

DESIGNING A RANGE SCANNER ON AN EMBEDDED PROCESSOR USING
COLOR CODED STRUCTURED LIGHT

by
Rıfat Benveniste

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy
in
Electrical and Electronics Engineering

Yeditepe University
2011

DESIGNING A RANGE SCANNER ON AN EMBEDDED SYSTEM USING COLOR
CODED STRUCTURED LIGHT

APPROVED BY:

Assoc. Prof. Dr. Cem Ünsalan
(Supervisor)



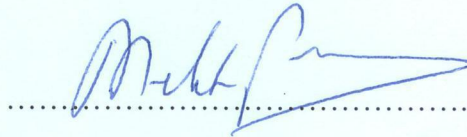
Prof. Dr. A. Coşkun Sönmez



Assist. Prof. Dr. Dionysis Goularas



Prof. Dr. Melih Pazarıcı



Assoc. Prof. Dr. Y. Sinan Akgül



DATE OF APPROVAL:/...../.....

ACKNOWLEDGEMENTS

Pursuing a PhD. Project is a both painful and enjoyable experience. It is a long journey with bitterness, hardships, frustration, encouragement, trust and with so many people's kind help. I would here like to express my thanks to the people who have been very helpful to me during the time it took me to write this thesis.

First of all I would like to thank to my supervisor Assoc. Prof. Dr. Cem ÜNSALAN. He has tirelessly devoted his time and patience to my study. I have learned a lot of him. Without his inspirational guidance, contributions and ideas, I could never finish my doctoral work.

I wish to express my gratitude to my advisory committee members Prof. Dr. Melih PAZARCI and Assist. Prof. Dr. Dionysis GOULARAS. With their constructive criticism and guidance they facilitated to complete my thesis work. I am also grateful to the official referees of the dissertation Assoc. Prof. Dr. Yusuf Sinan AKGÜL and Prof. Dr. A. Coşkun SÖNMEZ by their valuable reviews and revisions. I want to give a special thanks to my teachers, colleagues and friends in Yeditepe University Department of Electrical and Electronics Engineering and the Computer Vision Laboratory.

I want to express my grateful thanks to my mother. I felt of her prayers from heaven always in my heart. I am also very grateful to my father, who gave me the inspiration from the childhood to be an electronics engineer. I thank to my wife's family to feel their support behind me. I also thank to my sister and nephew for all their love and encouragement.

Finally, all the best love and wishes are to my wife Elif ÖZTÜRK BENVENİSTE, who makes my success significant. Without her support and trust, I could never complete this thesis. Although she was pregnant, without complaining she endured all the problems of my work at the laboratory and home. I also thank to my son Kağan BENVENİSTE for his endless kicks inside his mother in the middle of the night to make us awake.

ABSTRACT

DESIGNING A RANGE SCANNER ON AN EMBEDDED PROCESSOR USING COLOR CODED STRUCTURED LIGHT

Three dimensional range data provides useful information for various computer vision and computer graphics applications. For these, extracting the range data reliably is utmost important. Therefore, various range scanners based on different working principles are proposed in the literature. Among these, coded structured light based range scanners are popular and used in most industrial applications. Unfortunately, these range scanners cannot scan shiny objects reliably. Either highlights on the shiny object surface or the ambient light in the environment disturb the codeword. As the code is changed, the range data extracted from it will also be disturbed. In this Ph.D. study, we focus on developing a system that can scan shiny and matte objects under ambient light. Therefore, we propose color invariant based single stripe, binary, ternary, and quaternary coded structured light based range scanners. We hypothesize that, by using color invariants we can eliminate the effect of highlights and ambient light in the scanning process. Therefore, we can extract the range data of shiny and matte objects in a robust manner. We implemented these scanners using a TI DM6437 EVM board with a flexible system setup such that the user can select the scanning type. Furthermore, we implemented a TI MSP430 microcontroller based rotating table system that accompanies our scanner. By the help of this system, we can obtain the range data of the target object from different viewpoints. We also implemented a range image registration method to obtain the complete object model from the range data extracted. We tested our scanner system on various objects and provided their range and model data.

ÖZET

RENK KODLAMALI YAPISAL IŞIK KULLANARAK GÖMÜLÜ SİSTEM ÜZERİNDE BİR DERİNLİK TARAYICISI TASARIMI

Üç boyutlu derinlik verisi bilgisayarlı görü ve bilgisayar grafiği uygulamaları için kullanışlı bilgi sağlamaktadır. Bu tarayıcılar için en önemlisi güvenilir derinlik verisi çıkartmalarıdır. Bu nedenle, literatürde değişik çalışma prensibine sahip çeşitli derinlik tarayıcıları önerilmektedir. Bunların arasında kodlu yapısal ışık temelli derinlik tarayıcı yaygın ve endüstriyel uygulamalarda sıkça kullanılmaktadır. Ne yazık ki bu derinlik tarayıcıları parlak nesnelere güvenilir şekilde tarayamamaktadır. Parlak nesne yüzeyi üzerindeki ışık yansımaları ve ortam aydınlatması kod dizilimini karıştırmaktadır. Kod dizilimi değiştiğinden elde edilen derinlik verisi de karışmaktadır. Bu doktora çalışmasında biz aydınlık ortamda parlak ve mat nesnelere tarayabilen bir derinlik tarayıcısı tasarlamaya odaklandık. Bu nedenle renk değişmezi tabanlı tek çizgi, ikili, üçlü ve dördü kodlamalı yapısal ışık temelli derinlik tarayıcısı öneriyoruz. Renk değişmezlerini kullanarak parlama ve ortam aydınlatmasının etkilerini giderebileceğimizi hipotez olarak savunuyoruz. Bu sayede, parlak ve mat nesnelere derinlik verisini sağlıklı bir şekilde elde edebiliriz. Bu tarayıcıları TI DM6437 EVM kartı üzerinde kullanıcının tarayıcıyı seçebileceği esnek bir sistem olarak gerçekledik. Ayrıca TI MSP430 mikrokontrolör tabanlı bir döner tabla sistemi tarayıcımıza eşlik etmektedir. Bu sistemin yardımıyla, hedef nesnenin farklı bakış açılarından derinlik verisi elde edilebiliyoruz. Ayrıca bir derinlik görüntüsü eşleştirme yöntemi kullanarak elde edilen derinlik verisinden nesnenin tamamının modelini elde ediyoruz. Tarayıcı sistemimizi çeşitli nelerle test ederek modellerini elde ettik.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. STRUCTURED LIGHT BASED RANGE SCANNERS	3
2.1. SINGLE STRIPE BASED SCANNERS	4
2.2. MULTIPLE STRIPE BASED SCANNERS	5
2.2.1. Binary Coding	5
2.2.2. Gray Coding	6
2.2.3. N-ary Gray Coding	7
2.3. PROBLEM OF SCANNING SHINY SURFACES	7
2.4. SOLUTIONS PROPOSED IN THE LITERATURE	8
2.5. OUR SOLUTION FOR SCANNING SHINY SURFACES	9
3. HARDWARE OF THE DEVELOPED RANGE SCANNER SYSTEM	10
3.1. HARDWARE SETUP	10
3.2. THE GEOMETRIC MODEL OF THE CAMERA	12
3.2.1. Intrinsic and Extrinsic Camera Parameters	13
3.2.2. Camera Calibration	14
3.3. SYSTEM CALIBRATION IN PRACTICE	15
3.3.1. Camera Calibration in Practice	15
3.3.2. Projection Device Calibration in Practice	17
3.3.3. Overall System Calibration	18
3.4. 3D POINT CALCULATION BASED ON TRIANGULATION	18
3.4.1. Three Dimensional Point Cloud Extraction	18
3.5. EMBEDDED SYSTEMS	20
3.5.1. Microprocessors	20
3.5.2. Microcontrollers	22
3.5.3. DigitalSignalProcessors	23
4. RANGE SCANNING USING COLOR INVARIANTS	24
4.1. THE Ψ COLOR INVARIANT	24
4.1.1. Derivation of the Proposed Color Invariant using PCA	25
4.1.2. Properties of the Ψ Color Invariant	27
4.2. STRIPE SEGMENTATION USING THE Ψ COLOR INVARIANT	28

4.3.	THE ‘c’ COLOR INVARIANT SET	30
4.3.1.	Properties of the ‘c’ Color Invariant Set	31
4.4.	DECODING PATTERNS USING ‘c’ COLOR INVARIANTS.....	31
4.4.1.	Decoding Binary Patterns.....	32
4.4.2.	Decoding Ternary Patterns.....	36
4.4.3.	Decoding Quaternary Patterns.....	38
5.	EMBEDDED SYSTEM IMPLEMENTATION	40
5.1.	TI DM6437 EVM BOARD PROPERTIES.....	40
5.1.1.	The DSP Platform	40
5.1.2.	Video Input/Output Peripherals.....	41
5.1.3.	Programming and Debugging Issues.....	42
5.1.4.	DSP Configuration for Image Input/Output.....	42
5.1.5.	Properties of the Rotating Table.....	43
5.1.5.1.	Motor driving circuit.....	43
5.1.5.2.	The MSP430 microcontroller	44
5.2.	THE SCANNER SOFTWARE	45
5.2.1.	Pattern Generation.....	46
5.2.2.	Shadow Removal.....	47
5.2.3.	Image Capturing	48
5.2.4.	Pattern Decoding	48
5.2.5.	Three Dimensional Point Cloud Extraction	49
5.2.6.	Scanning Objects from Different Viewing Angles	49
5.2.7.	Transfer of Point Cloud Data	49
6.	OVERALL PERFORMANCE OF THE SCANNER SYSTEM.....	51
6.1.	EXTRACTED RANGE DATA USING STANDARD BINARY SCANNER.....	52
6.2.	EXTRACTED RANGE DATA USING OUR SCANNERS	53
6.2.1.	Binary Range Scanners	53
6.2.2.	The Ternary Range Scanner	54
6.2.3.	The Quaternary Range Scanner.....	55
6.3.	QUANTITATIVE COMPARISON OF THE SCANNER RESULTS.....	56
6.4.	TIMING PERFORMANCE OF THE SCANNER SYSTEM.....	58
6.5.	ACCURACY OF THE SCANNER SYSTEM.....	60
6.6.	COMMENTS ON THE PERFORMANCE.....	62

