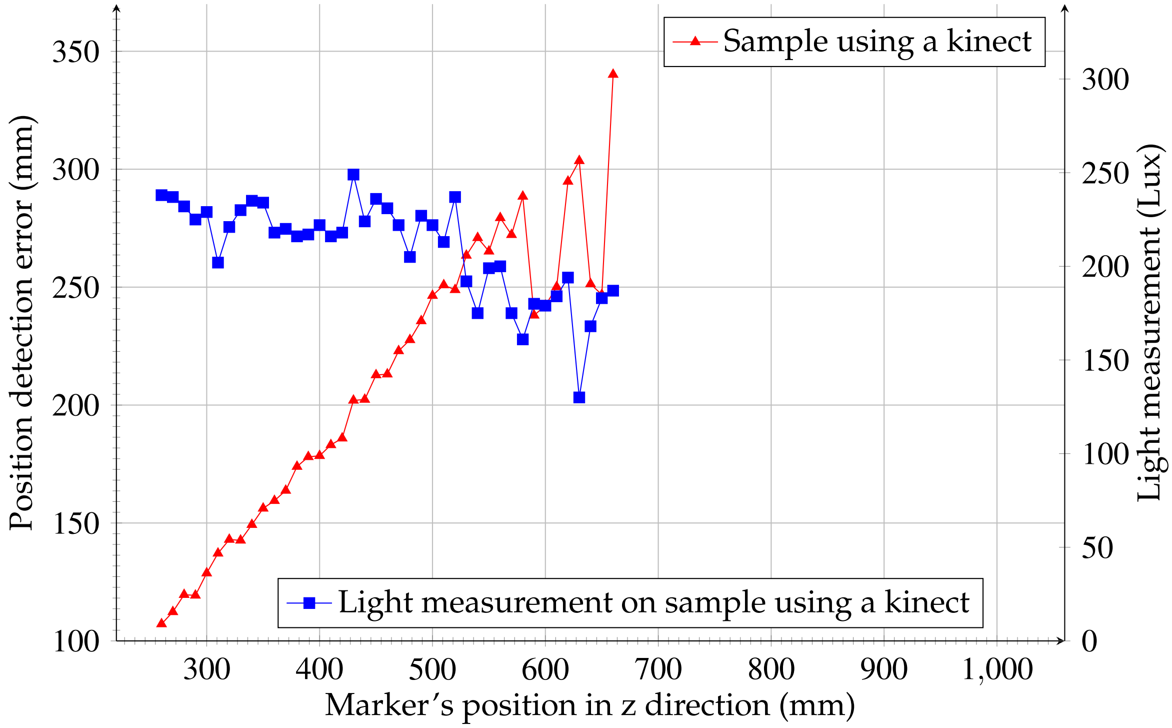


Figure 6.14: Experimental results of position detection error on the sample using a Kinect

In Table F.1 are shown the fit models generated for the proposed method and the simple linear regression. Also, in Table F.2 are shown the underfit models generated for the proposed method and the simple linear regression.

The Figure 6.15 shows the cumulative improvement of different approaches for the marker's position estimation in z-direction using a Kinect Xbox 360. The improvement of the proposed approach and the simple linear regression approach are compared to ARToolKit. In this sample using Kinect, it was also found that using the proposed method the marker's position estimation error can be reduced. The mean of the position error using the proposed approach is of 2.166mm, additionally, the mean of the marker's position estimation error using ARToolKit approach is 209.9307mm. Using the proposed approach, it's also obtained better results than using only ARToolKit approach. In the case of underfitting, the mean of the position error using the proposed approach is of 4.2455mm.

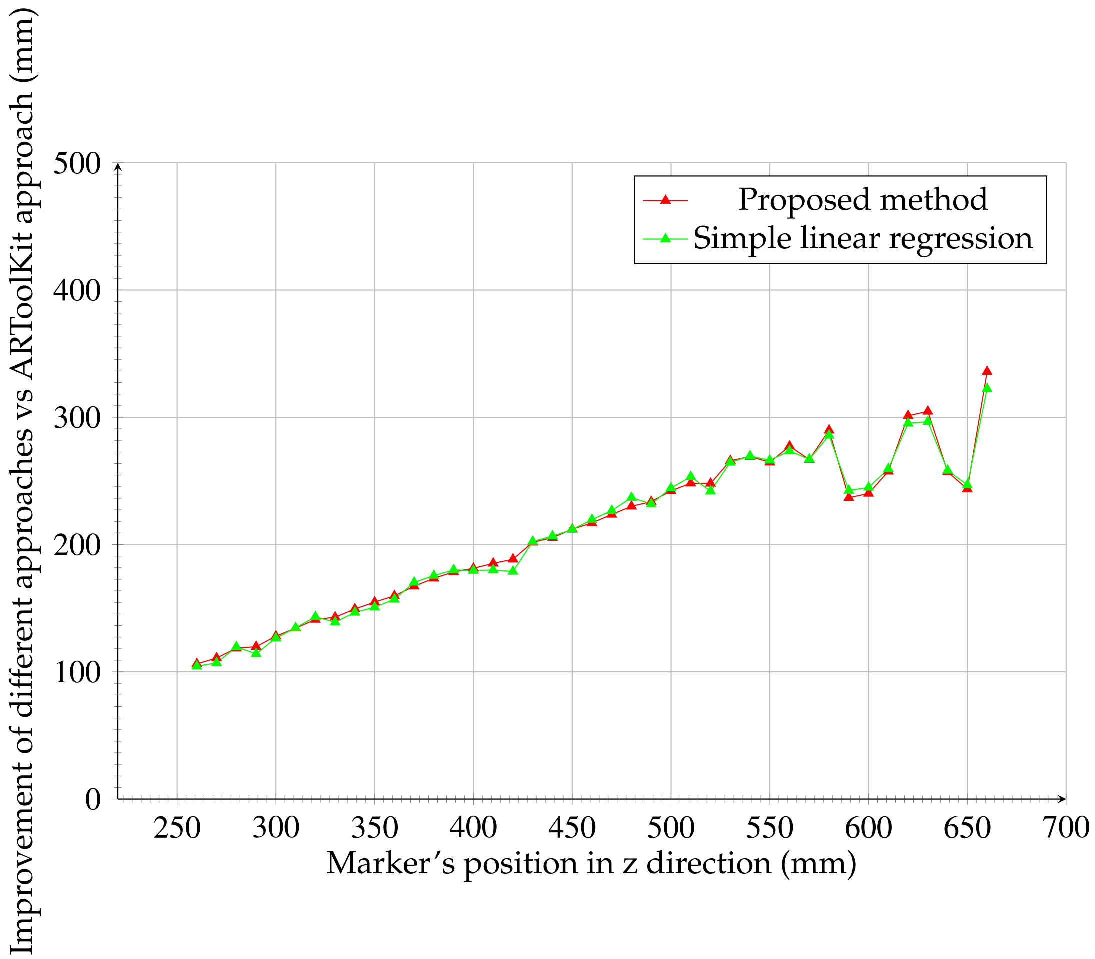


Figure 6.15: Experimental results of improvement of proposed approach and simple regression approach on Sample using a kinect

Furthermore, the proposed approach has better results if the training set is increased. The marker's position error was reduced 98.97% in this scenario. Furthermore, nearly 99.25% of the total variability in the response variable(marker's position) is accounted for by the estimated marker's position by ARToolKit and the light measurement. In this experiment the regression line models have a $R^2 = 0.999$ and $R^2 = 0.986$ that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker's position and the light conditions).

In the case of underfit proposed method, the marker's position error was reduced 97.97% in this scenario. Furthermore, nearly 99.64% of the total variability in the response variable (marker's position) is accounted for by the estimated marker's position by ARToolkit and the light measurement. In this experiment the regression line models have a $R^2 = 0.9936$ and $R^2 = 0.9992$ respectively that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker's position and the light conditions).

In Figure 6.15 is shown the maximum improvement of the proposed method vs the ARToolKit approach is of 335.8237mm and the minimum improvement is of

106.2425mm. In Figure 6.15 is shown the maximum improvement of the simple linear regression vs ARToolKit approach is of 322.3875mm and the minimum improvement is of 104.4134mm. In this case, the proposed method have a slightly better performance than the simple linear regression.

In Figure 6.17 is shown the maximum improvement of the underfit proposed method vs the ARToolKit approach is of 371.5629mm and the minimum improvement is of 106.6792mm. In Figure6.17 is shown the maximum improvement of the underfit simple linear regression vs ARToolKit approach is of 326.8494mm and the minimum improvement is of 105.4133mm. In this case, the proposed method have a slightly better performance than the simple linear regression.

In Figure 6.16 is shown the cumulative improvement (which is the sum of all the improvements in each position) of the proposed method vs simple linear regression in this sample is of 22.04%. In Figure 6.18 is shown the improvement of the underfit sample 2, the cumulative improvement is of 43.1221%. In this case, the proposed method have a better improvement when it is used less data than the simple linear regression.

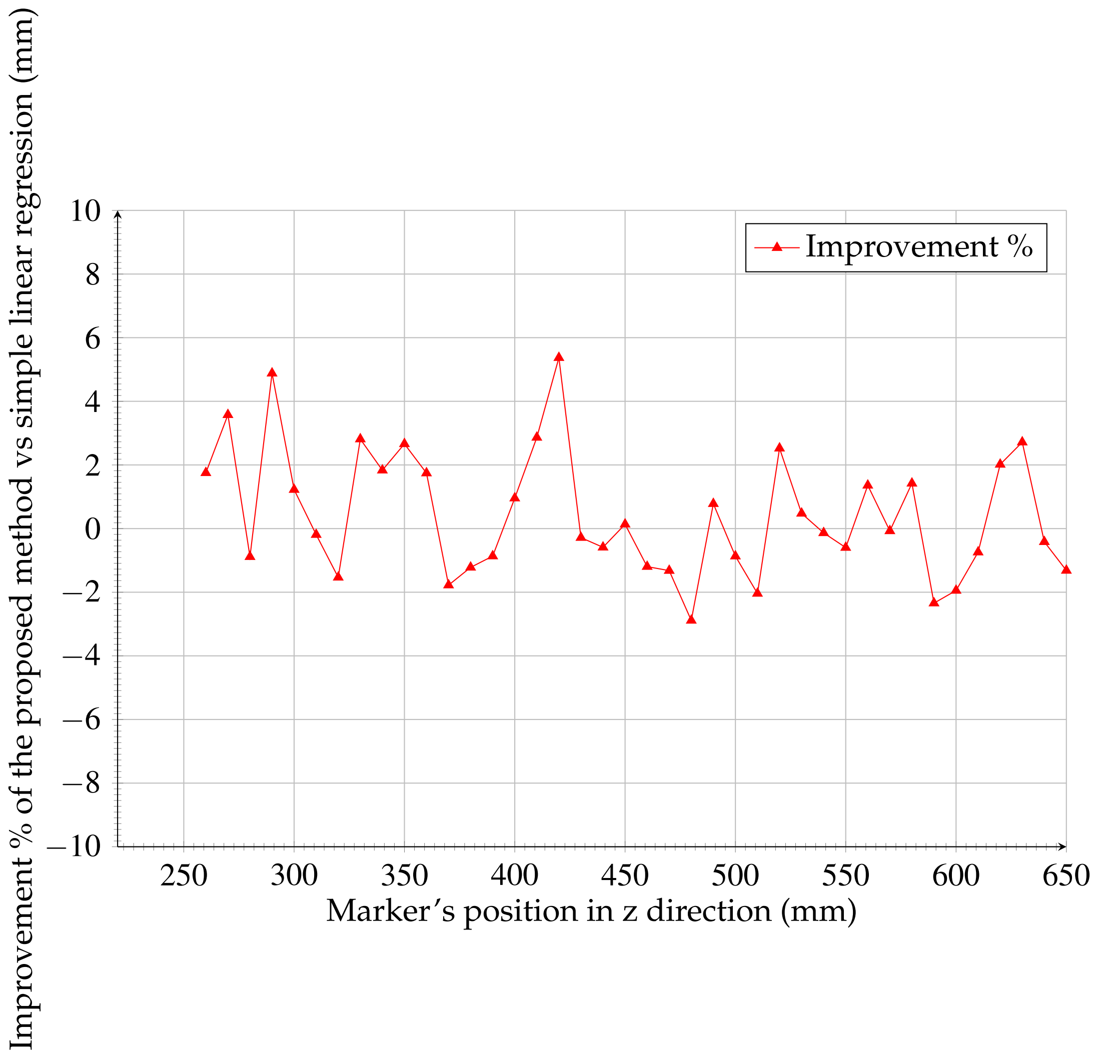


Figure 6.16: Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample using a kinect

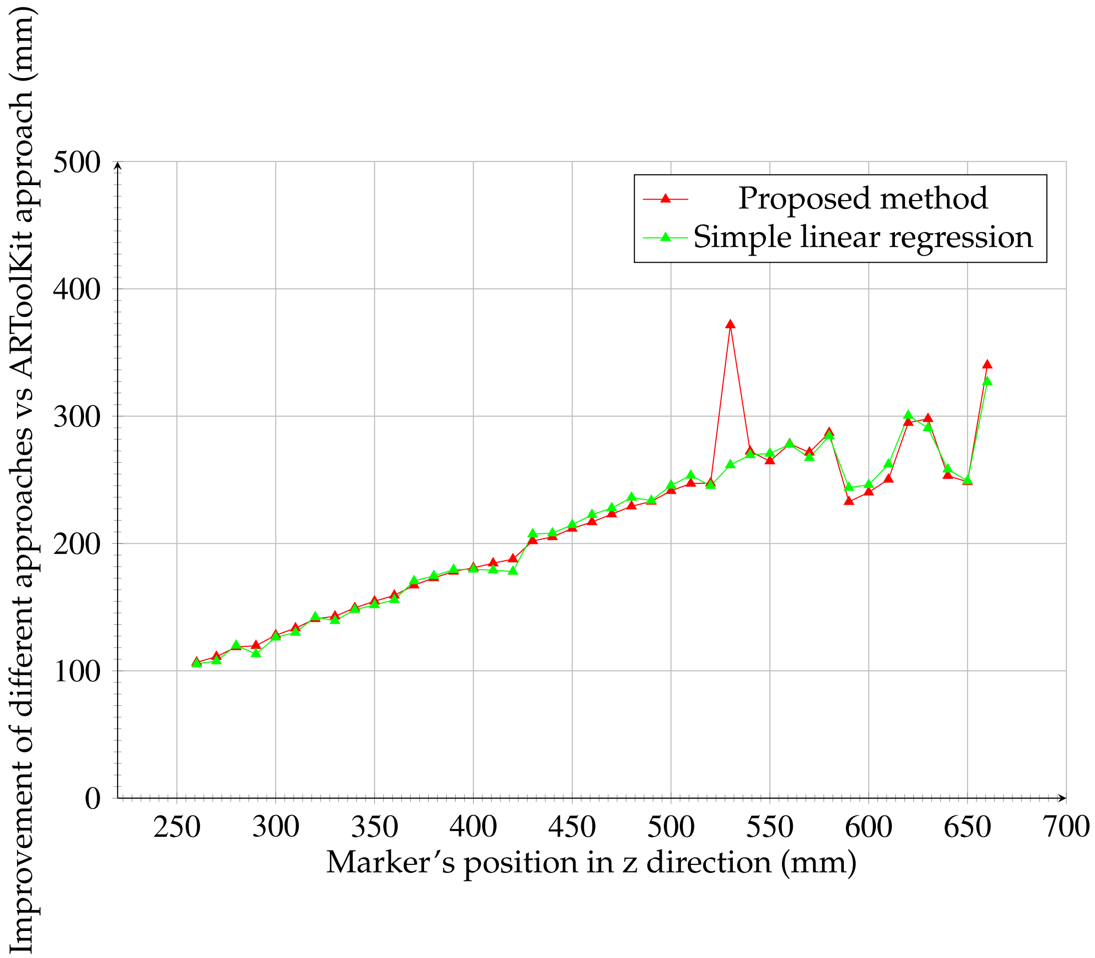


Figure 6.17: Experimental results of improvement of proposed approach and simple regression approach on Sample using a kinect with underfitting

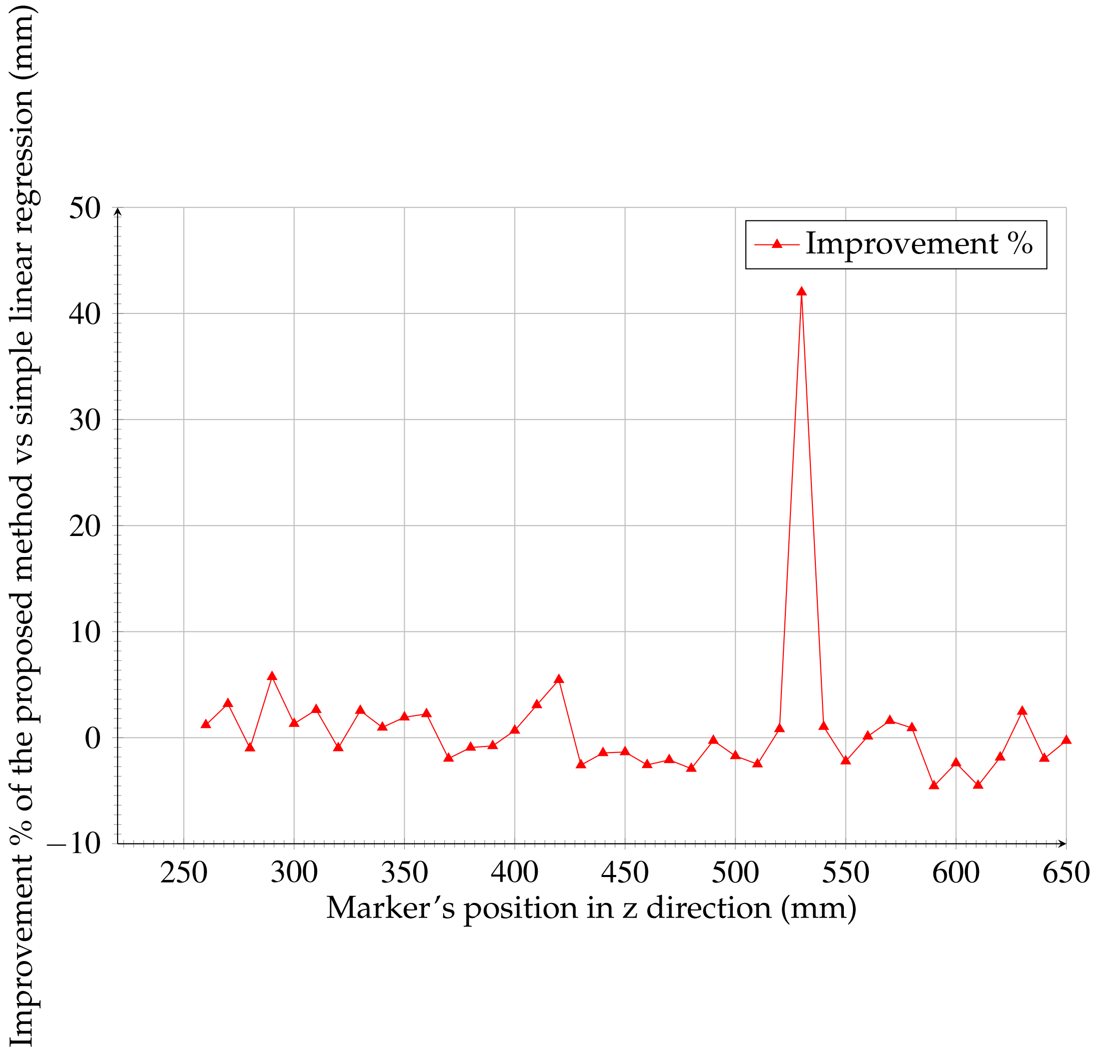


Figure 6.18: Experimental results of improvement % of proposed approach vs simple linear regression approach on Sample with underfitting using a kinect

6.7 Overfitting and Underfitting results

Overfitting and Underfitting are common problems for training in machine learning. Overfitting refers that the model fit the data almost 100%, which means that the model have noise and random fluctuations taken into account. The problem with this is that the model cannot be used with new data or sample and the model only works too well in the training data.