7. 3D MODEL CONSTRUCTION	63
7.1. 3D MODEL CONSTRUCTION USING ICP	63
7.2. PROPERTIES OF THE TEST OBJECTS.....	65
7.3. OBJECT MODELS EXTRACTED BY THE SCANNER SYSTEM.....	68
8. CONCLUSIONS.....	74
APPENDIX A: OTHER INVARIANTS FOR THE SCANNER SYSTEM.....	76
A.1. Invariants Tested for the Binary Coded Structured Light Scanner.....	77
A.2. Invariants Tested for the Ternary Coded Structured Light Scanner	81
A.3. Invariants Tested for the Senary Coded Structured Light Scanner.....	85
A.3.1 Opponent Color Theory.....	86
A.3.2 The Designed Pattern for Senary Coding	87
A.3.3 Color Segmentation for Senary Patterns	87
A.3.4 Extracted Range Data using Senary Coding	88
REFERENCES	90

LIST OF FIGURES

Figure 2.1. Schematic of a structured light system.....	3
Figure 2.2. Example of a single stripe scanning system.....	4
Figure 2.3. Example of three level binary patterns.....	5
Figure 2.4. Example of three level Gray coded patterns	6
Figure 2.5. Example of highlight problems on a shiny object surface	8
Figure 3.1. System layout	10
Figure 3.2. Our range scanner from different viewpoints.....	12
Figure 3.3. Perspective projection of a point P	13
Figure 3.4. The checkerboard used for calibration	15
Figure 3.5. An example on image taking for calibration.....	16
Figure 3.6. An example of checkerboard pattern projection from projection device.....	17
Figure 3.7. The user interface of Bouguet's camera calibration toolbox	17
Figure 3.8. Triangulation based on epipolar geometry.....	19
Figure 3.9. Typical layout of a microprocessor	21
Figure 3.10. Layout of a microcontroller.....	22

Figure 3.11. Typical layout of a DSP chip	23
Figure 4.1. Segmentation example for binary and single stripe scanners using ψ	29
Figure 4.2. Matte and shiny Atatürk objects.....	33
Figure 4.3. Binary pattern decoding results using c_1	34
Figure 4.4. Binary pattern decoding results using c_3	35
Figure 4.5. Binary pattern decoding results using black and white stripes	36
Figure 4.6. Ternary pattern decoding results using s	38
Figure 4.7. Quaternary pattern decoding results using s	39
Figure 5.1. The DM6437 EVM	41
Figure 5.2. The image input output structure.....	43
Figure 5.3. Rotating table motor driving schematic	44
Figure 5.4. Rotary table image.....	45
Figure 5.5. Starting the scanner from CCS.....	46
Figure 5.6. Selecting the scanning method by the user	46
Figure 5.7. Starting the scanning process	48
Figure 5.8. Transferring the point cloud data to the host computer.....	50
Figure 6.1. First set of test objects	51

Figure 6.2. Point clouds of eight test objects using the standard binary range scanner	52
Figure 6.3. Point clouds of eight test objects using the binary range scanner (with c_1).	53
Figure 6.4. Point clouds of eight test objects using the binary range scanner (with c_3).	54
Figure 6.5. Point clouds of eight test objects using the ternary range scanner.....	55
Figure 6.6. Point clouds of eight test objects using the quaternary range scanner.	56
Figure 6.7. The staircase object for testing the accuracy of range scanners.....	61
Figure 7.1. ICP implementation example under MATLAB	65
Figure 7.2. Second set of test objects.....	66
Figure 7.3. Models 1	68
Figure 7.4. Models 2	69
Figure 7.5. Models 3	70
Figure 7.6. Models 4	71
Figure 7.7. Models of the flat objects	72
Figure 7.8. Our face scans and their texture mapped versions	72
Figure 7.9. Our body and hand scans.....	73
Figure A.1. The metal plate object used as a benchmark	76

Figure A.2. Hue and normalized color results for the binary scanner.....	78
Figure A.3. The c color invariant results for the binary pattern	79
Figure A.4. The l color invariant results for binary pattern.....	80
Figure A.5. Lab, XYZ and xyY color space results	81
Figure A.6. Projected pattern and the Hue result.....	82
Figure A.7. Normalized color results for the ternary pattern.....	82
Figure A.8. The 'c' color invariant results for ternary pattern	83
Figure A.9. The l color invariant results for the ternary pattern.....	84
Figure A.10. Lab, XYZ and xyY color space results	85
Figure A.11. Opponent-color encoding schematic	86
Figure A.12. Segmented six colors in the senary pattern	88
Figure A.13. Range data extracted by the senary scanner.....	89

LIST OF TABLES

Table 2.1. Binary coding example for eight stripes	6
Table 2.2. Gray coding example for eight stripes	7
Table 6.1. Average percentage (%) of outliers for the scanners	57
Table 6.2. Average percentage (%) of missing points for the scanners.....	58
Table 6.3. Operation timings in milliseconds for the scanners.....	59
Table 6.4. Comparison of the actual and the measured depth values (in millimeters) on the staircase test object.....	61
Table 7.1. Dimensions of test objects (in millimeters) used in experiments	67

LIST OF SYMBOLS / ABBREVIATIONS

A	Intrinsic parameter matrix
a	Projector camera coupling parameters
B	Temporary matrix to calculate calibration parameters
B	Blue color value
b_p	Projected blue band color value
C_m	Covariance matrix
c	Color invariant set
D	Distance Function
d	Dimensionality of measurement
e	Distance
F	Set of points as reference
G	Green color value
g_p	Projected green band color value
H	Homography
h	Columns of homography
Kp_i	Covariance matrix
k	Reflectance parameters
L	Levenberg resulting matrix
l	Color invariant set
M	A set of correlated random vectors
n	Numeral
O	Set of points to be registered
P	Point in real world
p	Point in virtual plane
pc_i	Principal components
p_i	Image point corrupted by noise
pr_j	Projection on eigenvector
Q	Linear transformation matrix
q_i	Eigenvectors

R	Rotation matrix
<i>R</i>	Red color value
<i>RG</i>	Opponent colors
r	Columns of rotation matrix
r_p	Projected red band color value
<i>S</i>	Slope
s	Arbitrary scalar
<i>s</i>	Difference of c_1 and c_3 color invariants
\bar{T}	Translation vector
t	Translation vector
<i>u</i>	Image ideal horizontal axis
\hat{u}	Measured image coordinate
<i>v</i>	Image ideal vertical axis
\hat{v}	Measured image coordinate
<i>W</i>	Ambient white illumination
<i>X</i>	Focal plane axis
<i>x</i>	Real coordinates
\hat{x}	Measured normalized image coordinate
<i>Y</i>	Focal plane axis
<i>YB</i>	Opponent colors
<i>y</i>	Real coordinates
\hat{y}	Measured normalized image coordinate
<i>Z</i>	Focal plane axis
α	Image scale factor based on focal length
α_i	Scaling scalars
β	Image scale factor based on focal length
γ	Skewness
λ	Arbitrary scale factor
μ_m	Mean of sample vector
Ψ	Color invariant

3D	Three Dimensional
ADC	Analog to Digital Converter
ALU	Arithmetic Logic Unit
CCD	Charge Coupled Device
CCS	Code Composer Studio
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CSL	Chip Support Library
DAC	Digital to Analog Converter
DC	Direct Current
DLP	Digital Light Processor
DSP	Digital Signal Processor
DVSDK	Digital Video Software Development Kit
EVM	Evaluation Module
FIFO	First In First Out
GPIO	General Purpose Input Output
IC	Integrated Circuit
ICP	Iterative Closest Point
I2C	A two wire serial communication interface
KB	Kilobytes
LCD	Liquid Crystal Display
MAC	Multiply and Accumulate
MB	Megabytes
PCA	Principal Component Analysis
RGB	Red,Green,Blue
RTS	Runtime Support
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
VENC	Video Encoder
VPBE	Video Processing Back End
VPFE	Video Processing Front End
VPSS	Video Processing Subsystem

1. INTRODUCTION

From the time the first photograph was recorded by Joseph Niepce in 1826, 2D images got enormous usage areas. However, in today's technology they are becoming inadequate and the need for 3D data increases. Especially for the entertainment industry, generating realistic virtual environments is the main target. Three-dimensional cinematography, 3D televisions and game consoles are the examples of today's state of the art. As a result of technological progress in this direction, the importance of reliable three-dimensional data acquisition increases.

Devices used to capture three-dimensional data are generally named as range scanners. Till now, a variety of scanner systems have been developed [1-4]. These can be classified into two groups as: contact and non-contact scanners. Contact 3D scanners probe the subject through physical touch. Although these systems are very precise, they have a disadvantage. They have to contact the object being scanned. During this process, probes may also cause damage to the surface of the object. Therefore, non-contact scanners are more preferable. Among these, structured light based range scanners are the most promising ones.

Principally, structured light based range scanners project a set of coded patterns onto the object. These patterns are deformed based on object geometry. By capturing and processing the deformation on the pattern shape, the 3D range data is obtained. The pattern projection and image capturing operations in these scanners are highly effected by the surface reflectance and external illumination conditions. Especially, scanning shiny surfaces under ambient light conditions is a difficult task for these scanners. Highlights that occur on the shiny object surface based on illumination conditions cause wrong pattern decoding. This results in erroneous and noisy range data extraction.

In this PhD study, we focused on this problem and proposed a solution using color information. By robust segmentation of color without being effected by the illumination, we can make the correct pattern decoding. For this purpose, we proposed using color invariants. Color invariants help segmenting colors in an image without being effected by

lighting, shadow, and highlights. Using color information has another advantage on pattern coding. Depending on the number of colors, the codification can be made with less number of patterns. This means, we can have a faster scanning system. This is a requirement especially for applications such as 3D face recognition for security applications. Therefore, our second objective in this study is implementing a fast scanner system by segmenting as many colors as possible using color invariants. According to these objectives, we designed five different structured light based range scanners using color invariants. We implemented these in a general setup that gives the user the ability to choose the scanner that meets the needs of his or her application. To have a fast and stand alone system, we implemented our scanner system on a digital signal processor.

In the following chapters, we first give details of structured light scanner systems. Then, we give a brief explanation of real time systems that are used in structured light range scanners. The next subject we will cover is the setup we designed to implement our range scanners. Then, we focus on the details of color invariants and their usage in structured light scanning. We explain each scanner method and the experiments based on color invariants for these. The next chapter explains the embedded system implementation of the scanner. We introduce the hardware capabilities and the software written for the scanner system. Then, we evaluate the overall performance of the system by the range data obtained, their timings, and accuracy. Before concluding our study, we provide the three-dimensional object models generated using our range scanner system.

2. STRUCTURED LIGHT BASED RANGE SCANNERS

With the ability to scan the object surface without touching it, structured light based range scanners are commonly used for most applications. They work on a principle of projecting one or more light stripes onto the object surface by a light source (laser, projector, etc.). The stripe produces a line of illumination that appears deformed from other perspectives than that of the projector. This deformation is used to extract the three-dimensional range data of the object surface. Therefore, the basic setup of these scanners consists of a projector and a camera to capture the deformed stripe image. A schematic of the basic structured light based range scanner setup is shown in Figure 2.1.

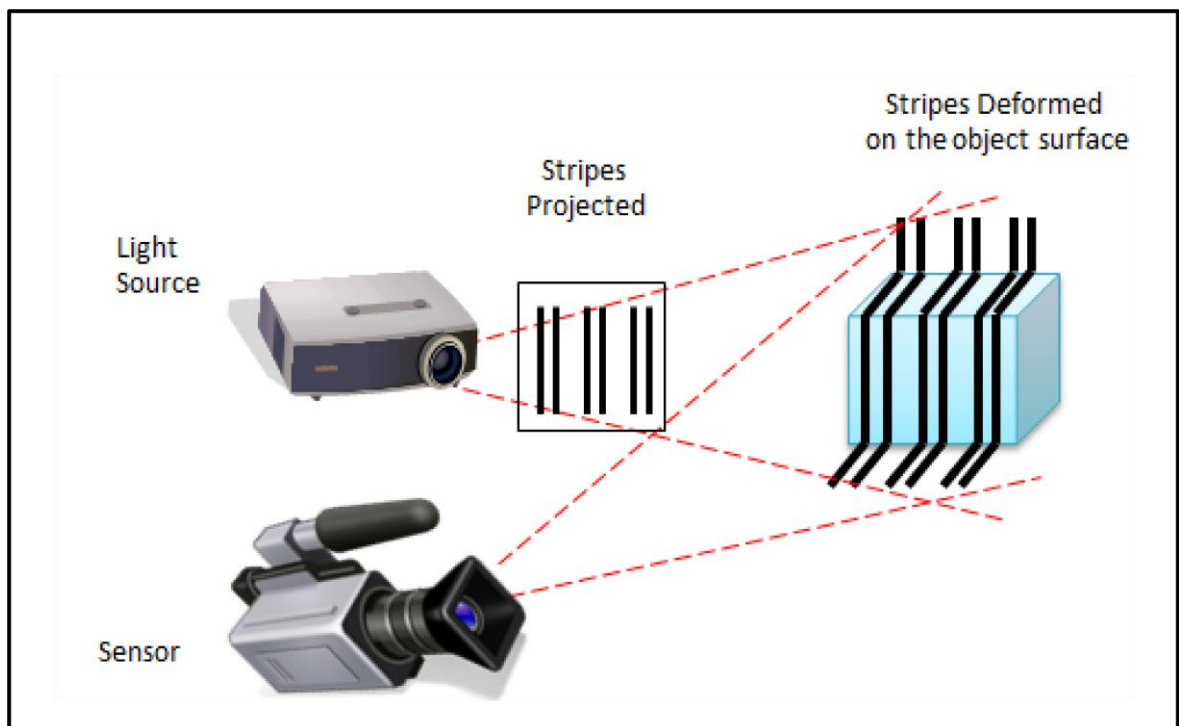


Figure 2.1. Schematic of a structured light system

One handicap of this method is matching the projected stripe with the one in the captured image. This is called the *correspondence problem* of structured light based range scanners. In a single stripe projecting system, it's relatively easy to match the stripe with a robust stripe segmentation. However, for multiple stripe based systems this becomes a

serious problem. Next, we give a brief explanation of a single stripe and multiple stripe based structured light scanner systems. We explain their working principles and the proposed solutions for the correspondence problem. Then, we explain problems of scanning shiny surfaces. We will discuss the proposed solutions in the literature. We also explain our solution in detail in the following chapters.

2.1. SINGLE STRIPE BASED SCANNERS

The single stripe based systems project a stripe onto the object by a light source (projection device or a line laser). Then a camera captures the stripe projected object image. In Figure 2.2 a sample of these systems is shown.

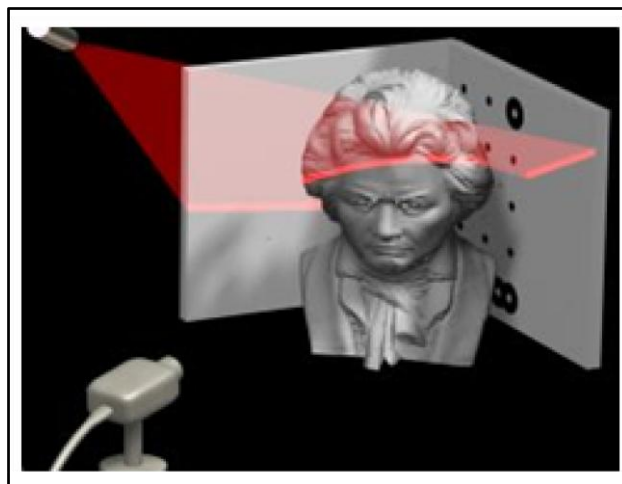


Figure 2.2. Example of a single stripe scanning system

Since a powerful light source projects the stripe, it is easily segmented by thresholding the intensity of the captured grayscale image. If the stripe is segmented robustly and the system is calibrated, the deformation information can be easily converted to the 3D range data [5-7]. In order to scan the entire object, the projected stripe should be shifted through the object surface. Therefore, single stripe based scanners are slow. However, as they give high resolution range data, they are still used in many applications. For faster scanning, multiple stripe based systems are proposed in the literature. Next, we give a brief explanation of these systems.

2.2. MULTIPLE STRIPE BASED SCANNERS

Multiple stripe based scanners work with the same principle as in single stripe scanners, as they project stripes by a projection device and compute the 3D range data from the deformation of the stripes from the captured image. As they project more than one stripe, they can extract the range data of the entire object faster. The correspondence problem occurs here distinctively, since each segmented stripe should be matched with the projected one correctly in order to calculate their deformation. A commonly accepted solution for this problem is time-multiplexed coding. Time-multiplexing works on a principle of projecting a set of patterns successively onto the object surface. The codeword of each pixel is formed by the sequence of illumination values for that pixel across the projected patterns. As the bits of the codewords are multiplexed in time, the codification is named as time-multiplexing. There are several techniques based on time-multiplexing. Here we will explain the binary and Gray coding technique since we use these in our scanner systems. The reader can access the details of other techniques in the survey paper by Salvi *et.al.* [1,2].

2.2.1. Binary Coding

Binary coding is the projection of a sequence of n patterns to encode 2^n stripes using a plain binary code. Here, there are two illumination levels corresponding to “0” and “1”. In commonly used systems, black and white colored patterns are projected. The white illuminated parts are coded as “1” and the black pixels are coded as “0”. At the end of the sequence, each pixel has its own codeword. Figure 2.3 shows the three levels of binary patterns to code eight stripes. The codification based on these stripes is given in Table 2.1.

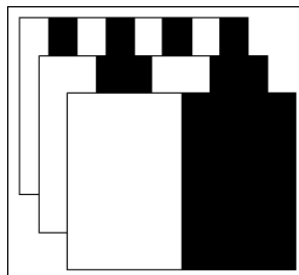


Figure 2.3. Example of three level binary patterns

Table 2.1. Binary coding example for eight stripes

Time	Pattern
I	01010101
II	00110011
III	00001111
Code	01234567

2.2.2. Gray Coding

In binary coding, ambiguities may occur on the crossing edges of the successive patterns. Because of the hardware limitations, there may be some shifts on the crossing edges of the patterns. These may cause wrong codification and noise in the range data. To prevent this problem, Inokuchi *et.al* [8] proposed to use the Gray code instead of plain binary coding. In Gray coding, the consecutive patterns have Hamming distance of one. Therefore, the successive patterns do not contain any crossing edges to cause any ambiguity. The example patterns of Gray code for eight stripes is given in Figure 2.4. The corresponding codification is given in Table 2.

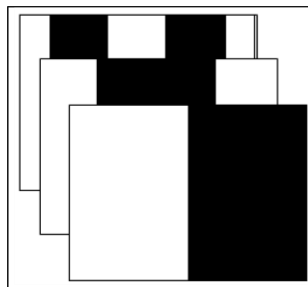


Figure 2.4. Example of three level Gray coded patterns

Table 2.2. Gray coding example for eight stripes

Time	Pattern
I	01100110
II	00111100
III	00001111
Code	01326754

2.2.3. N-ary Gray Coding

The drawback of binary coded structured light based range scanners is the need for large number of patterns to be projected. This slows down the scanning process and increases the computation load. To reduce the number of patterns to be projected, multi level Gray coding is proposed in the literature. For binary coding, for 2^n number of stripes, n number of patterns are needed. However, in Nary coding m^n stripes can be coded with the same number of patterns. Caspi *et. al.* [9] was the first to introduce multilevel Gray coding using color. The Nary Gray code constitutes m number of symbols, each associated with a color. The Nary Gray code is similar with the binary Gray code that has a Hamming distance of one on each sequence to prevent the ambiguity. Its main advantage is the ability of coding with less number of patterns. In our scanner systems, we also used this advantage to reduce the number of patterns to be projected.

2.3. PROBLEM OF SCANNING SHINY SURFACES

Single stripe and binary coded structured light based range scanners are widely used in various applications. However, they have a restriction on scanning objects having shiny surfaces. These reflective surfaces have highlights depending on the ambient light in the environment and the strong projector light directed to the object. These highlights affect the intensity values of the grabbed image. This leads to noisy stripe segmentation and wrong pattern decoding. Hence, the range data extracted becomes corrupted. This is a severe restriction, since applications such as outdoor scanning cannot avoid this type of problem. This problem is reported in various studies in the literature [10-15]. In Figure 2.5,

some examples on problems occurring during the stripe segmentation based on highlights are given.