In machine learning, it is better to have a model that has a constant performance with new data than a model that only works with the initial training data. The model mustn't explain 100% of the data, it must explain enough (like 80%) to be used with new training sets and have a constant performance.

In the case of underfitting, the model is not good enough to be used. In machine learning, an underfit model is not a suitable model and have a poor performance on the training data.

In this case, it was taken 400 replicates in each position, but only it was taken into account the last estimation of those 400 replicates. The samples were taken in 41 positions each 10mm. Using these training data, the machine learning algorithm is going to be used with those 41 position estimations and with 20 positions in order to know the performance of the underfit models.

In Table 6.3 are shown how the fit models (where all the data of samples was taken into account) have a significant or slightly better performance than the underfit models (where only half of the data was taken into account).

Table 6.3: Results of fit and underfit models for marker's position estimation

Results of fit and underfit models for marker's position estimation		
Camera	Method	Reduction of error
Camera Logitech	Underfit proposed method	62%
	Fit proposed method	87%
Camera Acteck	Underfit proposed method	97.84%
	Fit proposed method	98.33%
Kinect	Underfit proposed method	97.97
	Fit proposed method	98.97%

Chapter 7

Conclusions

A novel method for estimating marker position under semi-controlled environment in which the lighting conditions, brightness and contrast level change dynamically is proposed. This method uses multiple regression technique based on machine learning using as variables: the light measurements and estimations of marker's positions. The proposed method based on machine learning is to *learn* the relationship between the light and the marker's position estimations. The advantage of this method is the reduction of noise caused by light and other factors as extrinsic and intrinsic parameters of the camera. After the first *training* process, this method can be used in an uncontrolled environment where the light changes. The accuracy of the estimations using this method increases as the training sets increases. A better tracking can be performed after the first *training*. In our study case, it was used the proposed method to increase the accuracy of the estimation of marker's position for the *tracking* of pieces using markers of *QR* pattern, where a pick-and-place robot receives the piece (or marker's) position in order to grab the piece and place it in their place (for assembly). The accuracy of the estimations of the marker's position in the experiments goes from 99% to 99.25%. In the case of underfitting, the accuracy of the estimations of the marker's position in the experiments goes from 98.18% to 99.64%.

7.1 Contributions

This research was presented as a solution for a particular problem in the area of tracking. In augmented reality, there is not much research of maker's tracking in augmented reality using cameras or kinect. Furthermore, there isn't any research about the effect of light on marker's tracking. Indeed, it was studied the effect of light in different marker's positions in order to know how relevant for the error reduction or marker's position estimation is. The experiments were taken using ground truth (CNC machine tool guide) in different position in a range of 400mm with different light conditions.

7.2 Initial questions

In this research it was proposed several questions that in this work should be answered.

1. How to reduce error or noise of the position estimation of a marker?
2. Does the light affect the marker's position estimation?
3. How to measure the effect of light in the marker's position estimation?

The marker's position estimations have noise of different factors. The extrinsic and intrinsic parameters of the camera cause noise to the marker's position estimations. Another variable that cause noise to the marker's position is the brightness, illumination and occlusion. In order to get a reduction of this noise, it was proposed to measure the light in order to get a measurement to reduce the noise of the illumination and brightness. Furthermore, it was proposed to use machine learning and multiple regression technique in order to *learn* the noise cause by the light and other noise. After studying the data, it was concluded that the data has a normal distribution, then a linear multiple regression was proposed in order to do predictions of marker's position, which is *key* to get a better estimations.

It was studied the data of the samples under different light conditions, where the light changes dynamically and has a big variation depending on the position in the real world. As a result, it was concluded that the data can be subdivided into groups. Each group should have the same variance of error of marker's position estimation. Using these groups, it was set each *light range*. It was concluded that the best way to have more accuracy was to have multiple predictor functions for each *light range* for marker's position estimation. The proposed approach improves the accuracy of the camera-pose estimation under lightning conditions that can change dynamically.

In this experiment, the light conditions were different in each sample taken, which affected the position estimations, but the noise of light conditions and camera parameters was significantly reduced using the machine learning approach. The position error does not have a constant tendency because it depends on many factors such as the camera parameters and the lighting conditions. Additionally, our method improves the results when more training data is given.

In order to measure the effect of brightness, illumination and darkness, it is proposed to measure brightness of light reaching the marker before it is reflected, which is measured in *lux*. The sensitive cell always faces the camera. The Figure 7.1 is depicted how to take a light measurement of the marker using a luxmeter.

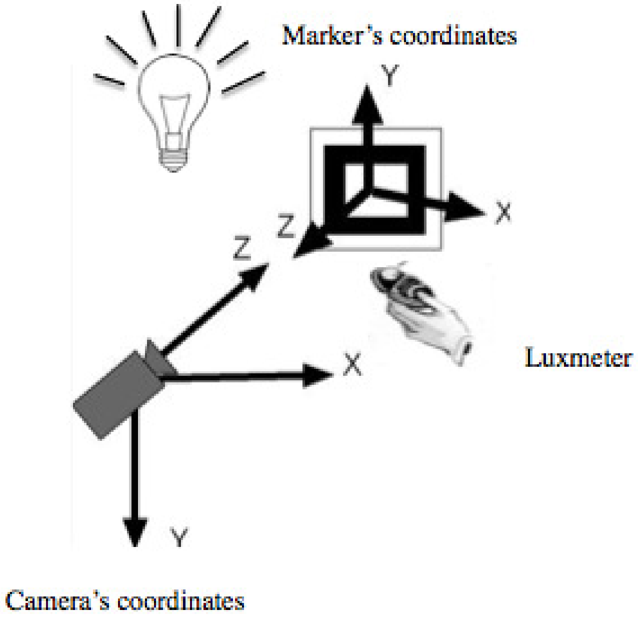


Figure 7.1: Light measurement system

The reduction of error is achieved because of the key elements as multiple linear regression, the divided data by *light range*, and the measurement of brightness or illumination noise. It was studied the patterns of the data (marker's position estimations using ARToolKit libraries) under different light conditions. In this research, it was concluded that the data has a normal distribution. Furthermore, it was established the *light ranges* that are used to divide the data. Then, it was established the right type of model for this data, i.e. *multiple linear regression*. It was concluded that machine learning is the best way to *learn* and get better estimation for marker's position estimation (based on the *distribution* of the data).

7.3 Limitations

Although this method increases the accuracy (in the experiments), which goes from 98.18% to 99.64%, this method have some limitations of usage. The camera cannot detect the marker when it is out of range (when the light is under 40 lux or when its above the 310 lux). The system needs an initial *training* to work. Another limitation is the amount of *training sets* that is given to the system because the system depends on the training set in order to get better predictor function. Furthermore, the occlusion problem wasn't studied in this research. The noise of the extrinsic and intrinsic parameters of the camera are not measured in this research.

Another limitation of this method is that the conditions of the *training set*. The samples of the marker's position estimation are taken in different distances from the camera to the marker. The predictor functions generated using this method only works under the same range of distance that the *training sets* are taken. In this experiment, the *training sets* are taken from 220 mm to 1060 mm.

7.4 Scope of Applicability

This method is used in order to get a better *tracking* of the marker. In this experiment, it was used the marker to locate pieces to be assembled, and are grabbed and placed using a pick-and-place robot. The position in marker's coordinates is converted to the pick-and-place robot coordinates. The marker's position estimations are used to pick-and-place the piece (to be assembled) using the robot.

In Figure 7.2 is depicted the pieces that must be grabbed and placed using the arm manipulator. The pieces have a marker to locate the position of each piece.

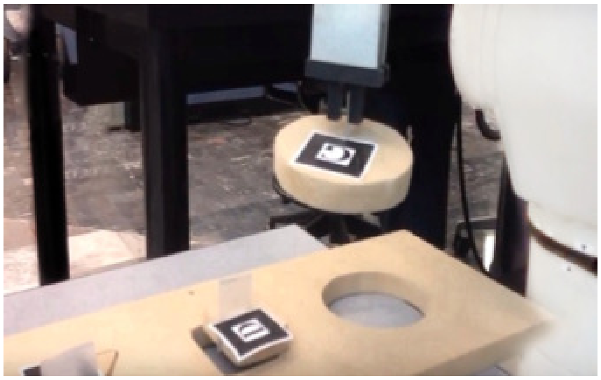


Figure 7.2: Study case: Pieces to be assembled

In Figure 7.3 is depicted the pick-and-place robot, the camera system, and the pieces to be placed in their respective place.

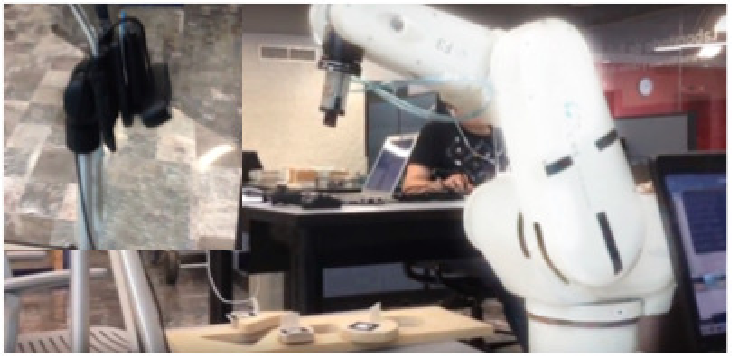


Figure 7.3: Study case: Pick-and-place robot and camera system

In Figure 7.4 is depicted the vision system used to estimate the position of each piece to be assembled.

Table 7.1: Tracking system approaches

Treatments	**Light changes**	**Method**	**Multiple cameras**
Rabbi's approach [53]	X	Multiple files	
Maidi's approach [41]		Statistical method	
Herout's approach [28]		Mathematical method	
Dhiman's approach [16]		Algebraic method	X
Yamauchi's approach [74]		Algebraic method	
Freeman's approach [24]		Statistical method	
Wang's approach [70]		Hierarchical contour analysis	
Our's approach	X	Statistical method	

Figure 7.4: Study case: Vision system interface

7.5 Comparative issues

In Table 7.1 are shown the different approaches for tracking markers for Augmented Reality. There are different methods used based in different techniques. These methods use statistical analysis, mathematical methods, algebraic method, and so on. Only one of these methods mentioned above take into account the light changes. This system uses stereo vision (multiple cameras).

This new approach of using machine learning for fitting AR marker estimations taking into account light measurement is a novel way to reduce noise and understand better the relationships between light measurement and AR marker position estimations.

7.6 Future research

Future research will be focused on studying the light noise and the noise of the extrinsic and intrinsic parameters of the camera. Future research of the measurement of the noise is very important to increase the accuracy of the marker's tracking.

The marker's tracking problem is going to be studied to increase the accuracy and be used in the industry, medical applications, or other application that need more accuracy.

Future research of the light noise under different distances from the camera to the marker must be done. Furthermore, the study of different types of the effect of light in the marker's tracking.

Appendix A

Optimal Linear Prediction

In linear regression, a response variable, called **Y**, is estimated using a p-dimensional vector of prediction variables or features $\vec{X}$. In order to find the optimal predictor function, it is used the conditional expectation of the p-dimensional vector of prediction variables or features $\vec{X}$, as shown in equation A.1.

$$r(\vec{x}) = E[Y|\vec{X} = \vec{x}] \tag{A.1}$$

The conditional expectation $r(\vec{x})$ is approximated by a linear function of $\vec{x}$, say $\vec{x} \cdot \beta$. This approximation is a decision (or choice) (for a person or the automated program). This approximation can be accurate. Using the Taylor series, the function $r(\vec{x})$ can be expanded about a point, say $\vec{u}$:

$$r(\vec{x}) = r(\vec{u}) \sum_{i=1}^{p} \left(\left. \frac{\partial r}{\partial x_i} \right|_{\vec{u}} \right) (x_i - u_i) + \mathscr{O}(||\vec{x} - \vec{u}||^2) \tag{A.2}$$

or, in more compact vector calculus notation,

$$r(\vec{x}) = r(\vec{u}) + (\vec{x} - \vec{u}) \cdot \nabla r(\vec{u}) + \mathscr{O}(||\vec{x} - \vec{u}||^2) \tag{A.3}$$

In optimization of the predictor function, the points $\vec{x}$ are close to $\vec{u}$, then the terms $\mathscr{O}(||\vec{x} - \vec{u}||^2)$ are small, so the linear approximation is good. In order to find the best function, the mean-squared error must be minimized again:

$$MSE(\beta) = E[(Y - \vec{X} \cdot \beta)^2] \tag{A.4}$$

Going through the optimization is parallel to the one-dimension case, with the conclusion that the optimal β is

$$\beta = V^{-1} Cov[\vec{X}, Y] \tag{A.5}$$

where V is the covariance matrix of $\vec{X}$, i.e., $V_{ij} = Cov[X_i, X_j]$, and $Cov[\vec{X}, Y]$ is the vector of covariances between the predictor variables and Y,i.e. $Cov[\vec{X}, Y]_i = Cov[X_i, Y]$.

Appendix B

Optimal linear predictor

In order to estimate β, the following probabilistic assumptions are needed: firstly, the observations $(\vec{X}_i, Y_i)$ are independent for different values of i, with unchanging covariances. Then, the sample covariances will coverge on the true covariances:

$$\frac{1}{n}X^TY \to Cov[\vec{X}, Y] \tag{B.1}$$

$$\frac{1}{n}X^TX \to V \tag{B.2}$$

where as before X is the data-frame matrix with one row for each data point and one column for each feature, and similarly for **Y**.

So, by continuity,

$$\hat{\beta} = (X^TX)^{-1}X^TY \to \beta \tag{B.3}$$

and it is a consistent estimator.

Furthermore, the residual sum of squares

$$RSS(\beta) \equiv \sum_{i=1}^{n}(y_i - \vec{x}_i \cdot \beta)^2 \tag{B.4}$$

is going to be minimized. The minimizer is the same $\hat{\beta}$ we got by plugging in the sample covariances. In this case, there is no need probabilistic assumptions.