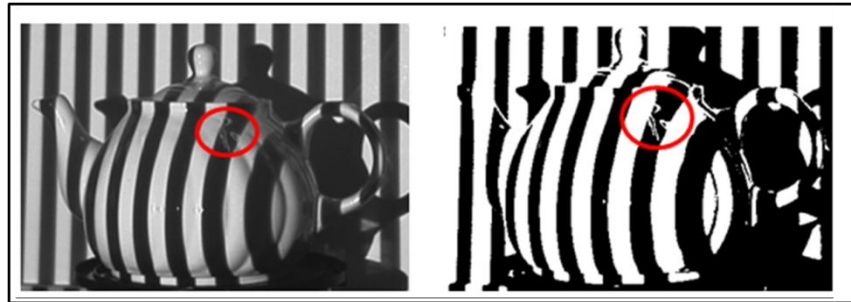


Figure 2.5. Example of highlight problems on a shiny object surface

One may think that scanning shiny objects can be performed under different constraints without any problem. A simple solution may be changing the reflectance property of the shiny surface either by painting it or by coating the surface with powder. This may not be feasible for most applications. One typical example emerges while scanning archeological findings. It may not be possible to paint them. Coating the surface may not be an option for others as well. Another solution may be scanning the object under dark. This may not be possible for some operations such as outdoor object scanning. Here, one may not control the illumination level on the object. Worse, the object may not be moved to a darker region. As in robotics applications, the illumination level may also change during the scanning operation. Most commercial range scanners use a specific filter passing only a specific color band. This solution is not always working properly. Besides, the color information (important for texture mapping) of the scanned object is lost at the end of this operation. To handle all these problems, a robust range scanning system is needed that can work under different ambient illumination levels.

2.4. SOLUTIONS PROPOSED IN THE LITERATURE

Several methods are proposed in the literature to solve this problem. Umasuthan and Wallace [16] tried to solve the problem from the obtained range data. They used a least squares estimator to remove outliers in the range data. Elgazzar *et al.* [17] developed a specific laser stripe based range sensor for indoor environment scanning. They modified

the lens of the camera with a mask in front of it. They counted the insensitivity to ambient light as one of the advantages of this setup. Levoy *et al.* [18] tried different lighting conditions to decrease the effect of laser stripe scattering. Singhal *et al.* [19] introduced a technique to eliminate spurious range values using two (or more) cameras and several consistency tests. Forest *et al.* [20] proposed an FIR filter approach to locate the laser stripe on different surface types. Koninckx and Van Gool [21] proposed an adaptive range finder. Skocaj and Leonardis [22] proposed a method based on changing the intensity of the light projector. Trobina [23] approached the problem from the error model perspective. Zhang and Yau [24] proposed a multiple fringe projection based method to scan object surfaces having high reflectance range. Xu and Aliaga [25] recently proposed a method specifically to overcome strong interreflections. The common drawback for these methods is the need for extra or enormous number of patterns to be projected onto the object to be scanned. This naturally slows down the range scanning process. Besides, as the authors mentioned, some of these methods still suffer from shiny surfaces.

2.5. OUR SOLUTION FOR SCANNING SHINY SURFACES

Since the illumination of the environment is the main effect of the shiny surface scanning problem, we should use a method that is robust to illumination effects. Therefore, we proposed to use color information for stripe segmentation. If we project colored stripes and can acquire the color data clearly, we can segment the stripes robustly. As mentioned above, by color coding we can also decrease the number of patterns to be projected and have a faster system. There are several structured light methods based on color information [9,26,27]. However, the main problem is separating the color without being effected by the illumination. Therefore, we proposed to use color invariants for this purpose. Color invariants help extraction of color information from an image without being effected by the environmental effects such as surface properties, illumination, highlights, and shadows. They were initially used for object recognition and content based image registration applications by Gevers and Smeulders [28-30]. We are the first in using these for range scanner applications. The detailed explanation of color invariants and the implementation for structured light range scanners based on these are given in Chapter 4.

3. HARDWARE OF THE DEVELOPED RANGE SCANNER SYSTEM

In the following chapters, we will explain the developed range scanner system in detail. Here, we review the general characteristics of it. First, we explain our hardware setup. Then we give a brief overview on stereo image geometry and triangulation principle. Then, we explain the used calibration method and its implementation. Finally, we give a brief overview on embedded processors, since we implemented our system on these.

3.1. HARDWARE SETUP

One of our objectives is to implement our range scanner system on an embedded processor. Although our novel single stripe binary, ternary, and quaternary range scanners have different pattern coding and decoding properties, their basic hardware setup is the same. Some portions of their software are also the same. Therefore, in this section, we explore these common properties in the same framework. The scanner system we implemented consists of four main parts; projection device, camera, DSP board, and the rotary table. We designed our setup to output the point cloud in terms of range data. Then, we process this data in computer to obtain the 3D object surface model. We provide the systematic representation of our system in Figure 3.1. Next, we give the details of the range scanner system we have implemented.

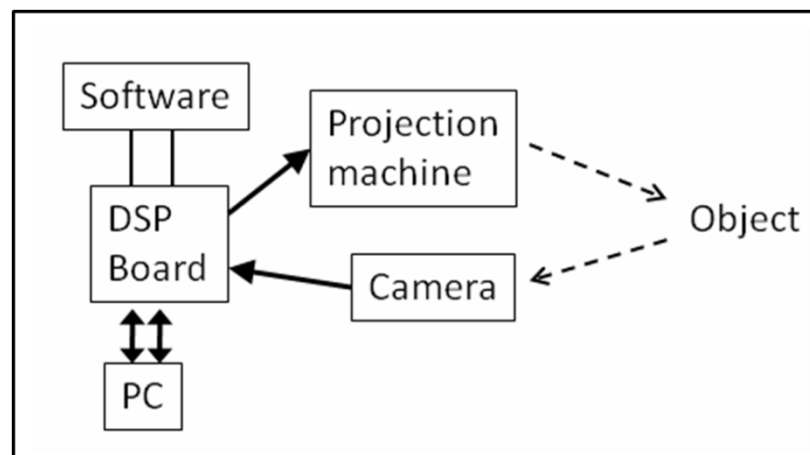


Figure 3.1. System layout

We need a projection device that is stable such that it can project the same color with the same intensity value on each operation. In the DLP projection devices, there is a disc rotating with a constant speed in front of the DLP chip. Without a triggering circuitry, it's not possible to capture accurate colors by this device since the camera cannot synchronize with the rotary disc. To reduce the system complexity, we used a Hitachi CP-X3010Z Multimedia 3LCD projection device to project the patterns in our scanner system. This is a high performance projector that gives constant and accurate color output.

To capture the pattern projected images, we used a Sony DXC-390P 3CCD camera. It's a standard definition 1/3" 3CCD 800 TV lines camera. It can give composite, S-video and RGB component outputs. It has a control output for digital zoom, auto focus, and auto iris applications. For our application, we used an 8 mm Fujinon C-mount lens. It is achromatic, eliminating chromatic aberrations especially occurring on the sharp edges of high intensity color images.

Texas Instruments DM6437 EVM board is the main processing unit of our scanner system. We will give the details of this board in the following chapters. The EVM board contains S-video, composite and component RGB outputs. For the best color quality, we connect it to the projection device by component outputs. Unfortunately, the EVM board contains only composite and S-video inputs. Therefore, we connect the camera to the board by S-video input. This causes sensor cross talk effects on the captured image. The control and data exchange with the the TI DM6437 EVM and computer is made by USB JTAG programming and debugging interface.

To rotate the target object (to be scanned) in a desired angle, we implemented a rotating table system based on the TI MSP430 microcontroller. We will give the details of this rotary table system in the following chapters. Basically, the microcontroller of the rotary table is connected to the EVM board's general purpose input output (GPIO) ports. According to the signal given by the EVM, the microcontroller drives the stepper motor circuitry to rotate the object to be scanned. The rotary table is placed approximately 80 cm to 1 m away from the camera and projector device setup. We provide the images of our scanner system in Figure 3.2.

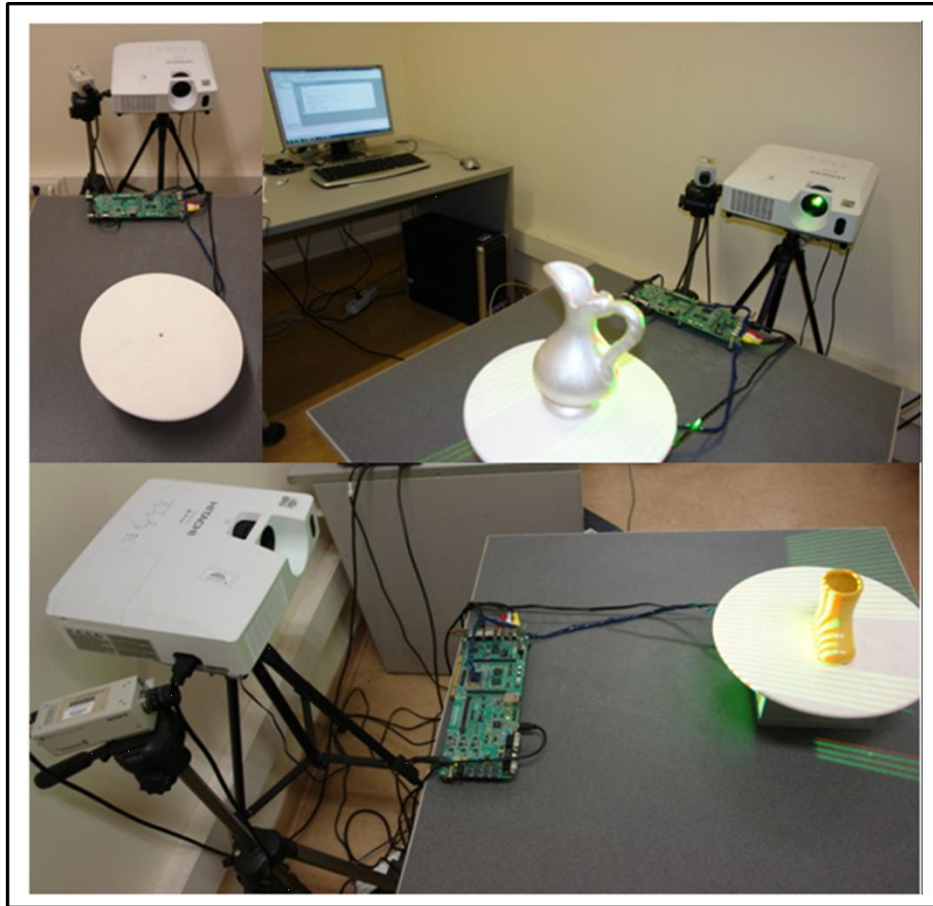


Figure 3.2. Our range scanner from different viewpoints

3.2. THE GEOMETRIC MODEL OF THE CAMERA

The basic camera is a box with a small opening that projects the incoming light to the capturing surface. Therefore, we can model the camera with a pinhole model. If the pinhole was a single point exactly, one light ray would pass through each point in the image plane. However, the pinhole has a finite size. Even more, real cameras are equipped with lenses. Although it is not fully realistic, the pinhole model projection (central perspective projection) is accepted since it is mathematically convenient. Despite its simplicity, it often provides an acceptable approximation of the imaging process. Perspective projection creates inverted images. However, it is more convenient to consider a virtual image on an image plane at the same distance with the pinhole as the actual plane. This image is not inverted, but equivalent to the inverted one. Figure 3.3 shows an example

of a point P in real world and its projection p on the virtual plane. The details of other projection methods can be found in [31].