Appendix C

ARToolKit-Kinect code for marker's position estimation

The solution to get the marker′s position using a kinect or a camera is written in C++. The solution project is composed of the next files:

1. main.cpp
2. MyKinect.cpp
3. MyKinect.h
4. MyVector.h
5. Math.h
6. Common.h
7. Util.h

```
#ifdef _WIN32
#include <windows.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#ifndef __APPLE__
#include <GL/gl.h>
#include <GL/glut.h>
#else
#include <OpenGL/gl.h>
#include <GLUT/glut.h>
#endif
#include <AR/gsub.h>
#include <AR/video.h>
#include <AR/param.h>
#include <AR/ar.h>

//#include "Common.h"
#include "MyKinect.h"
#include "HandDetectorOpenNI.h"

/* set up the video format globals */

#ifdef _WIN32
char             *vconf = "Data\\WDM_camera_flipV.xml";
#else
char             *vconf = "";
#endif

int             xsize = 640;
int         ysize = 480;
int             thresh = 100;
int             count = 0;

int             mode = 1;
char           *cparam_name    = "Data/MyCameraParameter4.dat";
ARParam         cparam;
char           *patt_name      = "Data/patt.g"; //"Data/patt.hiro"
int             patt_id;
int             patt_width     = 40.0;
double          patt_center[2] = {0.0, 0.0};
double          patt_trans[3][4];

float            size = 50.0f;
int              displayMode =1;
bool             drawFromKinect = false;

//MyKinect object
MyKinect         g_MyKinect;
HandDetectorOpenNI g_HandDetectorOpenNI;

static void   init(void);
static void   cleanup(void);
static void   keyEvent( unsigned char key, int x, int y);
static void   mainLoop(void);
static void   draw( double trans[3][4] );

void ClickPointerFunction()
{
      size += 10.0f;
```

```
}

int main(int argc, char **argv)
{
      printf("Sample1\n");
      glutInit(&argc, argv);
        init();

      //init for Kinect
      g_MyKinect.Init();

      g_MyKinect.StartGeneratingAll();

      arVideoCapStart();
         argMainLoop( NULL, keyEvent, mainLoop );
      return (0);
}

static void   keyEvent( unsigned char key, int x, int y)
{
    /* quit if the ESC key is pressed */
    if( key == 0x1b ) {
        printf("*** %f (frame/sec)\n", (double)count/arUtilTimer());
        cleanup();
        exit(0);
    }

    if( key == 'c' ) {
        printf("*** %f (frame/sec)\n", (double)count/arUtilTimer());
        count = 0;

        mode = 1 - mode;
        if( mode ) printf("Continuous mode: Using arGetTransMatCont.\n");
         else      printf("One shot mode: Using arGetTransMat.\n");
    }
      if(key == '1')
            displayMode = 1;
      if(key == '2')
            displayMode = 2;
      if(key == '3')
            displayMode = 3;
      if(key == 'k')
            drawFromKinect = true;
      if(key == 'c')
            drawFromKinect = false;
}

/* main loop */
static void mainLoop(void)
{
    static int      contF = 0;
    ARUint8         *dataPtr;
    ARMarkerInfo    *marker_info;
    int             marker_num;
    int             j, k;

      //update new data
      g_MyKinect.Update();
      if(drawFromKinect)
      {
            //get image data to detect marker
            if( (dataPtr = (ARUint8 *)g_MyKinect.GetBGRA32Image()) == NULL
) {
                  arUtilSleep(2);
                  return;
            }
      }
      else
      {
            /* grab a vide frame */
            if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
                  arUtilSleep(2);
                  return;
            }
      }


    if( count == 0 ) arUtilTimerReset();
    count++;

    /* detect the markers in the video frame */
    if( arDetectMarker(dataPtr, thresh, &marker_info, &marker_num) < 0 )
{
        cleanup();
        exit(0);
    }

      if(drawFromKinect)
      {
            //option . You can choose many display mode. image, Depth by
Color, depth mixed image
            if(displayMode == 2)
                  dataPtr = (ARUint8 *)g_MyKinect.GetDepthDrewByColor();
            else
                  if(displayMode == 3)
                        dataPtr = (ARUint8
*)g_MyKinect.GetDepthMixedImage();
      }

       argDrawMode2D();
       argDispImage( dataPtr, 0,0 );

       arVideoCapNext();

    /* check for object visibility */
    k = -1;
    for( j = 0; j < marker_num; j++ ) {
        if( patt_id == marker_info[j].id ) {
            if( k == -1 ) k = j;
            else if( marker_info[k].cf < marker_info[j].cf ) k = j;
        }
    }
    if( k == -1 ) {
        contF = 0;
        argSwapBuffers();
        return;
    }
    /* get the transformation between the marker and the real camera */
    if( mode == 0 || contF == 0 ) {
        arGetTransMat(&marker_info[k], patt_center, patt_width,
patt_trans);
    }
    else {
        arGetTransMatCont(&marker_info[k], patt_trans, patt_center,
patt_width, patt_trans);
    }
    contF = 1;

    draw( patt_trans );

    argSwapBuffers();
}

static void init( void )
{
    ARParam  wparam;
/********************************VIDEO*******************************
**************/
    /* open the video path */
    if( arVideoOpen( vconf ) < 0 ) exit(0);
    /* find the size of the window */
    if( arVideoInqSize(&xsize, &ysize) < 0 ) exit(0);
    printf("Image size (x,y) = (%d,%d)\n", xsize, ysize);
/********************************VIDEO*******************************
**************/
    /* set the initial camera parameters */
    if( arParamLoad(cparam_name, 1, &wparam) < 0 ) {
        printf("Camera parameter load error !!\n");
        exit(0);
    }

    arParamChangeSize( &wparam, xsize, ysize, &cparam );
    arInitCparam( &cparam );
    printf("*** Camera Parameter ***\n");
    arParamDisp( &cparam );

    if( (patt_id=arLoadPatt(patt_name)) < 0 ) {
        printf("pattern load error !!\n");
        exit(0);
    }

    /* open the graphics window */
    argInit( &cparam, 1.0, 0, 0, 0, 0 );
}

/* cleanup function called when program exits */
static void cleanup(void)
{
    arVideoCapStop();
    arVideoClose();
    argCleanup();
    //clean MyKinect
    g_MyKinect.Exit();
}

static void draw( double trans[3][4] )
{
    double    gl_para[16];
    GLfloat   mat_ambient[]     = {0.0, 0.0, 1.0, 1.0};
    GLfloat   mat_flash[]       = {0.0, 0.0, 1.0, 1.0};
    GLfloat   mat_flash_shiny[] = {50.0};
    GLfloat   light_position[]  = {100.0,-200.0,200.0,0.0};
    GLfloat   ambi[]            = {0.1, 0.1, 0.1, 0.1};
    GLfloat   lightZeroColor[]  = {0.9, 0.9, 0.9, 0.1};

    argDrawMode3D();
    argDraw3dCamera( 0, 0 );
    glClearDepth( 1.0 );
    glClear(GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);

    /* load the camera transformation matrix */
    argConvGlpara(trans, gl_para);
      //printf("%.2f  %.2f  %.2f\n", trans[0][3], trans[1][3],
trans[2][3]);
    glMatrixMode(GL_MODELVIEW);
    glLoadMatrixd( gl_para );

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambi);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMatrixMode(GL_MODELVIEW);
    glTranslatef( 0.0, 0.0, size/2.0f );
    glutSolidCube(size);
    glDisable( GL_LIGHTING );

    glDisable( GL_DEPTH_TEST );
}
```

```
#include "MyKinect.h"

//----------------------------------------------------------------------
------------------------------
//init
//----------------------------------------------------------------------
------------------------------
int MyKinect::Init()
{
      XnStatus nRetVal = XN_STATUS_OK;
      xn::EnumerationErrors errors;
      nRetVal = context.InitFromXmlFile(SAMPLE_XML_PATH, &errors);
      if (nRetVal == XN_STATUS_NO_NODE_PRESENT)
      {
            XnChar strError[1024];
            errors.ToString(strError, 1024);
            printf("%s\n", strError);
            return nRetVal;
      }
      else if (nRetVal != XN_STATUS_OK)
      {
            printf("Open failed: %s\n", xnGetStatusString(nRetVal));
            return nRetVal;
      }

      //config depth generator
      nRetVal = context.FindExistingNode(XN_NODE_TYPE_DEPTH, depthGen);
      CHECK_RC(nRetVal, "Find depth generator");

      //config image generator
      nRetVal = context.FindExistingNode(XN_NODE_TYPE_IMAGE, imageGen);
      CHECK_RC(nRetVal, "Find image generator");

      depthGen.GetMetaData(depthMD);
      imageGen.GetMetaData(imageMD);

       // Hybrid mode isn't supported in this sample
      if (imageMD.FullXRes() != depthMD.FullXRes() || imageMD.FullYRes()
!= depthMD.FullYRes())
      {
            printf ("The device depth and image resolution must be
equal!\n");
            return -1;
      }

      // RGB is the only image format supported.
      if (imageMD.PixelFormat() != XN_PIXEL_FORMAT_RGB24)
      {
            printf("The device image format must be RGB24\n");
            return -1;
      }

#ifdef USE_USERDETECTOR
      //config user generator
      nRetVal = context.FindExistingNode(XN_NODE_TYPE_USER, userGen);
      if (nRetVal != XN_STATUS_OK)
      {
            nRetVal = userGen.Create(context);
            CHECK_RC(nRetVal, "Find user generator");
      }
      //init userDetector
      userDetector = new UserDetector(&userGen, &userStatus);
#endif

      depthGen.GetAlternativeViewPointCap().SetViewPoint(imageGen);

#ifndef USE_MIRROR
      context.SetGlobalMirror(FALSE);
#endif

      //context.StartGeneratingAll();
      return 1;
}

//------------------------------------------------------------------------
------------------------------
//
//------------------------------------------------------------------------
------------------------------
void MyKinect::StartGeneratingAll()
{
      context.StartGeneratingAll();
}

//------------------------------------------------------------------------
------------------------------
//
//------------------------------------------------------------------------
------------------------------
void MyKinect::Update()
{
      context.WaitAnyUpdateAll();
}

//------------------------------------------------------------------------
------------------------------
//
//------------------------------------------------------------------------
------------------------------
void MyKinect::Exit()
{
#ifdef USE_USERDETECTOR
      delete userDetector;
#endif
      context.Shutdown();
}

//------------------------------------------------------------------------
------------------------------
//
//------------------------------------------------------------------------
------------------------------
unsigned char * MyKinect::GetBGRA32Image()
{
      depthGen.GetMetaData(depthMD);
      imageGen.GetMetaData(imageMD);

      const XnDepthPixel* pDepth = depthMD.Data();
      const XnUInt8* pImage = imageMD.Data();
```

```
	const XnRGB24Pixel* pImageRow = imageMD.RGB24Data();
	unsigned char* pBuffRow = imageBGRA32Buf + imageMD.YOffset();

	for (XnUInt y = 0; y < imageMD.YRes(); ++y)
	{
		const XnRGB24Pixel* pImage = pImageRow;
		unsigned char* pBuff = pBuffRow + imageMD.XOffset();

		for (XnUInt x = 0; x < imageMD.XRes(); ++x, ++pImage)
		{
		  pBuff[0] = pImage->nBlue;
		  pBuff[1] = pImage->nGreen;
		  pBuff[2] = pImage->nRed;
		  pBuff[3] = 0;
		  pBuff +=4;
		}
		pImageRow += imageMD.XRes();
		pBuffRow += imageMD.XRes()*4;
	}

	return imageBGRA32Buf;
}

//-----------------------------------------------------------------------
-----------------------------
//
//-----------------------------------------------------------------------
-----------------------------
unsigned char * MyKinect::GetDepthDrewByColor()
{
	depthGen.GetMetaData(depthMD);
	const XnDepthPixel* pDepth = depthMD.Data();

	// Calculate the accumulative histogram (the yellow display...)
	xnOSMemSet(depthHist, 0, MAX_DEPTH*sizeof(float));

	unsigned int nNumberOfPoints = 0;
	for (XnUInt y = 0; y < depthMD.YRes(); ++y)
	{
		for (XnUInt x = 0; x < depthMD.XRes(); ++x, ++pDepth)
		{
			if (*pDepth != 0)
			{
				depthHist[*pDepth]++;
				nNumberOfPoints++;
			}
		}
	}

	for (int nIndex=1; nIndex< MAX_DEPTH; nIndex++)
	{
		depthHist[nIndex] += depthHist[nIndex-1];
	}

	if (nNumberOfPoints)
	{
		for (int nIndex=1; nIndex< MAX_DEPTH; nIndex++)
		{
```

```
            depthHist[nIndex] = (unsigned int)(256 * (1.0f - 
(depthHist[nIndex] / nNumberOfPoints)));
        }
    }

    xnOSMemSet(depthDrewByColorBuf, 0, sizeof(depthDrewByColorBuf));

    const XnDepthPixel* pDepthRow = depthMD.Data();
    unsigned char* pBuffRow = depthDrewByColorBuf + depthMD.YOffset();

    for (XnUInt y = 0; y < depthMD.YRes(); ++y)
    {
        const XnDepthPixel* pDepth = pDepthRow;
        unsigned char* pBuff = pBuffRow + depthMD.XOffset();

        for (XnUInt x = 0; x < depthMD.XRes(); ++x, ++pDepth)
        {
            if (*pDepth != 0)
            {
                int nHistValue = depthHist[*pDepth];
                pBuff[2] = nHistValue;
                pBuff[1] = nHistValue;
                pBuff[0] = 0;
                pBuff[3] = 0;
             }
             pBuff += 4;
        }
        pDepthRow += depthMD.XRes();
        pBuffRow += depthMD.FullXRes()*4;
    }

    return depthDrewByColorBuf;
}

//------------------------------------------------------------------------
------------------------------
//
//------------------------------------------------------------------------
------------------------------
unsigned char * MyKinect::GetDepthMixedImage()
{
    xnOSMemSet(depthMixedImageBuf, 0, sizeof(depthMixedImageBuf));

    depthGen.GetMetaData(depthMD);
    const XnDepthPixel* pDepth = depthMD.Data();
    const XnDepthPixel* pDepthRow = depthMD.Data();
    unsigned char* pBuffRow = depthMixedImageBuf + depthMD.YOffset();
    unsigned char* pImgRow = imageBGRA32Buf + depthMD.YOffset();

    for (XnUInt y = 0; y < depthMD.YRes(); ++y)
    {
        const XnDepthPixel* pDepth = pDepthRow;
        unsigned char* pBuff = pBuffRow + depthMD.XOffset();
        unsigned char* pImg = pImgRow + depthMD.XOffset();

        for (XnUInt x = 0; x < depthMD.XRes(); ++x, ++pDepth)
        {
            if (*pDepth != 0)
            {
                pBuff[2] = pImg[2];
```

```
                        pBuff[1] = pImg[1];
                        pBuff[0] = pImg[0];
                        pBuff[3] = 0;
                  }
                  pBuff += 4;
                  pImg += 4;
            }

            pDepthRow += depthMD.XRes();
            pBuffRow += depthMD.FullXRes()*4;
            pImgRow += depthMD.FullXRes()*4;
      }

      return depthMixedImageBuf;
}

//---------------------------------------------------------------------------
------------------------------
//
//---------------------------------------------------------------------------
------------------------------
void MyKinect::GetImageSize(int *xSize, int *ySize)
{
      *xSize = (int)depthMD.FullXRes();
      *ySize = (int)depthMD.FullYRes();
}
```

```
#ifndef _MYKINECT_H_
#define _MYKINECT_H_

#include "Common.h"

#ifdef USE_USERDETECTOR
      #include "UserDetector.h"
      #include "PlayerStatus.h"
#endif

class MyKinect
{
private:
      unsigned char   imageBGRA32Buf[640*480*4];              //BGRA
      unsigned char   depthDrewByColorBuf[640*480*4];         //BGRA
      unsigned char   depthMixedImageBuf[640*480*4];          //BGRA
      float           depthHist[MAX_DEPTH];

public:
      //OpenNI
      Context               context;

#ifdef USE_USERDETECTOR
      UserGenerator   userGen;
      PlayerStatus    userStatus;
      UserDetector*   userDetector;
#endif

      DepthGenerator  depthGen;
      ImageGenerator  imageGen;
      DepthMetaData   depthMD;
      ImageMetaData   imageMD;

public:
      MyKinect(){}
      ~MyKinect(){}

      int Init();
      void StartGeneratingAll();
      void Update();
      void Exit();

      unsigned char * GetBGRA32Image();
      unsigned char * GetDepthDrewByColor();
      unsigned char * GetDepthMixedImage();
      void GetImageSize(int *xSize, int *ySize);
};

#endif
```

```
#ifndef _MYVECTOR_H_
#define _MYVECTOR_H_

#include "Common.h"
#include "Math.h"

class XV3 : public XnVector3D {
public:
        // ctors
        XV3() { X = Y = Z = 0; }
        XV3(float x, float y, float z) { X=x, Y=y, Z=z; }
        XV3(float* v) { X=v[0], Y=v[1], Z=v[2]; }
        XV3(const XnVector3D& v) { X=v.X, Y=v.Y, Z=v.Z; }

        // object lifecycle

        XV3& assign(float x, float y, float z) { X=x, Y=y, Z=z; return
*this; }
        XV3& assign(float* v) { X=v[0], Y=v[1], Z=v[2]; return *this; }
        XV3& assign(const XnVector3D& v) { X=v.X, Y=v.Y, Z=v.Z; return
*this; }
        XV3& operator=(const XnVector3D& v) { assign(v); return *this; }

        // add, sub, mul, div

        XV3& operator+=(const XnVector3D& v) { X+=v.X, Y+=v.Y, Z+=v.Z;
return *this; }
        XV3 operator+(const XnVector3D& v) const { return XV3(X+v.X,
Y+v.Y, Z+v.Z); }

        XV3& operator-=(const XnVector3D& v) { X-=v.X, Y-=v.Y, Z-=v.Z;
return *this; }
        XV3 operator-(const XnVector3D& v) const { return XV3(X-v.X, Y-
v.Y, Z-v.Z); }

        XV3& operator*=(float a) { X*=a, Y*=a, Z*=a; return *this; }
        XV3 operator*(float a) const { return XV3(X*a, Y*a, Z*a); }

        XV3& operator/=(float a) { X/=a, Y/=a, Z/=a; return *this; }
        XV3 operator/(float a) const { return XV3(X/a, Y/a, Z/a); }

        XV3 operator-() const { return XV3(-X, -Y, -Z); }

        // products

        float dot(const XnVector3D& v) const { return X*v.X + Y*v.Y +
Z*v.Z; }

        float dotNormalized(const XV3& v) const { return dot(v) /
magnitude() / v.magnitude(); }

        XV3& crossM(const XnVector3D& v) { assign(Y*v.Z-Z*v.Y, Z*v.X-
X*v.Z, X*v.Y-Y*v.X); return *this; }
        XV3 cross(const XnVector3D& v) const { return XV3(Y*v.Z-Z*v.Y,
Z*v.X-X*v.Z, X*v.Y-Y*v.X); }

        // magnitudes

        /** squared magnitude */
        float magnitude2() const { return X*X + Y*Y + Z*Z; }
        float magnitude() const { return sqrt(magnitude2()); }

        /** squared distance */
        float distance2(const XnVector3D& v) const { return (*this -
v).magnitude2(); }

        float distance(const XnVector3D& v) const { return
sqrt(distance2(v)); }

        // normalizations

        XV3& normalizeM()
        {
                float m = magnitude();
                return assign(X/m, Y/m, Z/m);
        }

        XV3 normalize() const
        {
                float m = magnitude();
                return XV3(X/m, Y/m, Z/m);
        }

        // other derivables

        XV3& interpolateM(const XnVector3D& v, float alpha = 0.5f)
        {
                return assign(::interpolate(X, v.X, alpha),
::interpolate(Y, v.Y, alpha), ::interpolate(Z, v.Z, alpha));
        }

        XV3 interpolate(const XnVector3D& v, float alpha = 0.5f) const
        {
                return XV3(::interpolate(X, v.X, alpha), ::interpolate(Y,
v.Y, alpha), ::interpolate(Z, v.Z, alpha));
        }
};

#endif
```

```
#ifndef _MATH_H_
#define _MATH_H_

#include "Common.h"

#define _USE_MATH_DEFINES
#include <math.h>

inline float interpolate(float x1, float x2, float alpha = 0.5f)
{
        return x1 + (x2 - x1) * alpha;
}

inline int square(int i)
{
        return i * i;
}

inline float square(float f)
{
        return f * f;
}

inline float DEG_TO_RADIAN(float deg)
{
      return (float)(deg*M_PI/180.0f);
}
#endif
```

```
#ifndef _COMMON_H_
#define _COMMON_H_

#include <XnOpenNI.h>
#include <XnCodecIDs.h>
#include <XnCppWrapper.h>

using namespace xn;

//#define USE_USERDETECTOR
//#define USE_MIRROR

#define MAX_DEPTH 10000
#define SAMPLE_XML_PATH "Data/SamplesConfig.xml"

#define CHECK_RC(nRetVal, what)
           \
      if (nRetVal != XN_STATUS_OK)
      \
      {
                      \
           printf("%s failed: %s\n", what, xnGetStatusString(nRetVal));\
           return nRetVal;
                \
      }


#endif
```

```
#ifndef _UTIL_H_
#define _UTIL_H_

#include "Common.h"

const float CONFIDENCE_THRESHOLD = 0.5f;
#ifndef USE_MACRO
inline bool isConfident(XnSkeletonJointPosition jointPos) {
        return jointPos.fConfidence >= CONFIDENCE_THRESHOLD;
}

inline bool isConfident(XnSkeletonJointOrientation jointOrientation)
{
        return jointOrientation.fConfidence >= CONFIDENCE_THRESHOLD;
}
#else
#define isConfident(jointPos) ((jointPos).fConfidence >=
CONFIDENCE_THRESHOLD)
#endif

#ifndef USE_MACRO
inline bool i2b(int i) { return !!i; }
#else
#define i2b(i) (!!(i))
#endif

#ifndef USE_MACRO
inline float b2fNormalized(unsigned char b) { return b/255.0f; }
#else
#define b2fNormalized(b) ((b)/255.0f)
#endif

#endif
```