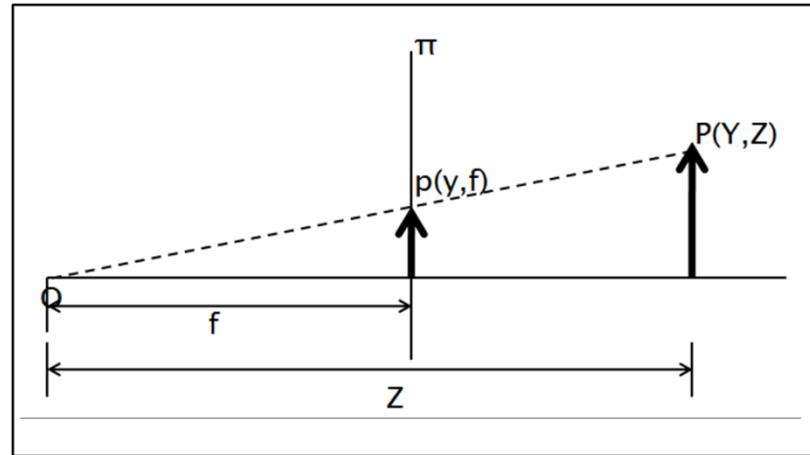


Figure 3.3. Perspective projection of a point P

As we introduced the fundamental model of perspective projection, we can show the constraints between the image measurements and the position of geometric objects in an arbitrary external coordinate system. We will introduce various physical parameters (intrinsic and extrinsic) relating the real world and camera coordinate frames and the general form of perspective projection equation in this setup.

3.2.1. Intrinsic and Extrinsic Camera Parameters

The relationship between a 3D point P on a model plane and its image projection p is given by

$$s\tilde{p} = \mathbf{A}\mathbf{R}\mathbf{t}\tilde{P} \quad (3.1)$$

where s is an arbitrary scale factor. $\tilde{p} = [u, v, 1]^T$ is the augmented vector of 2D point.

$\tilde{P} = [X, Y, Z, 1]^T$ is the augmented vector of the 3D point. The relation between the real world and the camera coordinate system is given by extrinsic parameters, denoted by (\mathbf{R}, \mathbf{t}) as rotation and translation. The camera intrinsic parameters \mathbf{A} are given by

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

where α and β are the scale factors in the image. u and v represent the axis and γ is the skewness parameter of the two image axes. We assume the model plane is placed on $Z = 0$ of the world coordinate system. This yields the point on model plane to be $P = [X, Y]^T$ and the augmented vector to be $\tilde{P} = [X, Y, 1]^T$. If we denote each column of rotation matrix \mathbf{R} by \mathbf{r}_i , then Eqn. (3.1 becomes

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (3.3)$$

Therefore, a point on the model plane and its image is related by homography, \mathbf{H}

$$H = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \quad (3.4)$$

Then, Eqn. 3.3 becomes

$$s\tilde{\mathbf{p}} = \mathbf{H}\tilde{\mathbf{P}} \quad (3.5)$$

3.2.2. Camera Calibration

Camera calibration is a necessary step in a vision system to extract real world coordinates from 2D images. Different approaches for camera calibration have been proposed in the literature. One of them was proposed by Heikkila and Silven [32]. They made the calibration by a combination of a pinhole camera and lens distortion model. A well known method, based on the two-stage technique was proposed by Tsai [33]. He aimed an efficient computation of camera external position and orientation relative to object reference coordinate system as well as the effective focal length, radial lens distortion, and image scanning parameters. In this thesis, we used Zhang's, [34,35] approach. He proposed a flexible new technique for camera calibration by viewing a plane from different and unknown orientations. He solved the camera calibration problem by an

analytical solution followed by a nonlinear optimization technique based on the maximum likelihood estimation. We used the toolbox designed based on this method in our scanner system. Next, we give the details of this calibration process in practice.

3.3. SYSTEM CALIBRATION IN PRACTICE

The real world coordinates of the range data is calculated using the intrinsic and the extrinsic parameters of the camera and the projection device. Therefore, these parameters should be obtained first. By calibrating the system, we can obtain these parameters. As a calibration tool, we used the Camera Calibration Toolbox for Matlab that is designed by Bouguet [36]. This toolbox implements a technique similar to Tsai's and Zhang's approaches on the camera calibration stage. It also includes the projector and overall system calibration.

3.3.1. Camera Calibration in Practice

The toolbox is basically designed to obtain the camera calibration. Therefore, we first calibrate the camera. By using calibrated camera values, we can use the same tool to find the projector and overall system parameters. The toolbox needs reference points that the distance between them are known. Therefore, we place a checker board pattern and take the images of it from different angles. Figure 3.4 shows the checkerboard we used during calibration.

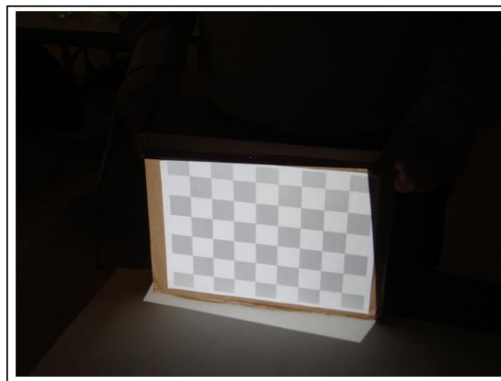


Figure 3.4. The checkerboard used for calibration

In this checkerboard, each square has dimensions of $30\text{mm} \times 30\text{mm}$. To distinguish the original checkerboard points from the projected ones, we used a gray toned pattern. As we mentioned in the previous section, the parameters are calculated iteratively. Therefore, the toolbox needs more than one image to calculate the correct calibration parameters. We take 20 pictures in our calibration procedure by placing the checkerboard pattern approximately 50 cm away from the camera. Figure 3.5 shows an example of checkerboard image capturing scene.



Figure 3.5. An example on image taking for calibration

After taking the checkerboard images, we repeat the same procedure by projecting a checkerboard image from the projection device. Again we take images from different angles in this stage. An example of the projected checkerboard pattern is shown in Figure 3.6.

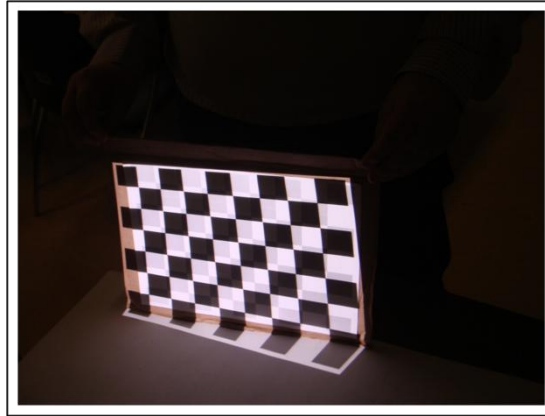


Figure 3.6. An example of checkerboard pattern projection from projection device. After taking images, we use Bouguet's toolbox to calculate the calibration parameters. The primary user interface of Bouguet's toolbox is as shown in Figure 3.7.



Figure 3.7. The user interface of Bouguet's camera calibration toolbox

First, we load and mark the corners of the checkerboards in the camera and projection device images. After marking corners, the toolbox automatically calculates the camera intrinsic and extrinsic parameters. These parameters are used on the next stage to calculate projection device calibration.

3.3.2. Projection Device Calibration in Practice

Projection device can be accepted as an inverse camera. By calibrating the camera in the first stage, we know the real world coordinates of the checkerboard image projected by the projection device since we also know the dimensions of the checkerboard image given to the projection device. The toolbox can calculate the intrinsic and extrinsic parameters of this device. On the second stage of the calibration process, the toolbox calculates these parameters automatically. The final stage is calibrating the overall system.

3.3.3. Overall System Calibration

Overall system calibration is needed to calculate the relative positions of the camera and the projection device to make the triangulation on the range data extraction. As a final stage, the toolbox calculates the relative positions of the camera and the projection device. As a result of calibration, we obtain the extrinsic parameters that give the relative positions of the camera and projection device, intrinsic parameters of the camera and intrinsic parameters of the projection device. We place these parameters to a header file named “*calibration header.h*” to be used by the software of the DSP system. Next, we give the details of the triangulation process.

3.4. 3D POINT CALCULATION BASED ON TRIANGULATION

Structured light scanners work on defining the correspondence of each projected pixel with the camera captured one by using the data based pattern decoding. As each stripe is coded with a unique number from the decoded image, we can extract the horizontal location of the corresponding projected pixel. The vertical correspondence cannot be calculated directly from decodification. Next, we explain the per pixel correspondence calculation based on triangulation that provides us the real world coordinates of the scanned object.

3.4.1. Three Dimensional Point Cloud Extraction

As explained above, we obtain the calibration data of our system once. This data provides the intrinsic and extrinsic parameters that will be used in triangulation. The 3D coordinates of a point \mathbf{P} in the scene may be computed from its pixel coordinates u_c on the camera image and its projector coordinate u_p . In the triangulation operation, the intersection of two rays from the optical centers of the projector C_p and camera C_c gives the 3D coordinates based on per pixel disparity in which the depth is computed.