Appendix D

Machine learning code for marker′s position estimation

The solution to get the marker′s position using the proposed approach is written in C Sharp. The solution project is composed of a main.cs file.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathNet.Numerics;
using MathNet.Numerics.LinearRegression;
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.LinearAlgebra.Double;
using Excel = Microsoft.Office.Interop.Excel;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;
using System.IO;




namespace multipleregression
{
    class Program
    {

        static void Main(string[] args)
        {

            String[] gz = Column(3);
            String[] rx = Column(4);
            String[] ry = Column(5);
            String[] rz = Column(6);
            String[] lux = Column(7);

            Double[] dgz = Array.ConvertAll(gz,Double.Parse);
            Double[] drx = Array.ConvertAll(rx, Double.Parse);
            Double[] dry = Array.ConvertAll(ry, Double.Parse);
            Double[] drz = Array.ConvertAll(rz, Double.Parse);
            Double[] dlux = Array.ConvertAll(lux, Double.Parse);


            var X = DenseMatrix.OfColumns(new Vector<double>[] {
DenseVector.OfArray(drx), DenseVector.OfArray(dry),
DenseVector.OfArray(drz), DenseVector.OfArray(dlux) });
            var y = DenseVector.OfArray(dgz);

            var betha = MultipleRegression.QR(X, y);

            double[][] xdata = new double[][] { drx,dry,drz,dlux};
            double[] ydata = dgz;

            Console.WriteLine(betha);
            var R2 = GoodnessOfFit.RSquared(xdata.Select(x=>
betha[0]+betha[1]*x[0]+betha[2]*x[1]+betha[3]*x[2]+betha[4]*x[3]),ydata);
            Console.WriteLine(R2.ToString());


        }

        public static string[] Column(int col)
        {
            Microsoft.Office.Interop.Excel.Application xlsApp = new
Microsoft.Office.Interop.Excel.Application();

            if (xlsApp == null)
            {
                Console.WriteLine("EXCEL could not be started. Check that
your office installation and project references are correct.");
                return null;
            }

            //Displays Excel so you can see what is happening
            //xlsApp.Visible = true;

            Excel.Workbook wb =
xlsApp.Workbooks.Open("C:\\Users\\Cristy\\Desktop\\Kinect-ARToolKit-
master\\Samples\\KinectOnly\\Sample1\\Sample1\\test.xlsx",
                0, true, 5, "", "", true, Excel.XlPlatform.xlWindows,
"\t", false, false, 0, true);
            Excel.Sheets sheets = wb.Worksheets;
            Excel.Worksheet ws = (Excel.Worksheet)sheets.get_Item(1);

            Excel.Range firstColumn = ws.UsedRange.Columns[col];
            System.Array myvalues =
(System.Array)firstColumn.Cells.Value;
            string[] strArray = myvalues.OfType<object>().Select(o =>
o.ToString()).ToArray();
            return strArray;
        }
    }
}


            /*
            ReadExcel readerObj = new ReadExcel();

            // "C:\\Users\\Cristy\\Documents\\Visual Studio
2015\\Projects\\multipleregression\\multipleregression\\bin\\Debug\\Book1
.xls"

            OleDbConnection excelConnection = null;
            OleDbDataAdapter adapter = null;

            OleDbConnection cn = new OleDbConnection();
            DataTable schemaTable;


            cn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Users\\Cristy\\Documents\\Visual Studio
2015\\Projects\\multipleregression\\multipleregression\\bin\\Debug\\Book1
.xlsx;Extended Properties='Excel 8.0;HDR=YES;'";
```

```
            cn.Open();

            schemaTable = 
cn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,new Object[] { null, 
null,"ID","TABLE"});

            for (int i = 0; i < schemaTable.Columns.Count; i++) {

Console.WriteLine(schemaTable.Columns[i].ToString()+":"+schemaTable.Rows[
0][1].ToString());
            }
            cn.Close();
            Console.ReadLine();

            try
            {
                excelConnection = new OleDbConnection();
                excelConnection.ConnectionString = 
"Provider=Microsoft.ACE.OLEDB.12.0;Data 
Source=C:\\Users\\Cristy\\Documents\\Visual Studio 
2015\\Projects\\multipleregression\\multipleregression\\bin\\Debug\\Book1
.xlsx;Extended Properties='Excel 8.0;HDR=YES;'";
                excelConnection.Open();
                DataTable dtTables = new DataTable();

                //to get the schema of the workbook.
                dtTables = excelConnection.GetSchema();



                //get the tables in the workbook
                dtTables = 
excelConnection.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, null);

                for (int i = 0; i < dtTables.Columns.Count; i++)
                {

Console.WriteLine("Columns"+dtTables.Columns[i].ToString());
                }

                for (int i = 0; i < dtTables.Rows.Count; i++)
                {

Console.WriteLine("Rows"+dtTables.Rows[i].ItemArray[2].ToString());
                   //
Console.WriteLine(dtTables.Rows[i]["ID"].ToString());
                    Console.ReadLine();
                }



                String[] excelSheets = null;
                if ((dtTables != null))
                {
                    excelSheets = new String[dtTables.Rows.Count];
                    int i = 0;



                    // Add the sheet name to the string array.
                    foreach (DataRow row in dtTables.Rows)
                    {
                        excelSheets[i] = row["TABLE_NAME"].ToString();

                        Console.WriteLine(excelSheets[i]);
                        i++;
                    }
                }
                DataSet ds = new DataSet();

                //prepare dataset from the tables in the workbook
                foreach (string sheet in excelSheets)
                {
                    OleDbCommand cmd = new OleDbCommand();
                    cmd.Connection = excelConnection;
                    cmd.CommandText = "Select * from [" + sheet + "]";
                    DataTable dtItems = new DataTable();
                    dtItems.TableName = sheet;

                    adapter = new OleDbDataAdapter();
                    adapter.SelectCommand = cmd;

                    // adapter.FillSchema(ds
                    adapter.Fill(dtItems);
                    ds.Tables.Add(dtItems);
                }
            }

            finally
            {
                adapter.Dispose();
                excelConnection.Dispose();
            }


        }


        public partial class ReadExcel : System.Web.UI.Page
        {
            public ReadExcel()
            {
                Page_Load();
            }
            protected void Page_Load()
            {
                ImportExcel("C:\\Users\\Cristy\\Documents\\Visual Studio
2015\\Projects\\multipleregression\\multipleregression\\bin\\Debug\\Book1
.xls");
                ImportExcel2007("C:\\Users\\Cristy\\Documents\\Visual
Studio
2015\\Projects\\multipleregression\\multipleregression\\bin\\Debug\\Book1
.xlsxc");
            }

            //Read To Excel 97-2003 File
            private Boolean ImportExcel(String strFilePath)
            {
                if (!File.Exists(strFilePath)) return false;
                String strExcelConn = "Provider=Microsoft.Jet.OLEDB.4.0;"
                + "Data Source=" + strFilePath + ";"
                + "Extended Properties='Excel 8.0;HDR=Yes'";
                OleDbConnection connExcel = new
OleDbConnection(strExcelConn);
                OleDbCommand cmdExcel = new OleDbCommand();
                try
                {
                    cmdExcel.Connection = connExcel;

                    //Check if the Sheet Exists
                    connExcel.Open();
                    DataTable dtExcelSchema;
                    //Get the Schema of the WorkBook
                    dtExcelSchema =
connExcel.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, null);
                    connExcel.Close();

                    //Read Data from Sheet1
                    connExcel.Open();
                    OleDbDataAdapter da = new OleDbDataAdapter();
                    DataSet ds = new DataSet();
                    string SheetName =
dtExcelSchema.Rows[0]["TABLE_NAME"].ToString();
                    cmdExcel.CommandText = "SELECT * From [" + SheetName
+ "]";
                    //Range Query
                    //cmdExcel.CommandText = "SELECT * From [" +
SheetName + "A3:B5]";

                    Console.WriteLine();

                    da.SelectCommand = cmdExcel;
                    da.Fill(ds);
                    connExcel.Close();
                    return true;
                }
                catch
                {
                    return false;
                }
                finally
                {
                    cmdExcel.Dispose();
                    connExcel.Dispose();
                }
            }
            //Read To Excel 97-2007 File
            private Boolean ImportExcel2007(String strFilePath)
            {
                if (!File.Exists(strFilePath)) return false;
                String strExcelConn =
"Provider=Microsoft.ACE.OLEDB.12.0;"
                + "Data Source=" + strFilePath + ";"
                + "Extended Properties='Excel 8.0;HDR=No'";
                OleDbConnection connExcel = new
OleDbConnection(strExcelConn);
                OleDbCommand cmdExcel = new OleDbCommand();
                try
                {
                    cmdExcel.Connection = connExcel;
                    //Check if the Sheet Exists
                    connExcel.Open();
                    DataTable dtExcelSchema;
                    //Get the Schema of the WorkBook

    dtExcelSchema = connExcel.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,
null);
                    connExcel.Close();

                    //Read Data from Sheet1
                    connExcel.Open();
                    OleDbDataAdapter da = new OleDbDataAdapter();
                    DataSet ds = new DataSet();
                    string SheetName =
dtExcelSchema.Rows[0]["TABLE_NAME"].ToString();
                    cmdExcel.CommandText = "SELECT * From [" + SheetName
+ "]";

                    //Range Query
                    //cmdExcel.CommandText = "SELECT * From [" +
SheetName + "A3:B5]";

                    da.SelectCommand = cmdExcel;
                    da.Fill(ds);
                    connExcel.Close();
                    return true;
                }
                catch
                {
                    return false;
                }
                finally
                {
                    cmdExcel.Dispose();
                    connExcel.Dispose();
                }

            }
        }

    }
}
    */
```

Appendix E

Marker′s position estimations and light measurements

The samples were taken using two different cameras with different resolution and a kinect. The data taken are the following:

1. Marker′s position estimation samples using a Logitech Camera
2. Marker′s position estimation with three different approaches using a Logitech Camera
3. Marker′s position estimation sample using a Kinect
4. Marker′s position estimation with three different approaches using a Kinect

Marker position in z-direction(mm)	Sample 1	Sample 2	Sample 3	Light measurement on sample 1	Light measurement on sample 2	Light measurement on sample 3
220	n/a	n/a	15.0111	n/a	n/a	235
230	n/a	n/a	14.35089	n/a	n/a	239
240	n/a	23.5996	16.1964	n/a	279	231
250	n/a	25.0029	16.18826	n/a	241	256
260	n/a	23.5033	17.64176	n/a	267	251
270	n/a	24.6639	15.73986	n/a	256	240
280	n/a	27.3303	19.21728	n/a	253	239
290	n/a	26.0913	19.41742	n/a	263	234
300	n/a	29.375	21.11121	n/a	224	240
310	n/a	29.9891	19.23355	n/a	236	233
320	n/a	26.3706	19.14162	n/a	238	222
330	n/a	26.9443	21.28199	n/a	245	220
340	n/a	34.1309	21.43634	n/a	214	220
350	n/a	30.7957	22.64834	n/a	218	205
360	n/a	31.3784	23.50922	n/a	224	225
370	n/a	28.6336	15.43759	n/a	255	227
380	n/a	32.9572	21.94681	n/a	207	221
390	n/a	29.3429	28.01275	n/a	206	215
400	n/a	33.9061	23.34109	n/a	216	229
410	n/a	33.7833	29.15536	n/a	232	210
420	n/a	39.3724	30.00048	n/a	203	194
430	n/a	34.8804	23.05413	n/a	228	185
440	n/a	47.0189	29.55238	n/a	213	201
450	n/a	32.6214	28.49078	n/a	224	168
460	n/a	33.0015	29.93659	n/a	220	168
470	n/a	40.9683	32.88354	n/a	211	178
480	n/a	28.6755	32.50513	n/a	207	170
490	n/a	33.5173	32.40691	n/a	203	157
500	n/a	41.8524	35.47276	n/a	195	166
510	n/a	31.7933	30.76759	n/a	194	166
520	40.70677	43.5899	29.01133	270	210	177
530	40.32662	38.1273	33.1881	274	186	166
540	41.20622	21.5459	3.303736	84	183	155
550	45.68391	32.1548	3.627462	94	155	160
560	43.65714	29.9311	21.87836	90	203	150
570	50.40752	35.2754	20.31003	94	195	146
580	39.11388	36.5422	6.568803	303	205	163
590	52.30216	36.0124	3.815341	87	168	140
600	59.24441	45.4997	3.969592	82	163	135

Marker position in z-direction(mm)	Sample 1	Sample 2	Sample 3	Light measurement on sample 1	Light measurement on sample 2	Light measurement on sample 3
610	34.3214	33.1234	8.791145	97	182	135
620	47.24361	36.8222	4.270113	82	156	116
630	55.22807	44.0509	n/a	87	132	n/a
640	54.10155	28.9228	n/a	78	124	n/a
650	35.76732	n/a	n/a	89	n/a	n/a
660	52.5987	n/a	n/a	82	n/a	n/a
670	29.59228	n/a	n/a	84	n/a	n/a
680	48.88712	n/a	n/a	79	n/a	n/a
690	29.07539	n/a	n/a	82	n/a	n/a
700	40.53143	n/a	n/a	84	n/a	n/a
710	23.58767	n/a	n/a	285	n/a	n/a
720	50.17313	n/a	n/a	76	n/a	n/a
730	26.28451	n/a	n/a	235	n/a	n/a
740	22.94787	n/a	n/a	70	n/a	n/a
750	18.97103	n/a	n/a	240	n/a	n/a
760	12.45333	n/a	n/a	70	n/a	n/a
770	-9.22997	n/a	n/a	270	n/a	n/a
780	9.733734	n/a	n/a	69	n/a	n/a
790	-32.5058	n/a	n/a	237	n/a	n/a
800	-4.62657	n/a	n/a	67	n/a	n/a
810	-29.9479	n/a	n/a	233	n/a	n/a
820	-30.5333	n/a	n/a	73	n/a	n/a
830	-44.0617	n/a	n/a	273	n/a	n/a
840	-27.887	n/a	n/a	227	n/a	n/a
850	-13.0692	n/a	n/a	64	n/a	n/a
860	-20.9957	n/a	n/a	204	n/a	n/a
870	-27.136	n/a	n/a	64	n/a	n/a
880	-23.1735	n/a	n/a	204	n/a	n/a
890	-28.844	n/a	n/a	66	n/a	n/a
900	-54.8585	n/a	n/a	213	n/a	n/a
910	-50.902	n/a	n/a	66	n/a	n/a
920	-60.0326	n/a	n/a	203	n/a	n/a
930	-58.8562	n/a	n/a	67	n/a	n/a
940	-67.5129	n/a	n/a	216	n/a	n/a
950	-69.1366	n/a	n/a	60	n/a	n/a
960	-125.741	n/a	n/a	193	n/a	n/a
970	-95.4772	n/a	n/a	151	n/a	n/a
980	-85.0467	n/a	n/a	151	n/a	n/a
990	-81.7366	n/a	n/a	137	n/a	n/a

Marker position in z-direction(mm)	Sample 1	Sample 2	Sample 3	Light measurem-ent on sample 1	Light measurem-ent on sample 2	Light measurem-ent on sample 3
1000	-74.9101	n/a	n/a	153	n/a	n/a
1010	-69.8427	n/a	n/a	144	n/a	n/a
1020	-62.3518	n/a	n/a	166	n/a	n/a
1030	-79.4263	n/a	n/a	146	n/a	n/a
1040	-82.0197	n/a	n/a	41	n/a	n/a
1050	-91.6475	n/a	n/a	156	n/a	n/a
1060	-79.9165	n/a	n/a	145	n/a	n/a

	Position detection error(mm)		
Marker position in z-direction	Proposed method	Simple linear regression	ARToolKit approach
220	0.168981333	-0.403286201	15.01109975
230	-0.146408264	-0.117970635	14.35089175
240	-0.184955137	-1.204366516	16.19640375
250	0.794057786	1.28895276	16.1882615
260	0.036329615	-0.945080279	17.641756
270	-0.58827236	0.142668583	15.73986
280	0.451188545	-0.42385778	19.217282
290	1.253588298	1.424298332	19.41741775
300	0.612355066	-0.829802001	21.11121
310	0.526554001	1.149252804	19.23355325
320	-0.609664664	-0.25114929	19.141621
330	-0.714199638	-1.826783959	21.281994
340	-0.350137118	-0.746660368	21.4363445
350	-2.399032181	-5.134853462	22.64834425
360	1.39019869	1.45367399	23.509224
370	-1.460486261	3.493992688	15.437591
380	-0.00338742	1.121193036	21.9468145
390	-0.86552779	-5.180399637	28.012749
400	-1.534848371	-2.168249947	23.34109475
410	0.974826034	-1.585811487	29.15536
420	2.452478828	1.144258924	30.00047925
430	0.35257623	3.791239936	23.054126
440	-0.156215221	-2.586841789	29.552377
450	-1.515632226	3.035957835	28.490775
460	-0.226262638	3.461408932	29.93658875
470	-2.469974152	1.980452697	32.88354025
480	0.718992163	3.500394119	32.505126
490	-0.869773894	-0.812921153	32.4069095
500	4.365994169	5.44118671	35.472763
510	6.782369717	7.270002222	30.76759275
520	-2.188456667	1.1808721	29.01133175
530	-1.650310661	-0.684342872	33.188101
540	1.018790329	1.186943452	3.3037365
550	-1.97779335	-1.385061322	3.6274625
560	-1.166098472	-3.63077959	21.87835825
570	2.027245466	-1.700224173	20.31002625
580	-0.261186801	-0.901127054	6.56880275
590	0.116244677	-4.677004604	3.815341
600	-4.465523756	-9.185376563	3.96959175
610	1.031379265	3.930552176	8.791145
620	0.729996831	0.384649386	4.27011325

REAL X	REAL Y	REAL Z	X	Y	Z	LUX
0	0	260	-43.840878	23.0565944	367.160763	238
0	0	270	-43.894012	23.3144769	382.306179	237
0	0	280	-44.17854	23.4678216	399.593157	232
0	0	290	-43.581208	23.422925	409.30541	225
0	0	300	-44.078577	23.7655017	428.728294	229
0	0	310	-44.358826	24.1121366	447.136147	202
0	0	320	-44.529291	24.241763	462.976867	221
0	0	330	-44.002088	24.1864164	472.638536	230
0	0	340	-44.137876	24.3623766	489.271979	235
0	0	350	-44.301503	24.6864183	506.200059	234
0	0	360	-44.114543	24.6460688	519.440789	218
0	0	370	-43.906858	24.3613373	533.774565	220
0	0	380	-44.593209	25.0809339	553.922792	216
0	0	390	-44.417708	25.0972793	568.038123	217
0	0	400	-44.204287	25.1627831	578.501239	222
0	0	410	-44.040086	25.3328317	593.103707	216
0	0	420	-43.711545	25.3931632	605.931612	218
0	0	430	-44.60539	25.7478512	631.976291	249
0	0	440	-44.164647	25.5475602	642.345935	224
0	0	450	-44.708618	26.1547278	662.742176	236
0	0	460	-44.358131	25.8731287	673.081575	231
0	0	470	-44.844406	26.402026	692.949312	222
0	0	480	-44.657471	26.2644332	707.650991	205
0	0	490	-44.798378	26.8464246	725.656811	227
0	0	500	-45.147546	27.118082	746.397804	222
0	0	510	-45.356992	27.2826375	760.886326	213
0	0	520	-44.605388	27.2397107	768.859265	237
0	0	530	-45.218802	27.5096579	793.435321	192
0	0	540	-39.769783	23.8974641	810.911381	175
0	0	550	-38.859464	23.4121206	815.195268	199
0	0	560	-39.313024	23.9788091	839.374853	200
0	0	570	-38.498372	23.7074716	842.203122	175
0	0	580	-39.052541	24.0475115	868.455434	161
0	0	590	-36.201736	22.581818	828.038779	180
0	0	600	-36.152184	22.7527335	842.220296	179
0	0	610	-36.412076	22.823994	860.101176	184
0	0	620	-37.699918	23.6692139	914.813577	194
0	0	630	-37.88221	24.2248813	933.551041	130
0	0	640	-35.541779	22.8973077	891.350872	168
0	0	650	-34.586158	22.6713506	896.800142	183
0	0	660	-37.80087	24.5907794	1000.13437	187

	Position detection error(mm)		
Marker position in z-direction	Proposed method	Simple linear regression	ARToolKit approach
260	0.918252062	2.747339583	107.1607626
270	1.461686078	5.288696451	112.3061791
280	1.054869972	-0.004167818	119.5931569
290	-0.462591728	5.111409532	119.3054105
300	0.741004574	2.289538065	128.7282945
310	2.835699638	2.579290457	137.1361471
320	1.860358156	-0.337025148	142.9768668
330	-0.282713285	3.626310829	142.6385364
340	-0.110821909	2.578663052	149.271979
350	1.405733328	5.419083203	156.2000587
360	-0.283388465	2.45325001	159.4407886
370	-3.495250015	-6.521629011	163.774565
380	0.409236183	-1.732780757	173.9227922
390	-0.478434474	-2.036251285	178.0381232
400	-2.827975577	-1.111057835	178.5012385
410	-2.113838298	3.042652506	183.1037074
420	-2.506730185	7.096089392	185.9316118
430	0.159248656	-0.416453849	201.9762913
440	-3.075070393	-4.291513408	202.3459353
450	0.665313301	0.951015697	212.7421764
460	-3.940453333	-6.566272379	213.0815747
470	-0.707168893	-3.701403366	222.9493118
480	-2.406784946	-9.245755779	227.6509909
490	1.923209084	3.73340656	235.6568113
500	4.145387682	2.025653845	246.3978042
510	2.695289501	-2.472743106	250.8863264
520	0.838684841	6.944739024	248.8592651
530	-2.406340512	-1.144511299	263.4353211
540	1.726873774	1.361298251	270.9113813
550	0.543665978	-1.046756539	265.1952676
560	2.09389733	5.80930359	279.3748533
570	5.389294908	5.192099561	272.2031223
580	-1.346386862	2.715015725	288.4554336
590	1.269864473	-4.412708241	238.0387791
600	2.068361914	-2.697468011	242.2202956
610	-7.421309709	-9.345948969	250.1011761
620	-6.410103352	-0.457794538	294.8135774
630	-1.06016818	6.99161656	303.5510413
640	-5.795696914	-6.868604656	251.3508724
650	3.186167534	-0.065872663	246.8001423
660	4.310641796	17.74686475	340.1343667

REALX	REALY	REALZ	X	Y	Z	LUX
0	0	250	-13.212406	23.5685584	322.78812	250
0	0	260	-13.176233	24.1656811	334.882	232
0	0	270	-13.088633	24.4559183	346.764059	228
0	0	280	-12.841844	25.3642813	360.920361	238
0	0	290	-12.596118	25.9963314	373.291932	217
0	0	300	-12.593733	26.2985485	388.654452	223
0	0	310	-12.346492	26.9076104	400.612222	235
0	0	320	-12.115846	27.4241366	413.028004	232
0	0	330	-12.04161	27.8601067	426.12914	221
0	0	340	-11.883244	28.5226653	439.627473	200
0	0	350	-11.729891	28.8352062	451.64089	208
0	0	360	-11.608048	29.8684968	465.394521	221
0	0	370	-11.384594	30.2658361	478.408636	206
0	0	380	-11.367928	30.7642696	491.452391	199
0	0	390	-11.056548	31.2986886	505.313771	215
0	0	400	-10.742736	31.6135149	515.654447	203
0	0	410	-10.770684	32.4277889	529.225385	199
0	0	420	-10.72502	33.2719594	546.966379	185
0	0	430	-10.634324	33.8430247	560.037709	178
0	0	440	-10.309235	34.8839423	577.006641	194
0	0	450	-10.140754	35.1573779	589.164353	200
0	0	460	-13.600903	37.7040878	598.175603	188
0	0	470	-13.861614	38.7763859	615.972071	185
0	0	480	-14.037326	39.6865246	630.756107	173
0	0	490	-14.074742	41.0937299	648.239158	165
0	0	500	-13.975764	41.5985651	656.230469	172
0	0	510	-12.44381	39.6542137	657.132098	86
0	0	520	-12.934264	39.0955064	687.487488	98
0	0	530	-13.105588	42.6462035	689.272447	26
0	0	540	-12.671468	39.9739835	687.74345	78
0	0	550	-13.383992	40.6279459	713.846672	79
0	0	560	0.46245098	30.062735	727.723656	96
0	0	570	-6.5082398	30.9788379	738.160467	34
0	0	580	-6.2802514	27.8512903	741.546091	53
0	0	590	-6.3130768	26.1271932	759.543115	32
0	0	600	-5.6835003	25.9139272	783.241039	52
0	0	610	-4.7889825	29.6417868	784.330721	77
0	0	620	-4.1300532	29.8267992	808.195314	62
0	0	630	-3.9220269	26.3209619	824.959257	96
0	0	640	-2.9241181	26.0678683	818.666602	135
0	0	650	-3.1046967	26.8705698	830.611323	110

	Position detection error(mm)		
Marker position in z-direction	Proposed method	Simple linear regression	ARToolKit approach
250	1.490746927	1.354706232	72.78811992
260	0.171026269	0.868192685	74.88200004
270	-0.67992184	-0.06005575	76.76405883
280	-0.595941862	0.163825762	80.92036104
290	0.102758366	-0.014959371	83.29193163
300	0.716735183	1.414593946	88.65445192
310	-0.186409506	0.00958423	90.61222182
320	-0.033597372	-0.599654209	93.02800419
330	-0.15456364	-0.455338717	96.12913985
340	0.579257542	0.203386407	99.62747293
350	-0.166759918	-0.958867513	101.6408898
360	-1.881193613	-1.181880937	105.3945213
370	-0.543644772	-0.961964034	108.4086363
380	1.181643506	-1.000529003	111.4523914
390	-0.417232711	-1.034717942	115.3137712
400	-0.250353314	-2.895743566	115.6544468
410	-1.655304862	-2.728104754	119.2253846
420	-0.061628149	0.891976215	126.9663794
430	-1.173536667	0.858578367	130.0377088
440	1.951969633	3.021590233	137.0066406
450	1.573266163	2.039865459	139.1643532
460	-1.589645679	-2.027035507	138.1756029
470	1.169225896	1.216220392	145.9720711
480	0.713103868	2.48081932	150.7561067
490	1.687437263	5.537870299	158.2391579
500	-2.130153099	1.269534247	156.2304687
510	-1.551862563	-4.814266518	147.1320984
520	11.96932908	8.097561407	167.4874877
530	-0.835787615	0.245004275	159.2724467
540	-9.008608856	-11.41594183	147.7434504
550	0.431242915	-1.870455161	163.8466724
560	3.213893257	3.403322372	167.7236563
570	-0.045388414	1.953753078	168.1604672
580	-5.451858668	-4.916363617	161.5460915
590	-1.919786809	-0.04978423	169.5431151
600	6.493077865	7.584200608	183.2410394
610	-3.714912942	-3.434370717	174.3307213
620	3.394070402	5.170450691	188.1953143
630	8.715056912	8.195094182	194.9592575
640	-4.720587826	-7.480813963	178.6666017
650	-6.989747029	-7.980597886	180.611323

Appendix F

Marker's position estimation functions

The samples taken was used to generate fit and underfit models for the proposed method and the simple linear regression approach. Furthermore, it was used three types of cameras:

1. Logitech C920 with a resolution of 1920 x 1080 RGB @ 30fps
2. Camera web ACTECK CW-760 standard with a resolution 640 x 480 RGB @ 30fps
3. Kinect Xbox 360 with a resolution 1280 x 960 RGB @ 12fps

Marker′s position estimation function		
Camera	Method	Function
Camera Acteck	Simple linear regression	$Zdist = 22.010 + 0.1169X - 0.3371Y + 0.76148Z - 0.02785LUX$
	Proposed method	For Range #1: $Zdist = 28.77 - 0.2211X - 0.4591Y + 0.74884Z + 0.03541LUX$ For Range #2: $Zdist = -3.56 - 1.556X - 2.016Y + 0.8325Z + 0.1002LUX$ For Range #3: $Zdist = 189.02 + 8.460X - 2.511Y + 0.75508Z - 0.04121LUX$
Kinect	Simple linear regression	$Zdist = 356.96 + 22.222X - 31.605Y + 0.37113Z + 0.04709LUX$
	Proposed method	For Range #2 $Zdist = 502.70 + 25.891X + 36.036Y + 0.24152Z + 0.06975LUX$ For Range #3 $Zdist = 97.18 + 5.552X + 8.704Y + 0.56697Z - 0.00732LUX$

Table F.1: Marker′s position estimation functions

Marker′s position estimation function with underfitting		
Camera	Method	Underfitting function
Logitech camera	Simple linear regression	$Zdist = 63.97 + 0.0X + 0.0Y + 0.92849Z - 0.2695LUX$
	Proposed method	For Range #2: $Zdist = -62.5 + 0.0X + 0.0Y + 1.1455Z - 0.2403LUX$ For Range #3: $Zdist = -18.28 + 0.0X + 0.0Y + 0.95227Z + 0.06004LUX$
Acteck camera	Simple linear regression	$Zdist = 37.07 + 1.0691X + 0.0262Y + 0.73752Z - 0.04349LUX$
	Proposed method	For Range #1: $Zdist = 61.47 + 1.303X + 0.078Y + 0.6983Z + 0.04107LUX$ For Range #2: $Zdist = 50.7586 - 2.58014X - 2.22990Y + 0.762931Z + 0LUX$ For Range #3: $Zdist = 228.8 + 8.639X - 4.888Y + 0.84677Z - 0.08781LUX$
Kinect	Simple linear regression	$Zdist = 389.11 + 23.363X + 34.738Y + 0.334Z - 0.1121LUX$
	Proposed method	For Range #2: $Zdist = 734.02 + 6.127X - 18.76Y + 0.6624Z - 0.2322LUX$ For Range #3: $Zdist = 95.44 + 5.919X + 9.861Y + 0.55817Z - 0.03262LUX$

Table F.2: Underfit marker′s position estimation functions

Appendix G

ARToolKit algorithm (to get position data)

Algorithm 5 ARToolKit algorithm (to get position data)

```
Open the file with name trainingData
Read and Save marker file into marker1 variable
Detect camera
if camera is detected then
    loop//for video streaming
        Grab an image of the video stream and Save it into image variable
        //Detect marker and Save the marker information into marker_info variable
        arDetectMarkerLite(dataPtr, thresh, &marker_info,
        &marker_num)
        //ARToolKit function to estimate marker's position
        arGetTransMat(marker_info, target_center, target_width, target_trans)
        //ARToolKit function to get the inverse of the marker's
        //position and orientation.
        arUtilMatInv(target_trans, cam_trans)
        Save x-y-z position into the file trainingData
Close file named trainingData
```

Appendix H

Machine learning algorithm (for training)

Algorithm 6 Machine learning algorithm (for training)