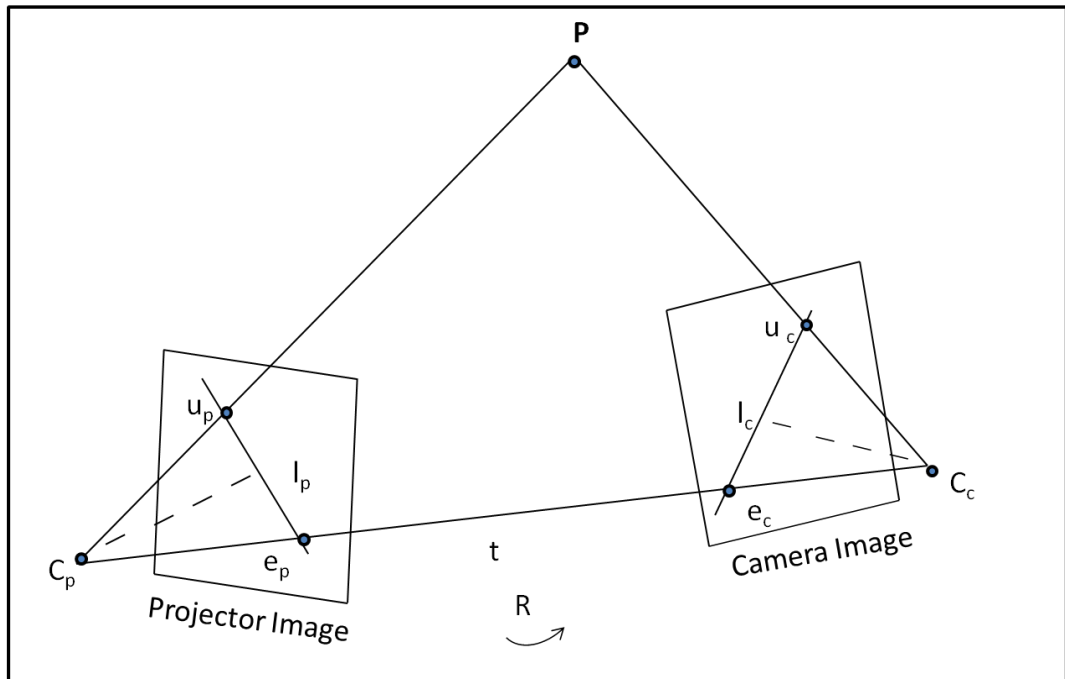


Figure 3.8. Triangulation based on epipolar geometry.

This plane intersects the image planes in the epipolar lines l_p and l_c . The ray $C_p\mathbf{P}$ represents all the possible positions of the point \mathbf{P} for the projection image plane. This is also projected on the epipolar line of the camera l_c . The coordinate system of the projector image plane can be transformed to the camera image plane by a translation t and rotation R from the projector optical center C_p and to the camera center C_c . If K_c and K_p are the calibration matrices of the camera and projector device, the left and right projection of the of the point \mathbf{P} is

$$u_p; [K_p | 0] \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} = K_p \mathbf{P} \quad (3.6)$$

$$u_c; [K_c R | -K_c R t] \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} = K_c (R\mathbf{P} - Rt) \quad (3.7)$$

Using the co-planarity we can write,

$$(K_p^{-1} u_p)^T (t \times R^{-1} (K_c)^{-1} u_c) = 0 \quad (3.8)$$

This equation is homogeneous with respect to t , so the absolute scale cannot be recovered. It is helpful to replace the vector by matrix multiplication of $S(t)$ created from t as

$$S(t) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad (3.9)$$

We can rewrite Eqn. 8 as

$$u_p^T (K_p^{-1})^T S(t) R^{-1} (K_c)^{-1} u_c = 0 \quad (3.10)$$

The middle part of this equation is expressed as a single matrix named the fundamental matrix \mathbf{F} as

$$u_p^T \mathbf{F} u_c = 0 \quad (3.11)$$

It can be seen that the fundamental matrix \mathbf{F} carries the coordinate information of pair of images from the projector and camera. As we find the correspondence by the pattern decodification, using the per pixel disparity and applying the fundamental matrix, we can calculate the real world coordinates of the object being scanned. More information on triangulation and the epipolar geometry can be found in [5-7].

3.5. EMBEDDED SYSTEMS

In developing our range scanners, we benefit from two different types of embedded systems as microcontrollers and DSP boards. In this section, we give a brief explanation to these. To be compact, we also explain the microprocessors.

3.5.1. Microprocessors

Microprocessor is a silicon chip, designed to perform arithmetic and logic operations through its programs. Typical microprocessor operations include adding, subtracting,

comparing two numbers, and fetching numbers from one memory area to another. These operations are the result of a set of instructions that are part of the microprocessor design. Typical layout of a microprocessor is as in Figure 3.9.

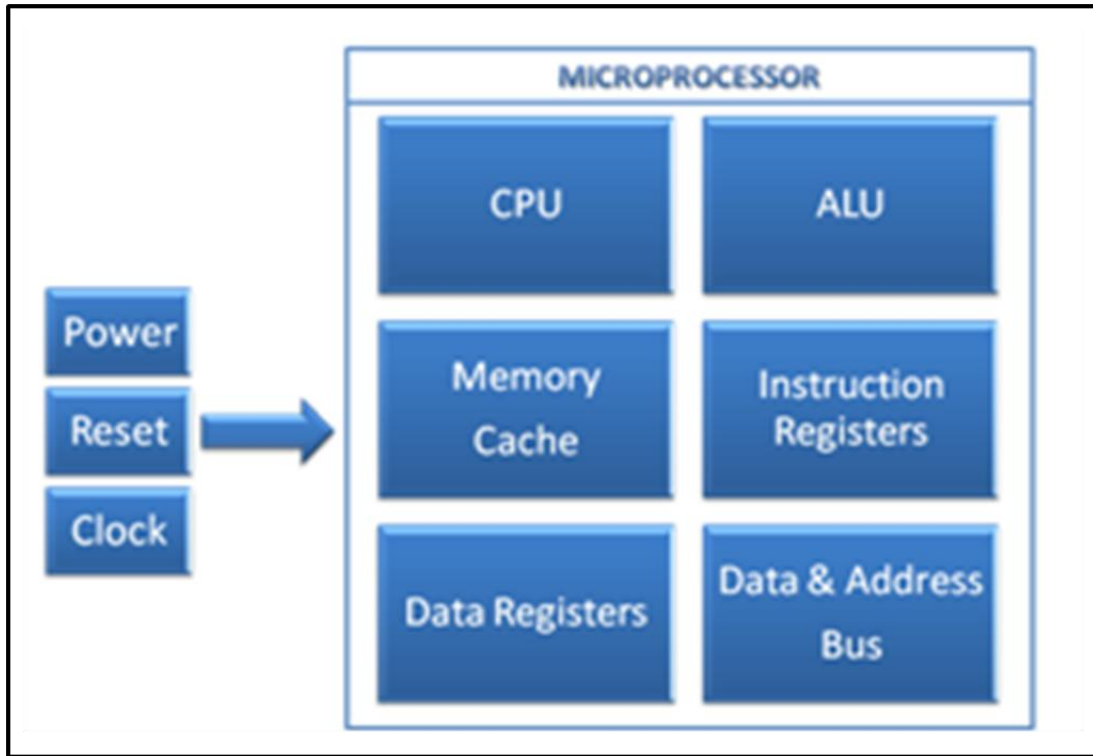


Figure 3.9. Typical layout of a microprocessor

The Central Processing Unit (CPU) is the basic logical structure that accomplishes instructions given by the program. Arithmetic Logic Unit (ALU) is the logical structure that accomplishes arithmetic operations like addition and subtraction. Memory is the structure that the data and the program are stored in. Common microprocessors do not have very large memory space inside the chip. Therefore, the main data and program is stored outside the microprocessor. Instruction registers are small memory blocks. They handle processes that the CPU can execute. The program sets the instruction registers in an order to execute each process required one at a time. The size of the instruction register defines the bit processing capability of the microprocessor.

3.5.2. Microcontrollers

Microprocessors can only perform data manipulation and computation. To interact with the outside world, there should be some peripherals connected to the microprocessor. Microcontrollers are silicon chips that contain both the microprocessor and peripherals. Typical layout of a microcontroller is as in Figure 3.10.

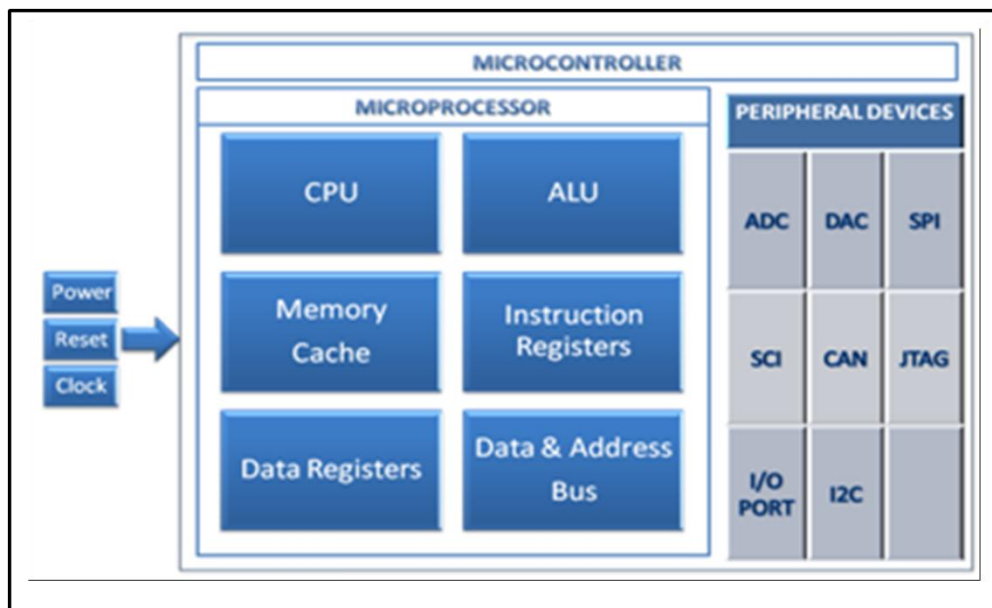


Figure 3.10. layout of a microcontroller

Microcontrollers may contain all or some of the peripherals according to their usage area. For example, to process an audio or a video signal, the microcontroller with an Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC) is needed. Serial Peripheral Interface (SPI), RS232 Serial Communication Interface (SCI), I2C (two wire communication protocol) or JTAG (A fast real-time communication protocol) interfaces will be needed to communicate with another processor or computer. The setup and usage of these peripheral devices are arranged by the control, status, and data registers. For example, to define the sampling rate of the ADC, the necessary bits in the ADC control register are set.