```
Open the file with name trainingDataForLightRange-X for input
Read and Save training data matrix A[i,k] ( which is an $m$ x $n$ matrix with the XYZ positions, the light measurements, and the ground truth of z-position)
Read and Save the first element of each list in the variables $x[0]$,$y[0]$,$z[0]$,$lux[0]$ and $z_{groundTruth}[0]$
Close file named trainingDataForLightRange-X
//Gaussian elimination
for integer $k$= 1 to min($m$,$n$) do:
    //Find the k-th pivot
    $i_{max} := argmax$ (integer $i = k$ to $m$, $abs(A[i,k])$)
    if $A[i_{max},k] = 0$ then
        error "Matrix is singular!"
    swap rows$(k, i_{max})$
    //Do for all rows below pivot:
    for  integer $i = k+1$ to $m$ do:
        $f := A[i,k]/A[k,k]$
        Do for all remaining elements in current row:
        for integer $i = k+1$ to $n$ do:
            $A[i,j] := A[i,j] - A[k,j] * f$
        //Fill lower triangular matrix with zeros:
        $A[i,k] := 0$
    //Save the coefficients of the model on the vector $p$
    $p[i] = A[i,j]/A[i,i]$
//Compute the error of the model
$error = z_{groundTruth}[0] - p[0] - p[1] * x[0] - p[2] * y[0] - p[3] * z[0] - p[4] * lux[0]$
//Save the error in the variable p[0]
$p[0] = error$
//Start optimization process
$optimizationLossFunction(\&p)$
PRINT the model in the form Y=p[0]+p[1]x+p[2]y+p[3]z+p[4]lux
```

Bibliography

[1] Augmented Reality Market Size. Tech. rep., Global Market Insights, 2016.

[2] ABABSA, F., DIDIER, J.-Y., TAZI, A., AND MALLEM, M. Software architecture and calibration framework for hybrid optical ir and vision tracking system. In *Control & Automation, 2007. MED'07. Mediterranean Conference on* (2007), IEEE, pp. 1–6.

[3] ABABSA, F., AND MALLEM, M. A robust circular fiducial detection technique and real-time 3d camera tracking. *Journal of Multimedia 3*, 4 (2008), 34–41.

[4] ABABSA, F., AND MALLEM, M. Robust camera pose tracking for augmented reality using particle filtering framework. *Machine Vision and Applications 22*, 1 (2011), 181–195.

[5] ABABSA, F., MALLEM, M., AND ROUSSEL, D. Comparison between particle filter approach and kalman filter-based technique for head tracking in augmented reality systems. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (2004), vol. 1, IEEE, pp. 1021–1026.

[6] AUER, T., AND PINZ, A. Building a hybrid tracking system: Integration of optical and magnetic tracking. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on* (1999), IEEE, pp. 13–22.

[7] AZUMA, R., BAILLOT, Y., BEHRINGER, R., FEINER, S., JULIER, S., AND MACINTYRE, B. Recent advances in augmented reality. *IEEE computer graphics and applications 21*, 6 (2001), 34–47.

[8] AZUMA, R., HOFF, B., NEELY, H., AND SARFATY, R. A motion-stabilized outdoor augmented reality system. In *Virtual Reality, 1999. Proceedings., IEEE* (1999), IEEE, pp. 252–259.

[9] AZUMA, R. T. A survey of augmented reality. *Presence: Teleoperators and virtual environments 6*, 4 (1997), 355–385.

[10] AZUMA, R. T., HOFF, B. R., NEELY III, H. E., SARFATY, R., DAILY, M. J., BISHOP, G., CHI, V., WELCH, G., NEUMANN, U., YOU, S., ET AL. Making augmented reality work outdoors requires hybrid tracking. In *Proceedings of the First International Workshop on Augmented Reality* (1998), vol. 1.

[11] BLESER, G., HENDEBY, G., AND MIEZAL, M. Using egocentric vision to achieve robust inertial body tracking under magnetic disturbances. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 103–109.

[12] BLESER, G., WUEST, H., AND STRICKER, D. Online camera pose estimation in partially known and dynamic scenes. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on* (2006), IEEE, pp. 56–65.

[13] CHAKRABORTY, U. *Advances in differential evolution*, vol. 143. Springer, 2008.

[14] COMPORT, A. I., MARCHAND, É., AND CHAUMETTE, F. A real-time tracker for markerless augmented reality. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality* (2003), IEEE Computer Society, p. 36.

[15] DAME, A., AND MARCHAND, E. Accurate real-time tracking using mutual information. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 47–56.

[16] DHIMAN, V., RYDE, J., AND CORSO, J. J. Mutual localization: Two camera relative 6-dof pose estimation from reciprocal fiducial observation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (2013), IEEE, pp. 1347–1354.

[17] DONOSER, M., KONTSCHIEDER, P., AND BISCHOF, H. Robust planar target tracking and pose estimation from a single concavity. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 9–15.

[18] DORFMULLER-ULHAAS, K., AND SCHMALSTIEG, D. Finger tracking for interaction in augmented environments. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on* (2001), IEEE, pp. 55–64.

[19] DRETTAKIS, G., ROBERT, L., AND BOUGNOUX, S. Interactive common illumination for computer augmented reality. In *Rendering Techniques' 97*. Springer, 1997, pp. 45–56.

[20] FELDMAN, A., TAPIA, E. M., SADI, S., MAES, P., AND SCHMANDT, C. Reachmedia: On-the-move interaction with everyday objects. In *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)* (2005), IEEE, pp. 52–59.

[21] FISCHER, J., REGENBRECHT, H., AND BARATOFF, G. Detecting dynamic occlusion in front of static backgrounds for ar scenes. In *Proceedings of the workshop on Virtual environments 2003* (2003), ACM, pp. 153–161.

[22] FOURNIER, S. *A consumer-brand relationship framework for strategic brand management*. PhD thesis, University of Florida Gainesville, FL, 1994.

[23] Foxlin, E., Altshuler, Y., Naimark, L., and Harrington, M. Flighttracker: A novel optical/inertial tracker for cockpit enhanced vision. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality* (2004), IEEE Computer Society, pp. 212–221.

[24] Freeman, R. M., Juliet, S., and Steed, A. J. A method for predicting marker tracking error. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* (2007), IEEE, pp. 157–160.

[25] Fuhrmann, A., Hesina, G., Faure, F., and Gervautz, M. Occlusion in collaborative augmented environments. *Computers & Graphics 23*, 6 (1999), 809–819.

[26] Fuhrmann, A. L., Splechtna, R., and Přikryl, J. Comprehensive calibration and registration procedures for augmented reality. In *Immersive Projection Technology and Virtual Environments 2001*. Springer, 2001, pp. 219–227.

[27] Harris, C. Tracking with rigid models. In *Active vision* (1993), MIT press, pp. 59–73.

[28] Herout, A., Szentandrasi, I., Zacharia, M., Dubska, M., and Kajan, R. Five shades of grey for fast and reliable camera pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (2013), IEEE, pp. 1384–1390.

[29] Hirota, G., Chen, D. T., Garrett, W. F., Livingston, M. A., et al. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 429–438.

[30] Hoff, W. A., Nguyen, K., and Lyon, T. Computer-vision-based registration techniques for augmented reality. In *Photonics East'96* (1996), International Society for Optics and Photonics, pp. 538–548.

[31] ISO, I. 10218-2: 2011: Robots and robotic devices–safety requirements for industrial robots–part 2: Robot systems and integration. *Geneva, Switzerland: International Organization for Standardization* (2011).

[32] Ito, E., Okatani, T., and Deguchi, K. Accurate and robust planar tracking based on a model of image sampling and reconstruction process. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 1–8.

[33] Kanbara, M., Okuma, T., Takemura, H., and Yokoya, N. A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *Virtual Reality, 2000. Proceedings. IEEE* (2000), IEEE, pp. 255–262.

[34] Kato, H. Artoolkit. *http://www. hitl. washington. edu/artoolkit/* (1999).

[35] KATO, H., BILLINGHURST, M., POUPYREV, I., IMAMOTO, K., AND TACHIBANA, K. Virtual object manipulation on a table-top ar environment. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on* (2000), Ieee, pp. 111–119.

[36] KIM, K., LEPETIT, V., AND WOO, W. Keyframe-based modeling and tracking of multiple 3d objects. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 193–198.

[37] LEPETIT, V., AND BERGER, M.-O. A semi-automatic method for resolving occlusion in augmented reality. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (2000), vol. 2, IEEE, pp. 225–230.

[38] LIEBERKNECHT, S., HUBER, A., ILIC, S., AND BENHIMANE, S. Rgb-d camera-based parallel tracking and meshing. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 147–155.

[39] LIU, Y., AND WANG, Y. Ar-view: An augmented reality device for digital reconstruction of yuangmingyuan. In *Mixed and Augmented Reality-Arts, Media and Humanities, 2009. ISMAR-AMH 2009. IEEE International Symposium on* (2009), IEEE, pp. 3–7.

[40] MAIDI, M., ABABSA, F., AND MALLEM, M. Handling occlusions for robust augmented reality systems. *Journal on Image and Video Processing 2010* (2010), 4.

[41] MAIDI, M., DIDIER, J.-Y., ABABSA, F., AND MALLEM, M. A performance study for camera pose estimation using visual marker based tracking. *Machine Vision and Applications 21*, 3 (2010), 365–376.

[42] MICHALSKI, R. S., CARBONELL, J. G., AND MITCHELL, T. M. *Machine learning: An artificial intelligence approach.* Springer Science & Business Media, 2013.

[43] MILGRAM, P., TAKEMURA, H., UTSUMI, A., AND KISHINO, F. Augmented reality: A class of displays on the reality-virtuality continuum. In *Photonics for industrial applications* (1995), International Society for Optics and Photonics, pp. 282–292.

[44] MISTRY, P., KUROKI, T., AND CHANG, C. Tapuma: tangible public map for information acquirement through the things we carry. In *Proceedings of the 1st international conference on Ambient media and systems* (2008), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 12.

[45] MOHRING, M., LESSIG, C., AND BIMBER, O. Video see-through ar on consumer cell-phones. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality* (2004), IEEE Computer Society, pp. 252–253.

[46] NAIMARK, L., AND FOXLIN, E. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality* (2002), IEEE Computer Society, p. 27.

[47] NARZT, W., POMBERGER, G., FERSCHA, A., KOLB, D., MÜLLER, R., WIEGHARDT, J., HÖRTNER, H., AND LINDINGER, C. Augmented reality navigation systems. *Universal Access in the Information Society 4*, 3 (2006), 177–187.

[48] NILSSON, J., ÖDBLOM, A. C., FREDRIKSSON, J., ZAFAR, A., AND AHMED, F. Performance evaluation method for mobile computer vision systems using augmented reality. In *2010 IEEE Virtual Reality Conference (VR)* (2010), IEEE, pp. 19–22.

[49] NILSSON, N. J. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[50] PARK, J., YOU, S., AND NEUMANN, U. Natural feature tracking for extendible robust augmented realities. In *Proc. Int. Workshop on Augmented Reality* (1998), vol. 2, pp. 2–2.

[51] PARK, S.-Y., AND PARK, G. G. Active calibration of camera-projector systems based on planar homography. In *Pattern Recognition (ICPR), 2010 20th International Conference on* (2010), IEEE, pp. 320–323.

[52] PARK, Y., LEPETIT, V., AND WOO, W. Texture-less object tracking with online training using an rgb-d camera. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 121–126.

[53] RABBI, I., ULLAH, S., RAHMAN, S. U., AND ALAM, A. Extending the functionality of artoolkit to semi-controlled/uncontrolled environment. *International journal on information*, 17 (2014), 2823–2832.

[54] REITMAYR, G., AND DRUMMOND, T. Going out: robust model-based tracking for outdoor augmented reality. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2006), IEEE Computer Society, pp. 109–118.

[55] REITMAYR, G., AND SCHMALSTIEG, D. Location based applications for mobile augmented reality. In *Proceedings of the Fourth Australasian user interface conference on User interfaces 2003-Volume 18* (2003), Australian Computer Society, Inc., pp. 65–73.

[56] ROLLAND, J. P., DAVIS, L., AND BAILLOT, Y. A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augmented reality 1*, 1 (2001), 67–112.

[57] SÁNCHEZ, J. R., ÁLVAREZ, H., AND BORRO, D. Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 185–192.

[58] SCHALL, G., MULLONI, A., AND REITMAYR, G. North-centred orientation tracking on mobile phones. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 267–268.

[59] SCHMALSTIEG, D., FUHRMANN, A., AND HESINA, G. Bridging multiple user interface dimensions with augmented reality. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on* (2000), IEEE, pp. 20–29.

[60] SIMON, G. Tracking-by-synthesis using point features and pyramidal blurring. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 85–92.

[61] STEINBIS, J., HOFF, W., AND VINCENT, T. L. 3d fiducials for scalable ar visual tracking. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (2008), IEEE Computer Society, pp. 183–184.

[62] THEFABRICATOR.COM. Playing it safe with robotic welding, 2004.

[63] UCHIYAMA, H., AND MARCHAND, E. Toward augmenting everything: Detecting and tracking geometrical features on planar objects. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on* (2011), IEEE, pp. 17–25.

[64] ULLAH, S. *Multi-modal Assistance for Collaborative 3D Interaction: Study and analysis of performance in collaborative work*. PhD thesis, Citeseer, 2011.

[65] VACCHETTI, L., LEPETIT, V., AND FUA, P. Stable real-time 3d tracking using online and offline information. *IEEE transactions on pattern analysis and machine intelligence 26*, 10 (2004), 1385–1391.

[66] VILLEGAS-HERNANDEZ, Y. S., AND GUEDEA-ELIZALDE, F. Marker's position estimation under uncontrolled environment for augmented reality. *International Journal on Interactive Design and Manufacturing (IJIDeM)* (2016), 1–9.

[67] WAECHTER, C., HUBER, M., KEITLER, P., SCHLEGEL, M., KLINKER, G., AND PUSTKA, D. A multi-sensor platform for wide-area tracking. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on* (2010), IEEE, pp. 275–276.

[68] WAGNER, D., AND SCHMALSTIEG, D. *Artoolkitplus for pose tracking on mobile devices*. na, 2007.

[69] WAGNER, D., AND SCHMALSTIEG, D. Making augmented reality practical on mobile phones, part 1. *IEEE Computer Graphics and Applications 29*, 3 (2009), 12–15.

[70] WANG, J., KOBAYASHI, E., AND SAKUMA, I. Coarse-to-fine dot array marker detection with accurate edge localization for stereo visual tracking. *Biomedical Signal Processing and Control 15* (2015), 49–59.

[71] Wang, X., and Dunston, P. S. Design, strategies, and issues towards an augmented reality-based construction training platform. *Journal of information technology in construction (ITcon) 12*, 25 (2007), 363–380.

[72] White, S., Lister, L., and Feiner, S. Visual hints for tangible gestures in augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2007), IEEE Computer Society, pp. 1–4.

[73] Wuest, H., Vial, F., and Stricker, D. Adaptive line tracking with multiple hypotheses for augmented reality. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality* (2005), IEEE Computer Society, pp. 62–69.

[74] Yamauchi, M., and Iwamoto, K. Combination of optical shape measurement and augmented reality for task support: I. accuracy of position and pose detection by artoolkit. *Optical review 17*, 3 (2010), 263–268.

[75] Yang, J., and Maurer, F. Literature survey on combining digital tables and augmented reality for interacting with a model of the human body. Tech. rep., University of Calgary, 2010.

[76] Yang, Y.-R., Tsai, M.-P., Chuang, T.-Y., Sung, W.-H., and Wang, R.-Y. Virtual reality-based training improves community ambulation in individuals with stroke: a randomized controlled trial. *Gait & posture 28*, 2 (2008), 201–206.

[77] Zhou, F., Duh, H. B.-L., and Billinghurst, M. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (2008), IEEE Computer Society, pp. 193–202.

Publications

During the course of his PhD. the author had the opportunity to publish an indexed Journal.

The author's published Journal articles is: Marker's position estimation under uncontrolled environment for augmented reality - *International Journal on Interactive Design and Manufacturing (IJIDeM), (2016)*.

Int J Interact Des Manuf
DOI 10.1007/s12008-016-0356-x

ORIGINAL PAPER

Marker's position estimation under uncontrolled environment for augmented reality

Yazmin S. Villegas-Hernandez[1] · **Federico Guedea-Elizalde**[1]

Received: 12 September 2016 / Accepted: 6 October 2016
© Springer-Verlag France 2016

Abstract The optical tracking information in manufacturing can provide valuable support and time saving for autonomous operations, but ill environment conditions prevent a better performance of vision systems. In this work, a method for estimating object position under semi-controlled environment where lighting conditions change dynamically is proposed. This method incorporates regression analysis that combines light measurement and an augmented reality (AR) system. Augmented reality (AR) combines *virtual objects* with *real environment*. Furthermore, every AR application uses a video camera to capture an image including a *marker* in order to place a virtual object, which gives user an enriched environment. Using a tracking system to estimate the *marker's* position with respect to the camera coordinate frame is needed to positioning a virtual object. Most research studies on tracking system for AR are under controlled environment. The problem is that tracking systems for *markers* are sensitive to variations in lighting conditions in the *real environment*. To solve this problem, a method is proposed to better estimate a marker position based on regression analysis, where lighting conditions are taken into account. Our approach improves the accuracy of the marker position estimation under different lighting conditions. The experimental data obtained under a laboratory context with changes on light condition are fitted with this approach with an accuracy of 99 %.

Keywords Augmented reality · Marker position estimation · Lighting conditions · Marker tracking

1 Introduction

In the field of Interactive Design [1], numerous prototyping techniques have emerged, including virtual and augmented reality solutions. Prototyping techniques enhance cognitive interactions between the user and the future product, and these techniques supports decision making in design and manufacturing [2]. Virtual and augmented representations are useful for analyze engineering defaults, analyze new ideas, and allowing the brainstorming of ideas in the design process. However, the primary focus of this research is to improve augmented reality experience by improving its fidelity tracking. In order to enhancing the experience of the end-product user and have a stable 3D marker tracking, while is using augmented reality for prototyping.

Augmented reality (AR) [3] combines *virtual objects* with *real world environment*, where the users interact in real-time with it. As shown in Fig. 1, Augmented Reality lies on the right of *Real Environment*, which means that *real world environment* is augmented by adding *virtual objects*. A continuum between the *real environment* and the *virtual environment* is shown.

In order to distinguish between the different terms used in Fig. 1, a brief explanation of what *Real Environment*, *Augmented Reality*, *Augmented Virtuality* and *Virtual Environment* is given, these are full-developed on [4].

- Virtual reality (VR) environment is a completely synthetic world (with virtual objects), in which the user (participant observer) is immersed.

✉ Yazmin S. Villegas-Hernandez
yazmin.sarahi@gmail.com

Federico Guedea-Elizalde
fguedea@itesm.mx

[1] Escuela de Ingenieria y Ciencias, Tecnologico de Monterrey, Ave. Eugenio Garza Sada 2501 South, Col. Tecnologico, 64849 Monterrey, Nuevo Leon, Mexico

Published online: 24 October 2016

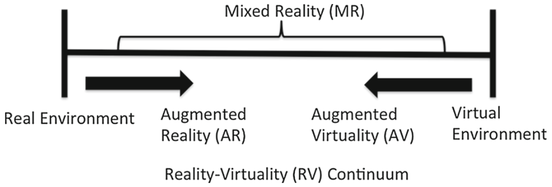


Fig. 1 Miligram's reality-virtuality (RV) continuum [4]

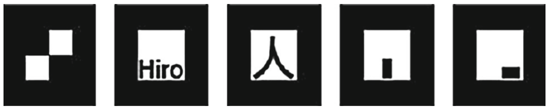


Fig. 2 Sample markers of ARToolKit

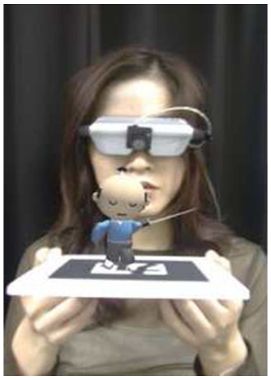


Fig. 3 ARToolKit example of augmented reality [6]

- Real environment (RE) consists solely the view of the real world when is viewed directly in person, or via some sort of a (video) display.
- Mixed reality (MR) environment combines the real world with the virtual world in a single display.
- Augmented reality (AR) refers to an indirect or direct view of the real world environment is augmented by adding virtual computer-generated information to it in real time.
- Augmented Virtuality (AV) refers to merging of real world objects into virtual worlds.

In fact every AR application uses a camera to identify a *marker* in order to place over it a *virtual object*. Different types of *markers*, used by ARToolKit, are shown in Fig. 2. As shown in Fig. 3, the *real environment* is augmented using these *markers* and a *virtual object*.

The main problem for AR systems is the inaccurate estimation of the position and pose of the *marker* in the *real environment*. The estimation of position and pose in real time is called *tracking*. In a tracking problem, *markers* are used commonly to superposed a *virtual object*, as Fig. 3. In AR, there are many types of markers, where the type of marker depends on the used toolkit. These toolkits are ARToolKit [5], ARToolkitPlus [7], ARTag [8], A Library for Virtual and Augmented Reality (ALVAR) [9], Designer's Augmented Reality Toolkit (DART) [10], etc.

The world's most widely used tracking library for augmented reality is ARToolKit [5]. This library provides an easy way to develop AR applications. Furthermore, computer vision algorithms are used in AR applications to solve the *tracking problem*.

The main challenge is that ARToolKit needs an accurate tracking system that estimates the marker coordinates with respect to the users viewpoint (or camera pose). In most cases, the accuracy of marker tracking is either ignored, assumed to be a constant value, or determined using an interpolation scheme (where error is previously measured). Most research studies on the tracking system for AR are under controlled environment. The problem is that tracking systems for markers are sensitive to variations in lighting conditions in the environment. To solve this problem, a novel method approach for marker position estimation based on machine learning is proposed and lighting conditions are taken into account. The proposed approach improves the accuracy of the marker position estimation under lighting conditions that change dynamically. In this work, the camera used only recognize markers under a light range of 40–310 lux.

In the following section, a brief definition of interactive design is given. Furthermore, a brief description of related work about different techniques used to estimate the marker position is given. Ultimately, the proposed solution and the results of the experiments are given.

2 Interactive design

The interactive design consists in making design choices from the possible interactions that could exist between the product and its environments, necessarily including Human [2]. The prototype used in the design process could be physical or virtual. The usage and adaptation of new augmented reality, augmented virtuality and virtual techniques to enhance and to enrich the experience of the interaction between the product and its user. Deeper layers of interaction with the real world, virtual-objects, virtual simulations, and the product-end user by using these technologies. Furthermore, there is also the haptic technology (which is an augmented reality technology) that is the vibration and sensation added to interaction with graphics. This technology is commonly used within a virtual reality setting to enhance the experience. Blending together all these technologies, it creates a whole new experience for the user, which leads the making-decision process.

3 Related work

Tracking systems for *markers* (*fiducials*) are used in AR applications. The tracking system can identify the *marker* using a camera in order to place the *virtual object* over the marker. There are many kinds of markers, where each toolkit has its own marker type. Most research studies are about different tracking systems on ARToolkit or another toolkit.

Rabbi et al. [11] extended the functionality of ARToolKit to a semi-controlled or uncontrolled environment using multiple files, which are recorded in different environmental conditions. This approach improved the marker tracking performance under different lighting conditions, brightness and contrast level. This approach may increase processing time, which is controlled by implementing a priority queue. This queue provides a priority to the pattern that is mostly used for tracking in the environment.

Maidi et al. [12] developed a hybrid approach for pose estimation, which mixes an iterative method based on the extended Kalman filter (EKF) and an analytical method with a direct resolution of pose parameters computation. This approach improves stability, convergence and accuracy of the pose parameters.

Herout et al. [13] introduced an improved design of the Uniform Marker Fields and an algorithm for their fast and reliable detection. This marker field is designed to be detected and to be recognized for camera pose estimation: in various lighting conditions, under a severe perspective, while heavily occluded, and under a strong motion blur. This marker field detection harnesses the fact that the edges within the marker field meet at two vanishing points and that the projected planar grid of squares can be defined by a detectable mathematical formalism. The modules of the grid are gray-scale and the locations within the marker field are defined by the edges between the modules. The detection rates and accuracy are slightly better and faster compared with state-of-the-art marker-based solutions.

Dhiman et al. [14] developed a cooperative localization method (called mutual localization), which uses two cameras (each one with a fiducial marker in a sensor specific coordinate frame), in order to estimate the 6-Degrees of freedom pose of multiple cameras. Using this approach, it can be obviated the common assumption of sensor ego-motion. This approach uses an algebraic formulation to estimate the pose of the two-camera mutual localization setup under these assumptions. This approach can localize significantly more accurately than ARToolKit.

Yamauchi et al. [15] studied the position and pose error detected by an augmented reality system (using ARToolKit library). As shown in Fig. 4, the marker is perpendicular to the line of sight of the camera. The characteristics of the marker detection are summarized as follows. The position of a marker is determined with sufficient accuracy for the directions perpendicular to the line of sight of the camera. The rotation angle of a marker around the line sight (of the camera) is also determined accurately. However, the position of the marker in the line of sight direction cannot be accurately determined. The detection error in this direction was revealed to be proportional to the square of the distance between the camera and the marker. The pose angles other than the rotation angle are also difficult to determine accurately.

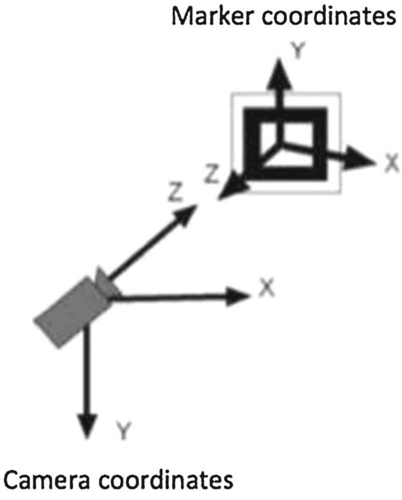


Fig. 4 Marker coordinates and camera coordinates [15]

Freeman et al. [16] proposed a method for predicting marker-tracking error in order to quantify the accuracy of the marker position. This statistical approach uses a modified Scaled Spherical Simplex Unscented Transform (SSSUT) algorithm in order to establish the maximum and minimum error of the marker-position estimations (estimated by the augmented reality system).

Wang et al. [17] presented a coarse-to-fine marker detection algorithm with sub-pixel edge localization. In this work, it is proposed a marker with a dot pattern, which is detected and matched to a predefined descriptor in a fast way using a simple threshold and hierarchical contour analysis. The algorithm yielded a feature detection error of less than 0.1 pixel (up to noise level $\sigma_n \leqslant 0.35$) with real-time performance.

In Table 1, the differences between each approach described above are shown.

Most of these approaches used for marker detection do not take into account many factors as light intensity, brightness and contrast. Rabbi's approach takes into account light intensity, but this approach may increase processing time according to the number of pattern files. The main problem is the noise of the marker position estimation, which depends on the illumination and the rotation of the marker. In this

Table 1 Tracking system approaches

Treatments	Light changes	Method	Multiple cameras
Rabbi's approach [11]	X	Multiple files	
Maidi's approach [12]		Statistical method	
Herout's approach [13]		Mathematical method	
Dhiman's approach [14]		Algebraic method	X
Yamauchi's approach [15]		Algebraic method	
Freeman's approach [16]		Statistical method	
Wang's approach [17]		Hierarchical contour analysis	
Our's approach	X	Statistical method	

paper, it is proposed to use machine learning in order to find the correlation between lighting conditions and position estimation of the marker.

4 Marker tracking using ARToolKit

ARToolKit uses a camera to identify a *marker* in order to track it and place over it a *virtual object*. As shown in Fig. 2, different patterns are around thick black edges.

The marker tracking process is as follows: first, the marker image is converted to a black and white binary image, which is called threshold, using a threshold value (in order to know if the pixel is going to be black or white). Second, the black square of the marker is detected. Third, the position of the four corners of the marker are estimated, which are used to estimate the center position of the marker as well marker pose.

ARToolKit uses functions *arDetectMarker* and *arGetTransMat* to detect and to estimate the position and pose of a marker. The *arDetectMarker* function detects the black edges of the marker as well as the pattern inside it, which is registered in the app. The *arGetTransMat* function identifies the position and pose of a marker, which is represented by a matrix, as shown in Eq. (1).

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = p \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix} \quad (1)$$

The $\left(\mathbf{x}', \mathbf{y}', \mathbf{z}'\right)$ coordinates represent the transform matrix of the camera relative to the marker frame coordinates. The transform matrix **p** is multiply by the vector of the marker coordinates relative to the camera coordinates $(\mathbf{x_m}, \mathbf{y_m}, \mathbf{z_m})$.

The marker pose is represented by three rotation angles called pitch Θ, yaw ϕ and roll ω, which denote rotations around the *x, y, z* axes, respectively. The rotation angles are calculated from the components of the transform matrix in Eq. (1) as follows:

$$\Theta = tan^{-1}\left(\frac{p_{21}}{p_{11}}\right) \quad (2)$$

$$\phi = tan^{-1}\left(\frac{p_{32}}{p_{33}}\right) \quad (3)$$

$$\omega = sin^{-1}\left(p_{31}\right) \quad (4)$$

5 Machine learning method

In machine learning, the most common technique used, depending of the problem, is the multiple linear regression. Multiple linear regression is used to predict the value of a variable **y** (called criterion variable) using multiple variables $(\mathbf{x_i}, \ldots, \mathbf{x_m})$ (called predictor variables). These predictor variables could be known or unknown. When the predictor variables are known, it is called *supervised machine learning* and the predictor variables are called *labeled* variables. When the predictor variables are unknown, it is called *unsupervised machine learning*. This leads to the following multiple regression function:

$$h\left(x_1, \ldots, x_m\right) = \Theta_0 + \sum\nolimits_{i=1}^{m} \Theta_i x_i \quad (5)$$

where Θ_0 is called the intercept and the Θ_i are called coefficients.

The process of learning consists on using mathematical algorithm in order to optimize the predictor function $\mathbf{h}(\mathbf{x_i})$, which is an estimation of the criterion variable **y**. For the optimization process, it is used training examples, which are input data of both predictor variables $(\mathbf{x_1}, \ldots, \mathbf{x_m})$ and criterion variable **y**, which is known in advance.

The loss function, as shown in Eq. (6), is used to measure the improvement of the predictor function $h(x_{t,i})$. The input Θ represents all of the coefficients that we are using in the predictor function. In this case we are using only two coefficients Θ_0 and Θ_1.

$$J\left(\Theta_0, \Theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x_{t,i}\right) - y\right)^2 \quad (6)$$

The loss function and training examples are used to know the difference between the criterion variable **y** and the estimated value $\mathbf{h}\,(\mathbf{x_1}, \ldots, \mathbf{x_m})$, which is called *error*, in order to measure the improvement of the predictor function. In the optimization process, the intercept Θ_0 and the coefficients Θ_i are changed constantly in order to converge on the best values that minimize the *error*.

6 Proposed solution

In supervised machine learning, the program is *trained* on a pre-defined set of *labeled training data*, which is used to reach an accurate *predictor function* when given new data. Each *training example* is a pair consisting of *input data* and a desired *output value* (called *supervisory signal*). In the proposed approach, it is going to be used *labeled training examples* where the outcome is marker position data and the variables are estimated marker position and light condition. The light label is subdivided into *n* subsets in order to produce *n* inferred functions. The system learns different predictor functions according to the intensity light range that the environment has in this moment. Using this approach, the noise caused by the light and other factors is minimized and estimations of marker position can be done.

The proposed method needs an initial preparation offline before using this method properly in order to make any marker position estimations. Firstly, before taking any sample, it is needed to prepare the experimental setup (as shown in Fig. 7) with the luxmeter tool, the camera (Logitech C920), the marker (fiducial), the computer (2.8 GHz Intel Core i7 MacBook Pro), CNC rail (for training), and stages (to fix these tools). Secondly, the samples (of estimated marker positions under different light conditions) are taken and saved in a text file when the fiducial along the CNC rail is moving and the light measurement is taken with a luxmeter tool. It was used CNC rail in order to get the ground-truth position of the marker. The data is divided into three categories of light range and saved into three different files. Thirdly, the data saved in a text file by the ARToolKit application is the input of the machine learning application written in c++, which made the regression analysis with the data. Algorithm 1 shows the training process using multiple linear regression to generate a model for each range light. The machine learning application returns predictor functions for each light range. These predictor functions are used to estimate the marker position.

The light range used in this work was chosen through the statistical analysis of the sample data taken. In the analysis of the data, it was found that three groups have the same mean error. In the light range from 40 to 100 lux, the mean marker position error was of 40.69 mm. In the light range from 100 to 200 lux, it was found that the mean marker error was of 40.13 mm. In the light range from 200 to 310 lux, it was found that the mean marker position error was of 69.76 mm. Then, these three groups of light measurement are used in our experiment as three range light.

Algorithm 1 Machine learning algorithm (for training)