3.5.3. Digital Signal Processors

Digital Signal Processor (DSP) is a microcontroller designed specifically for signal processing applications. This is achieved as follows. Commonly used operations in signal processing applications are convolution, filtering, and frequency-time domain conversions. These need recursive multiplication and additions. In other words, they need multiply and accumulate (MAC) operations. Standard microprocessors execute the multiplication operation as a recursive addition operation. This means for a standard microprocessor, the MAC operation is processed by excessive number of addition operations. This takes time. However, DSPs contain special MAC units that can execute the same operation in a single machine cycle. For example, a 150 MIPS DSP can process approximately 32 million data samples per second. For a standard 150 MIPS microprocessor, this reduces to two million data samples per second. Like microcontrollers, DSPs are equipped with different peripheral devices according to their usage area. Typical layout of a DSP chip is as in Figure 3.11.

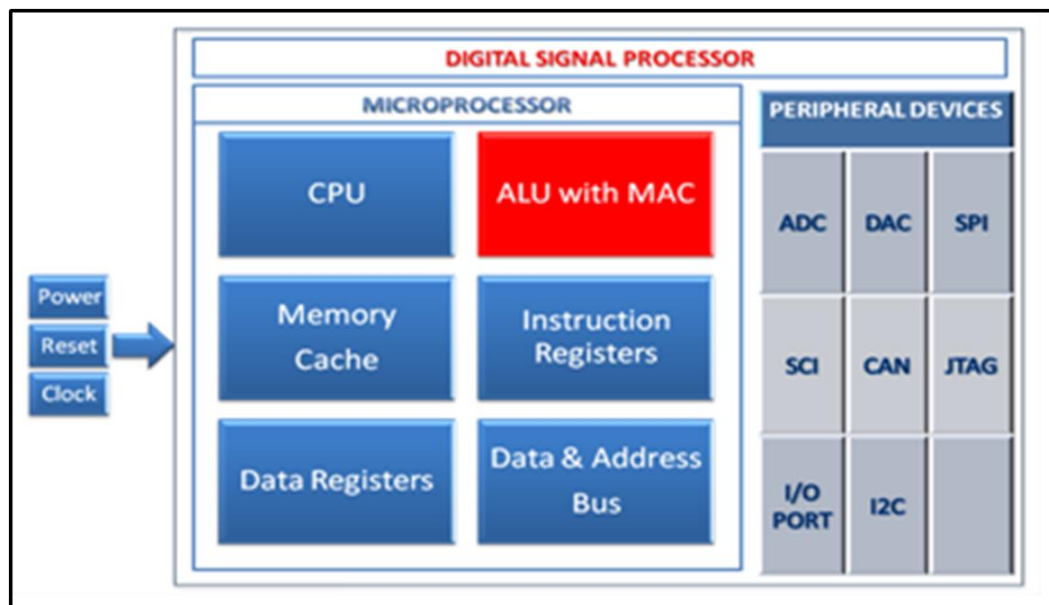


Figure 3.11 Typical layout of a DSP chip

4. RANGE SCANNING USING COLOR INVARIANTS

The main contribution of this thesis is using color invariants in range image scanning. In this chapter, we explore this in detail. Color invariants help extracting color properties of objects without being effected by imaging conditions such as the illumination of the environment, surface properties of the object, the highlights or shadows on the object, and the angle of view. In this thesis, we benefit from these to eliminate the illumination effects in stripe segmentation. Both in single stripe and multiple stripe scanners, we project stripes in different colors. By applying color invariants, we can robustly extract the color information. Therefore, we can segment the stripe without being effected by the illumination conditions.

For the single stripe based and the binary scanner, we project two colors. Therefore, we proposed a color invariant based on two colors (red and green). However, we further implemented ternary and quaternary scanners that use three (red, green, and blue) and four (red, green, blue, cyan) colors respectively. Naturally, our color invariant cannot handle such cases. Therefore, we proposed different color invariants based on Gevers and Smeulders [28-30]. These were initially introduced for object recognition and content based image retrieval problems. In the following sections, we first explain our color invariant proposed for single stripe and binary scanner system. Then, we explain our color invariant based on Gevers and Smeulders' method.

4.1. THE Ψ COLOR INVARIANT

For the single stripe based scanner, we project a red colored stripe on a green background (or a green stripe with red background) image by the projection device. Similarly, for the binary scanner system we project red and green (or blue and green) colored stripes in each pattern for binary codification. For both systems, we need to segment these two colors robustly for clean stripe segmentation and decoding. For this purpose, we proposed a color invariant that can segment out two colors robustly.

The color invariant we proposed is originated from a previous study on multispectral satellite images [37]. There, the Principal Component Analysis (PCA) is applied to the data in order to decorrelate the multispectral image components. Here, we apply the same procedure to the RGB image to obtain a new color invariant (specific for stripe segmentation). By this approach, we aim to suppress the effect of highlights (originating from the ambient light) in stripe detection.

4.1.1. Derivation of the Proposed Color Invariant using PCA

PCA is a methodology for linearly transforming a correlated data set into a new space which has uncorrelated components [38]. The correlated data set is rotated about the origin using a linear transformation matrix to obtain the new space. For a set of correlated random vectors \mathbf{M} , the linear transformation matrix \mathbf{Q} is calculated as follows. Let \mathbf{M} be a $d \times n$ matrix as

$$\mathbf{M} = (m_1, m_2, \dots, m_n) \quad (4.1)$$

where d is the dimensionality of the measurement and n is the number of data vectors (corresponding to the number of observations).

For our application, we use the red and green color band pixel values as observations, hence $d = 2$. To calculate the transformation matrix, \mathbf{Q} , first we obtain the covariance matrix as

$$\mathbf{C}_m = E[(\mathbf{M} - \mu_m)(\mathbf{M} - \mu_m)^T] \quad (4.2)$$

where μ_m is the mean of the sample vector. Next, eigenvectors a_i that satisfy $\mathbf{C}_m a_i = \lambda_i a_i$, $i = 1, 2$ are calculated.

The eigenvalue-eigenvector pairs are indexed such that $\lambda_1 \geq \lambda_2$. We also normalize the eigenvectors such that $\|q_i\| = 1 \forall i$. The value of λ_i corresponds to the spread of the data (with respect to its mean) along the direction of q_i . The transformation matrix \mathbf{Q} is then formed by arranging the eigenvectors, one per row as

$$\mathbf{Q} = [q_1 q_2 \dots q_n]^T \quad (4.3)$$

For our problem, \mathbf{Q} takes the form

$$Q = \begin{bmatrix} -\alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_1 \end{bmatrix} \quad (4.4)$$

where α_1 and α_2 values are obtained from eigenvectors as mentioned above. When this transformation is applied to any data vector m_j , it is projected onto each eigenvector as

$$pr_j = \mathbf{Q}(m_j - \mu_m) \quad (4.5)$$

For our derivations, we shift the principal components by transformed means as

$$pc_j = \mathbf{Q}m_j \quad (4.6)$$

where $pc_j = pr_j + \mathbf{Q}\mu_m$. We therefore work in non-centered spaces.

To define the new color invariant, we transform the color components to an uncorrelated space using the PCA transformation as

$$\begin{bmatrix} pc_1 \\ pc_2 \end{bmatrix} = Q \times \begin{bmatrix} R \\ G \end{bmatrix} \quad (4.7)$$

where R corresponds to the red color band (pixel value) of the image, similarly G corresponds to the green color band (pixel value) of the image. This yields to

$$pc_1 = -\alpha_1 R + \alpha_2 G \quad (4.8)$$

$$pc_2 = \alpha_2 R + \alpha_1 G \quad (4.9)$$

As known, pc_1 and pc_2 are statistically uncorrelated. A slope on them can be defined as

$$S = -\frac{pc_1}{pc_2} = \frac{\alpha_1 R - \alpha_2 G}{\alpha_2 R + \alpha_1 G} \quad (4.10)$$

We take the negative value to emphasize the red band in extracting the color information of the stripe. A normalized angle (having values in ± 1 range) corresponding to this slope is defined as

$$\Psi = \frac{4}{\pi} \arctan\left(\frac{\alpha_1 R - \alpha_2 G}{\alpha_2 R + \alpha_1 G}\right) \quad (4.11)$$

4.1.2. Properties of the Ψ Color Invariant

We take the angle Ψ in Eqn. 4.11 as our color invariant in segmenting the stripe. Next, we explore its properties in depth on matte and shiny object surfaces. To note here, our segmentation method does not depend the object type.

We can justify the usage of the Ψ invariant in our range scanner systems by Caspi *et al.*'s [9] framework. They developed a structured light based range scanner using color codes similar to our system, but without using color invariants. In their study, they analyzed the overall process (from pattern projection to camera image acquisition). As a result, they introduced an equation relating the projected and received color values through projector and camera setup for structured light scanning. The equation proposed by Caspi *et al.* is

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} a_{rr} & a_{rg} & a_{rb} \\ a_{gr} & a_{gg} & a_{gb} \\ a_{br} & a_{bg} & a_{bb} \end{bmatrix} \begin{bmatrix} k_r & 0 & 0 \\ 0 & k_g & 0 \\ 0 & 0 & k_b \end{bmatrix} \begin{bmatrix} r_p \\ g_p \\ b_p \end{bmatrix} + \begin{pmatrix} R_0 \\ G_0 \\ B_0 \end{pmatrix} \quad (4.12)$$

Separating the equation for each color channel (R, G, B), we get

$$R = a_{rr}k_r r_p + a_{rg}k_g g_p + a_{rb}k_b b_p + R_0 \quad (4.13)$$

$$G = a_{gr}k_r r_p + a_{gg}k_g g_p + a_{gb}k_b b_p + G_0 \quad (4.14)$$

$$B = a_{br}k_r r_p + a_{bg}k_g g_p + a_{bb}k_b b_p + B_0 \quad (4.15)$$

where, a values represent projector-camera coupling values. Specifically, a_{rg} , a_{gr} , a_{br} , a_{rb} , a_{bg} , a_{gb} correspond to sensor crosstalk values. k_r , k_g , and k_b are the red, green, and blue reflectance values at the pixel. r_p , g_p , and b_p are the projected color values (after a nonlinear transformation in the projection machine). R_0 , G_0 , and B_0 are the ambient light levels in the environment.