```
Open the file with name trainingDataForLightRange-X for input
Read and Save training data matrix A[i,k] ( which is an $m$ x $n$ matrix with the XYZ positions, the light measurements, and the ground truth of z-position)
Read and Save the first element of each list in the variables $x[0]$,$y[0]$,$z[0]$,$lux[0]$ and $z_{groundTruth}[0]$
Close file named trainingDataForLightRange-X
//Gaussian elimination
for integer $k$= 1 to min($m$,$n$) do:
    //Find the k-th pivot
    $i_{max} := argmax$ (integer $i = k$ to $m$, $abs(A[i,k])$)
    if $A[i_{max}, k] = 0$ then
        error "Matrix is singular!"
    swap rows($k$, $i_{max}$)
    //Do for all rows below pivot:
    for integer $i = k + 1$ to $m$ do:
        $f := A[i,k]/A[k,k]$
        Do for all remaining elements in current row:
        for integer $i = k + 1$ to $n$ do:
            $A[i,j] := A[i,j] - A[k,j] * f$
        //Fill lower triangular matrix with zeros:
        $A[i,k] := 0$
    //Save the coefficients of the model on the vector $p$
    $p[i] = A[i,j]/A[i,i]$
//Compute the error of the model
$error = z_{groundTruth}[0] - p[0] - p[1]*x[0] - p[2]*y[0] - p[3]*z[0] - p[4]*lux[0]$
//Save the error in the variable p[0]
$p[0] = error$
//Start optimization process
$optimizationLossFunction(\&p)$
PRINT the model in the form Y=p[0]+p[1]x+p[2]y+p[3]z+p[4]lux
```

Figure 5 depicts the proposed method. Firstly, the system receives as input the sample data under different position and light conditions. The light measurements are taken with a luxmeter tool. Initial marker position estimations (or sample data) are taken using ARToolKit application. In Algorithm 2 is shown the process for getting the marker's position. Secondly, machine learning system (Algorithm 1) builds a model to fit the marker position data using the initial position estimations and light condition data. The training is finished when the system stops receiving training examples (or samples). Thirdly, the system find the best predictor function for marker position estimation.

In the proposed machine learning system, a multiple linear regression algorithm is used. The system learns from previous computations and new training examples in order to produce new models for AR marker position estimation. As shown in Fig. 6, The training process ends when the system stops receiving training data.

Algorithm 2 ARToolKit algorithm (to get position data)

```
Open the file with name trainingData
Read and Save marker file into marker1 variable
Detect camera
if camera is detected then
    loop//for video streaming
        Grab an image of the video stream and Save it into image variable
        //Detect marker and Save the marker information into marker_info variable
        arDetectMarkerLite(dataPtr, thresh, &marker_info,
        &marker_num)
        //ARToolKit function to estimate marker's position
        arGetTransMat(marker_info, target_center, target_width, target_trans)
        //ARToolKit function to get the inverse of the marker's
        //position and orientation.
        arUtilMatInv(target_trans, cam_trans)
        Save x-y-z position into the file trainingData
Close file named trainingData
```

Experiments were conducted to determine the accuracy of optical tracking using this proposed method. The experimental setup is shown in Fig. 7. A fiducial (marker) with a 40 mm was fixed in a rail of a CNC machine. A video camera (Logitech c920) was fixed in front of the fiducial. A professional luxmeter was placed next to the fiducial. The video camera and the fiducial were adjusted using stage support in order to have their axes aligned. The experiment consists in keeping constant the x- and y-positions of the fiducial when the fiducial along the CNC rail is moving. Using a video camera (Logitech c920) with a resolution of 1080p, images were processed by a computer to get the xyz positions of the fiducial using ARToolKit functions. Using a luxmeter, the light of the fiducial area was measured. Three experiments were repeated under different light conditions while the z-direction was incremented by 10 mm. The samples of xyz positions estimated by ARToolKit were taken using dynamic light changing in order to find a predictor function for marker position estimation taking into account *light variable*.

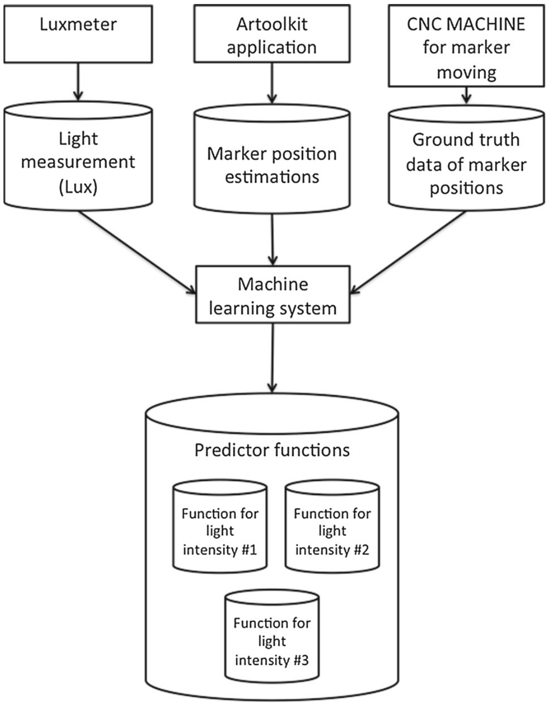

Luxmeter
Artoolkit application
CNC MACHINE for marker moving
Light measurement (Lux)
Marker position estimations
Ground truth data of marker positions
Machine learning system
Predictor functions
Function for light intensity #1
Function for light intensity #2
Function for light intensity #3

Fig. 5 Machine learning system for AR marker position estimation

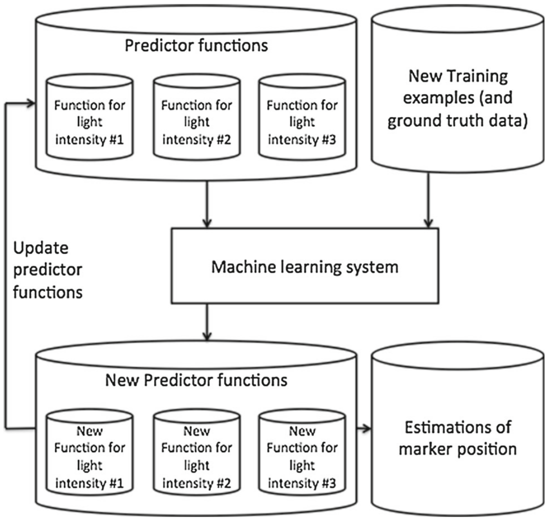

Predictor functions
Function for light intensity #1
Function for light intensity #2
Function for light intensity #3
New Training examples (and ground truth data)
Update predictor functions
Machine learning system
New Predictor functions
New Function for light intensity #1
New Function for light intensity #2
New Function for light intensity #3
Estimations of marker position

Fig. 6 Optimization process of predictor

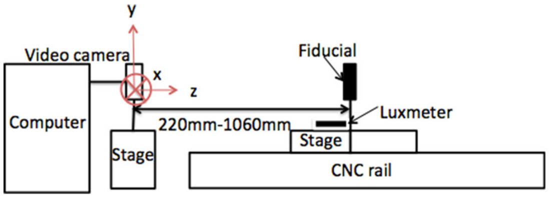


Fig. 7 Experimental setup

Yamauchi et al. [15] shows that the position errors (using ARToolKit libraries) in the x- and y- directions are relative small and that the errors in the z-direction are large. So, the z-position error is the only error variable that is going to be analyzed in this work. Three different samples were taken in order to clarify the position error of the fiducial position (estimated by ARToolKit functions) in the z-direction using dynamic light changing. Figs. 8, 9, and 10 shows the experimental results of position detection error on Sample 1, 2, and 3 respectively. In the first experiment, the fiducial was moving through z-direction from 520 to 1060 mm. In the sec-

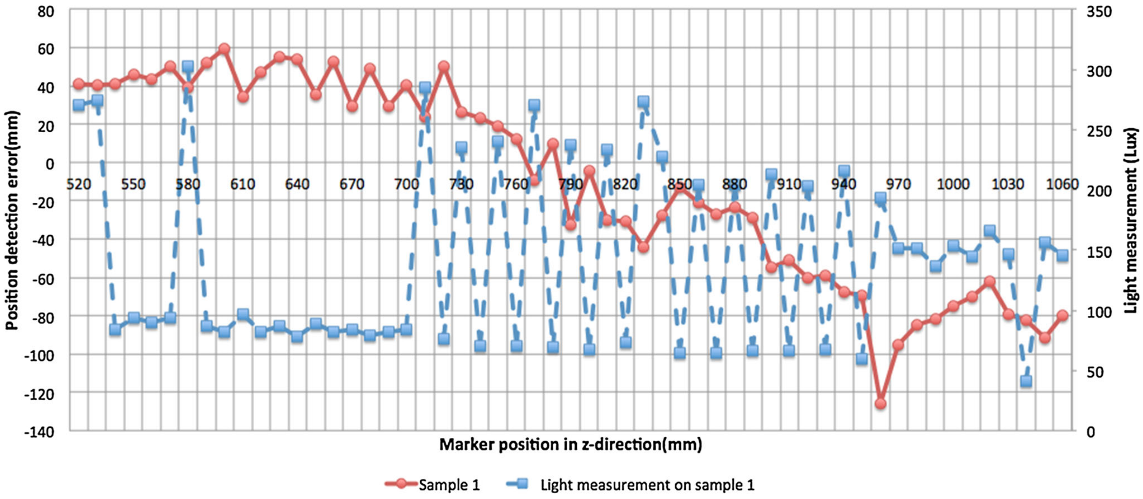


Fig. 8 Experimental results of position detection error on Sample 1

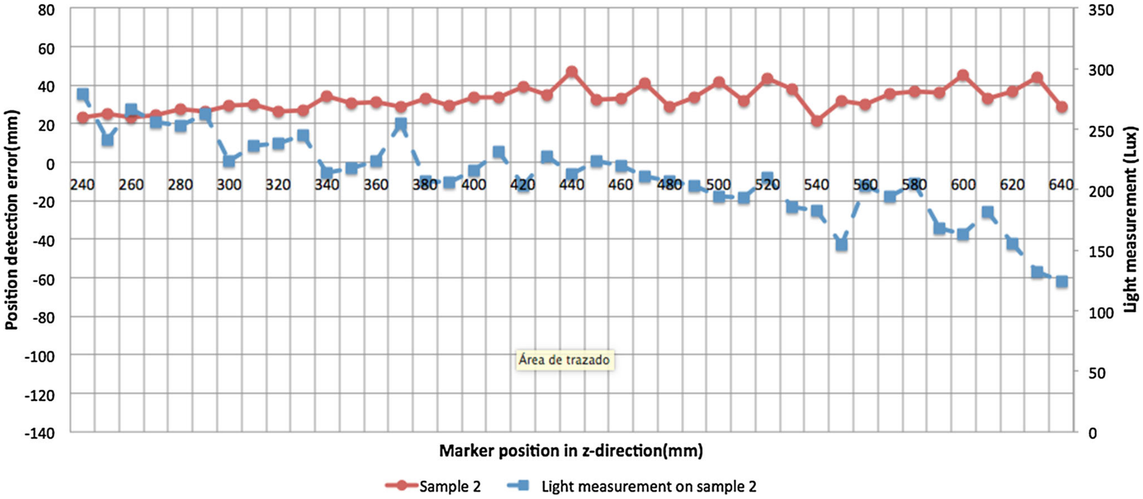


Fig. 9 Experimental results of position detection error on Sample 2

ond experiment, the fiducial was moving through z-direction from 240 to 640 mm. In the third experiment, the fiducial was moving through z-direction from 220 to 620 mm. In addition, the light conditions change dynamical from the range 41–303 lux. In this experiment, it was found that out the range of 41–303 lux the camera system cannot detect the marker.

7 Experiment and results

In sample 1, Fig. 8, the light range used in this sample goes from 41 to 303 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker position error using ARToolKit approach is −10.49 mm.

In sample 2, Fig. 9, the light range used in this sample goes from 124 to 279 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker position error using ARToolKit approach is −49.70 mm.

In sample 3, Fig. 10, the light range used in this sample goes from 116 to 256 lux, where the light changes dynamically and randomly. In this sample, the mean of the marker position error using ARToolKit approach is 20.58 mm.

In these three samples, it was found that the light conditions change the marker position estimation using ARToolKit libraries. The mean of the marker position error was not constant in these samples because of the light conditions were different in each sample. In our proposed approach, the *light*

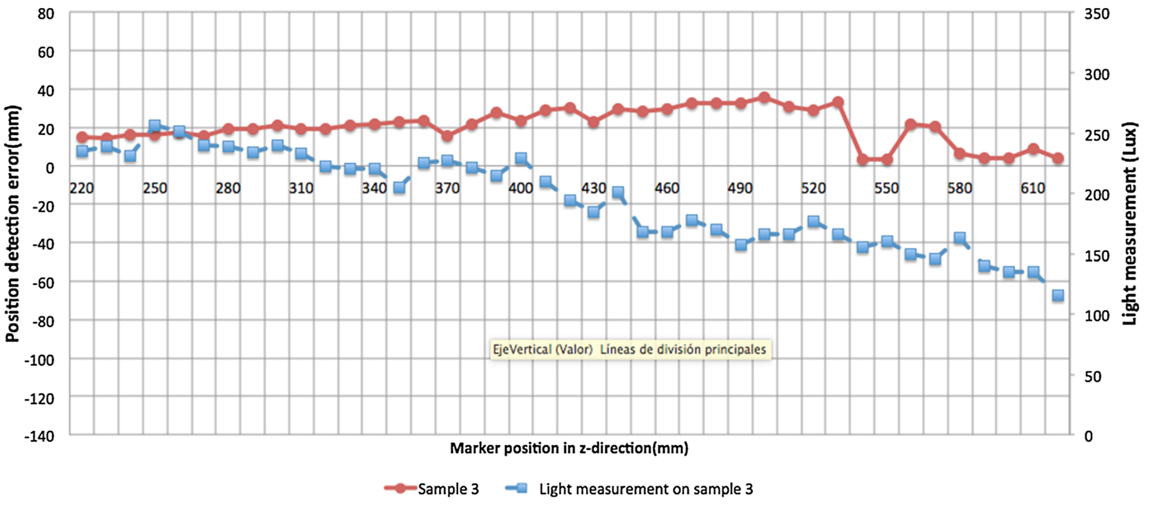


Fig. 10 Experimental results of position detection error on Sample 3

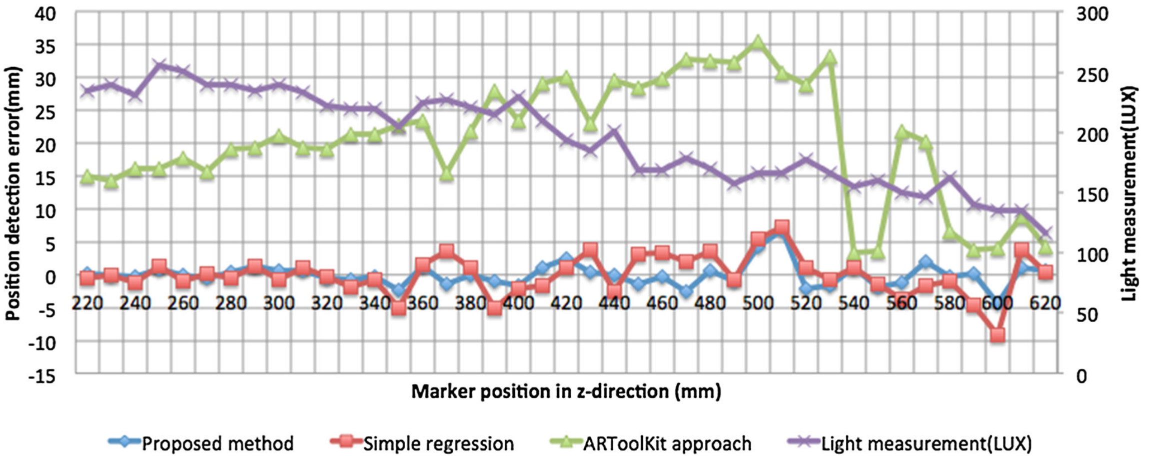


Fig. 11 Experimental results of position detection error on Sample 3 using three different approaches

condition variable is taken into account in order to have better marker position estimations.

In the statistical analysis of the samples, it was found that the data has a normal distribution. It was used the Anderson-Darling normality test on the three samples in order to know if the samples have a normal distribution. It is possible to use multiple linear regression in order to generate models to estimate marker's position, because the data came from a normal distribution.

Figure 11 shows the position error of the marker position in z-direction using three different approaches in sample 3, which are ARToolKit approach, simple regression approach, and the proposed approach. The ARToolKit approach refers to get estimations of marker position using ARToolKit libraries. The simple regression analysis approach refers to generate predictor functions of the data without be classified by light range. The proposed approach consists in classify the data by light range and then do a multiple regression analysis to the data. In this experiment, the three light ranges used are from 40 to 100, 100 to 200 and 200 to 310 lux.

As shown in Fig. 11, it was found that using the proposed method and taking into account light conditions, the error of position estimation can be reduced. The mean of the position error using this approach is of 2.657 mm, additionally, the mean of the position error using ARToolKit approach is 20.58 mm. Using this approach, it is obtained better results than using only ARToolKit approach. Furthermore, this approach has better results if the training set is increased. The position error was reduced 87 % in this scenario. Furthermore, nearly 99 % of the total variability in the response variable (marker position) is accounted for by the estimated marker position by ARToolKit and the light measurement. In this experiment, the regression line models have a $R^2 = 0.99$ that indicates a strong linear relationship between the ground truth and the predictor variables (the estimated marker position and the light conditions).

8 Conclusions

A novel method for estimating marker position under semi-controlled environment in which the lighting conditions, brightness and contrast level change dynamically is proposed. Augmented reality combines virtual models with the real environment. The light condition changes dynamically and has a big variation depending on the position in the real world, furthermore under some light range the marker position estimation has the same variance. The proposed approach improves the accuracy of the camera-pose estimation under lightning conditions that change dynamically.

In this experiment, the light conditions were different in each sample taken, which affected the position estimations, but the noise of light conditions and camera parameters was significantly reduced using the machine learning approach. The position error does not have a constant tendency because it depends on many factors such as the camera parameters and the lighting conditions. Additionally, our method improves the results when more training data is given.

This new approach of using machine learning for fitting AR marker estimations taking into account light measurement is a novel way to reduce noise and understand better the relationships between light measurement and AR marker position estimations.

References

1. Fischer, X., Nadeau, J.: Research in Interactive Design vol. 3. Springer, Paris (2011)
2. Fischer, X., Nadeau, J.: Research in Interactive Design vol. 2. Springer, Paris (2006)
3. Furht, B.: Handbook of Augmented Reality. Springer Science, Business Media. Boca Raton, Florida (2011)
4. Milgram, P., Takemura, H., Utsumi, A., Kishino, F.: Augmented reality: a class of displays on the reality-virtuality continuum. Photonics for Industrial Applications. In: International Society for Optics and Photonics, pp. 282–292 (1995)
5. Kato, H.: Inside ARToolKit. In: 1st IEEE International Workshop on Augmented Reality Toolkit (2007)
6. Carvalho, E., Maçães, G., Brito, P., Varajão, I., Sousa, N.: Use of Augmented Reality in the furniture industry. In: First Experiment Int. Conference (2011)
7. Wagner, D., Schmalstieg, D.: ARToolKitPlus for pose tracking on mobile devices. In: Proceedings of 12th Computer Vision Winter Workshop, pp.139–146 (2007)
8. Fiala, M.: ARTag, a fiducial marker system using digital techniques. Computer Vision and Pattern Recognition. In: IEEE Computer Society Conference. 2, pp. 590–596 (2005)
9. Ajanki, A., Billinghurst, M., Gamper, H., Järvenpää, T., Kandemir, M., Kaski, S., Koskela, M., Kurimo, M., Laaksonen, J., Puolamäki, K.: An augmented reality interface to contextual information. Vir. Real. **15**, 161–173 (2011)
10. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.: DART: a toolkit for rapid design exploration of augmented reality experiences. In: Proceedings of the 17th annual ACM symposium on User interface software and technology. ACM, pp. 197–206 (2004)
11. Rabbi, I., Ullah, S., Rahman, S., Alam, A.: Extending the functionality of ARToolKit to semi-controlled/uncontrolled environment. Int. J. Inf. **17**, 2823–2832 (2014)
12. Maidi, M., Didier, J., Ababsa, F., Mallem, M.: A performance study for camera pose estimation using visual marker based tracking. Mach. Vis. Appl. **21**, 365–376 (2010)
13. Herout, A., Szentandrasi, I., Zacharia, M., Dubska, M., Kajan, R.: Five shades of grey for fast and reliable camera pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 1384–1390 (2013)
14. Dhiman, V., Ryde, J., Corso, J.: Mutual localization: two camera relative 6-DOF pose estimation from reciprocal fiducial observation. In: Intelligent Robots and Systems. IEEE, pp. 1347–1354 (2013)
15. Yamauchi, M., Iwamoto, K.: Combination of optical shape measurement and augmented reality for task support: I. Accuracy of position and pose detection by ARToolKit. Opt. Rev. **17**, 263–268 (2010)
16. Freeman, R., Juliet, S., Steed, A.: A method for predicting marker tracking error. In: Mixed and Augmented Reality. IEEE, pp. 157–160 (2007)
17. Wang, J., Kobayashi, E., Sakuma, I.: Coarse-to-fine dot array marker detection with accurate edge localization for stereo visual tracking. Biomed. Signal Process. Control **15**, 49–59 (2015)

Curriculum Vitae

Yazmín Sarahí Villegas Hernández was born in Monterrey Nuevo León, México on November 25th, 1987. She received the degree of Bachelor of Industrial Engineering from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey in December 2009. She received the degree of Master of Science in Manufacturing Systems, Campus Monterrey in December 2011. She was accepted in the graduate program in Science and Engineering in January 2012 where she is currently a Ph.D. candidate. Her research interests include automated systems, robotics, machine learning, and augmented reality.

She was Project Engineer Jr. in Cemex from 2007-2008 in the area of Information technology. She was Project Engineer Jr. in Mabe from 2008-2009 in the area of Quality Control.

This document was typed in using LaTeX 2ε[a] by Yazmín Sarahí Villegas Hernández.

---

[a] The style file `phdThesisFormat.sty` used to set up this thesis was prepared by the Center of Intelligent Systems of the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey Campus