Caspi *et al.* used a 3CCD camera as we do in this study. After tests, they observed that a_{rr} , a_{gg} , a_{bb} values are approximately one. a_{rg} , a_{gr} , a_{br} , a_{rb} , a_{bg} , a_{gb} values are close to zero. We use the same values in this study. In our setup, the ambient light in the environment can be taken as white. Hence, we can define a unique ambient light level as $R_0 = G_0 = B_0 = W$. After these assumptions and replacing the R , G values in Eqn. 4.11 by the values in Eqns. 4.13,4.14,4.15 we obtain

$$\Psi = \frac{4}{\pi} \arctan \left(\frac{-g_p k_g + k_r r_p}{2W + g_p k_g + k_r r_p} \right) \quad (4.16)$$

We will use this equation for the stripe segmentation next.

4.2. STRIPE SEGMENTATION USING THE Ψ COLOR INVARIANT

For the single stripe or binary pattern sequences we use red or green color. Therefore, we examine the response of the invariant according to the red or green color projected pixels. For the projected red pixels, $r_p = 1$ and $g_p = 0$. According to these values, Eqn. 4.16 becomes

$$\Psi = \frac{4}{\pi} \arctan\left(\frac{k_r}{2W+k_r}\right) \quad (4.17)$$

As indicated in Eqn. 4.17, if there is no ambient light, $\Psi = 1$. If there is an ambient light, then $0 \leq \Psi < 1$. Therefore to segment the red stripes, we can threshold the invariant applied image by 0 independent of the reflectance of the object.

Similarly for the projected green pixels, $r_p = 0$ and $g_p = 1$. Then, Eqn. 4.16 becomes

$$\Psi = \frac{4}{\pi} \arctan\left(-\frac{k_g}{2W+k_g}\right) \quad (4.18)$$

In Eqn. 4.18, if there is no ambient light $\Psi = -1$. If there is an ambient light then $-1 \leq \Psi < 0$. As a result, the green pixels can be segmented by thresholding the invariant image for the values less than 0.

The justifications above shows that, our color invariant can be used for both red and green colored line stripes. Besides, for other colored line stripes, this method can be generalized with the same derivation steps. In Figure 4.1, we provide a segmentation example of binary and single stripe scanners of binary and single red stripe image with a green background.

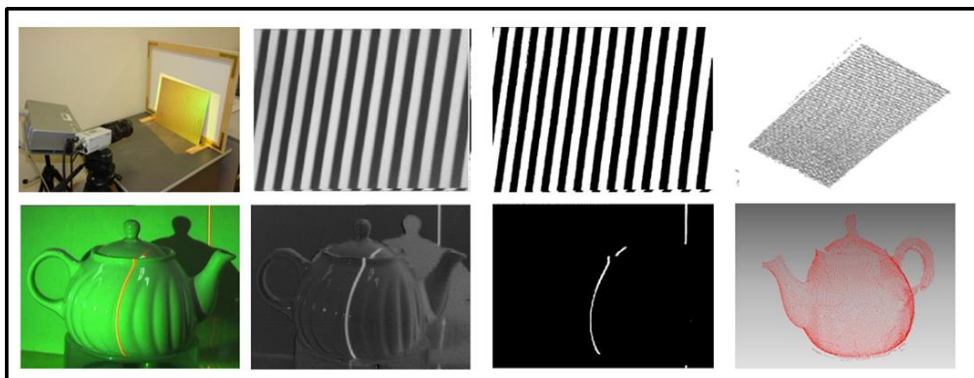


Figure 4.1 Segmentation example for binary and single stripe scanners using ψ

We have published the single stripe based scanner with this color invariant in [39]. An extended version covering both the laser and the projection device system is published in [40]. The implementation of this scanner for the binary coded structured light scanners is published in [41,42].

4.3. THE ‘c’ COLOR INVARIANT SET

The Ψ invariant is successful in segmenting two colors. However, as we mentioned in Section N-ary Gray Coding 2.2.3, to reduce the number of stripes to be projected, we have to increase the number of colors used in the patterns. Naturally, our color invariant can not handle such cases. Therefore, we tried different color invariants. We give the details of the tests on other invariants in Appendix A. Although there are several color invariants in the literature, they are basically introduced for other purposes. After extensive testing, we decided that the ‘c’ color invariant set introduced by Gevers and Smeulders [28] serve our purposes well for our range scanners. To protect the integrity of the scanner system, we will not use the Ψ color invariant further for the binary and single stripe scanner systems. We will use the ‘c’ color invariants and their combination for both binary, ternary, and quaternary scanner systems. These color invariants are

$$c_1 = \arctan\left(\frac{R}{\max\{G,B\}}\right) \quad (4.19)$$

$$c_2 = \arctan\left(\frac{G}{\max\{R,B\}}\right) \quad (4.20)$$

$$c_3 = \arctan\left(\frac{B}{\max\{R,G\}}\right) \quad (4.21)$$

We refer the reader to the mentioned reference for more details on these color invariants. In our range scanner implementations, the color invariants c_1 and c_2 provide similar results. Therefore, we will not deal with the c_2 color invariant further. In the following sections, we benefit from c_1 , c_3 , and their combination in scanning shiny and

matte object surfaces in our range scanner system. We explore the properties of these color invariants in our setup next.

4.3.1. Properties of the ‘c’ Color Invariant Set

As in the previous section, we justify the ‘c’ color invariant set with Caspi *et al.*'s equation that relates the projected and the captured image. By replacing the R , G , B values in Eqns. 4.19, 4.21 by the values in Eqns. 4.13, 4.14, 4.15, we obtain

$$c_1 \approx \arctan\left(\frac{k_r r_p + R_0}{\max\{k_g g_p + G_0, k_b b_p + B_0\}}\right) \quad (4.22)$$

$$c_3 \approx \arctan\left(\frac{k_b b_p + B_0}{\max\{k_r r_p + R_0, k_g g_p + G_0\}}\right) \quad (4.23)$$

In our setup, the ambient light in the environment can be taken as white. Hence, we can define a unique ambient light level as $R_0 = G_0 = B_0 = W$. Using this, we can further simplify Eqns. 4.22 and 4.23 as

$$c_1 \approx \arctan\left(\frac{r_p k_r + W}{\max\{g_p k_g, b_p k_b\} + W}\right) \quad (4.24)$$

$$c_3 \approx \arctan\left(\frac{b_p k_b + W}{\max\{r_p k_r, g_p k_g\} + W}\right) \quad (4.25)$$

These two equations will serve for decoding binary, ternary, and quaternary patterns in the next section.

4.4. DECODING PATTERNS USING ‘c’ COLOR INVARIANTS

In this section, we benefit from the c_1 , c_3 color invariants and their combination to decode binary, ternary, and quaternary patterns projected onto the test object. To decode each pattern, we apply a different method. Therefore, we explore each method separately next.

4.4.1. Decoding Binary Patterns

We implement our binary range scanner in two different ways using c_1 and c_3 color invariants separately. To use the first color invariant, c_1 , we project eight red and green colored patterns (in terms of line stripes with varying widths) onto the object. To decode these projected patterns from the grabbed camera images, we apply thresholding after obtaining their color invariant images. We explain this methodology next.

For the pixels corresponding to the red stripe on the grabbed image, we have $r_p = 1$ and $g_p = 0$. Remember, we do not project the blue color. Therefore, $b_p = 0$ for this scenario. Then, we have

$$c_1 \approx \arctan\left(\frac{k_r+W}{W}\right) \quad (4.26)$$

Similarly for the pixels corresponding to the green stripes, we have $g_p = 1$ and $r_p = 0$. As in the previous derivation, $b_p = 0$. Based on these, we have

$$c_1 \approx \arctan\left(\frac{W}{k_g+W}\right) \quad (4.27)$$

We know that k_g and W values are greater than zero. Therefore, $k_r + W > W$ for any ambient light level (W) and the red reflectance value (k_r). Similarly, $k_g + W > W$ for any ambient light level (W) and the green reflectance value (k_g). These lead to the following conclusion. For the red stripes in the pattern, the term inside the arctangent function in Eqn. 4.26 will be greater than one. Since, the arctangent function is monotonic (within the $[-\pi, \pi]$ range); $c_1 > \arctan(1)$ or $c_1 > 0.8$. Similarly, for the green stripes in the pattern, the term inside the arctangent function in Eqn. 4.27 will be less than one. Therefore, $c_1 \leq 0.8$. Hence, the red and green stripes can be extracted from the c_1 image easily by taking the threshold value of 0.8. As can be seen, the ambient light level W has no effect on selecting this threshold value. The same derivations can be made for the c_3 color invariant. We apply this strategy to all eight red-green colored patterns. Then, we construct the decoded line stripes for each pattern.

In structured light based range scanners, we project patterns onto the whole object surface. Therefore, highlights will be either very bright red or very bright green depending on their location. By the justification above, we can claim that the intensity of the color does not affect the stripe extraction. Therefore, the stripes corresponding to the highlight locations can also be extracted reliably.

We pick the matte and shiny Atatürk objects, as given in Figure 4.2, to show the difficulty in decoding the patterns from shiny object surfaces. We will also use these two test objects in the following sections for comparison. We picked these two objects since they have the same surface properties. Only their reflectance properties change. Therefore, we can have a controlled test environment.



Figure 4.2. Matte and shiny Atatürk objects

We provide the binary pattern decoding example using c_1 on one of the eight patterns for our binary range scanner in Figure 4.3. For the c_1 color invariant, we project red-green colored stripes (pattern) on these test objects. First, we provide the single layer of these coding images. In the same figure, we provide the extracted color invariant images and the decoded patterns (in color coded form) for both objects. As can be seen, binary pattern decoding results on matte and shiny Atatürk objects are fairly good using color coding and the c_1 color invariant.

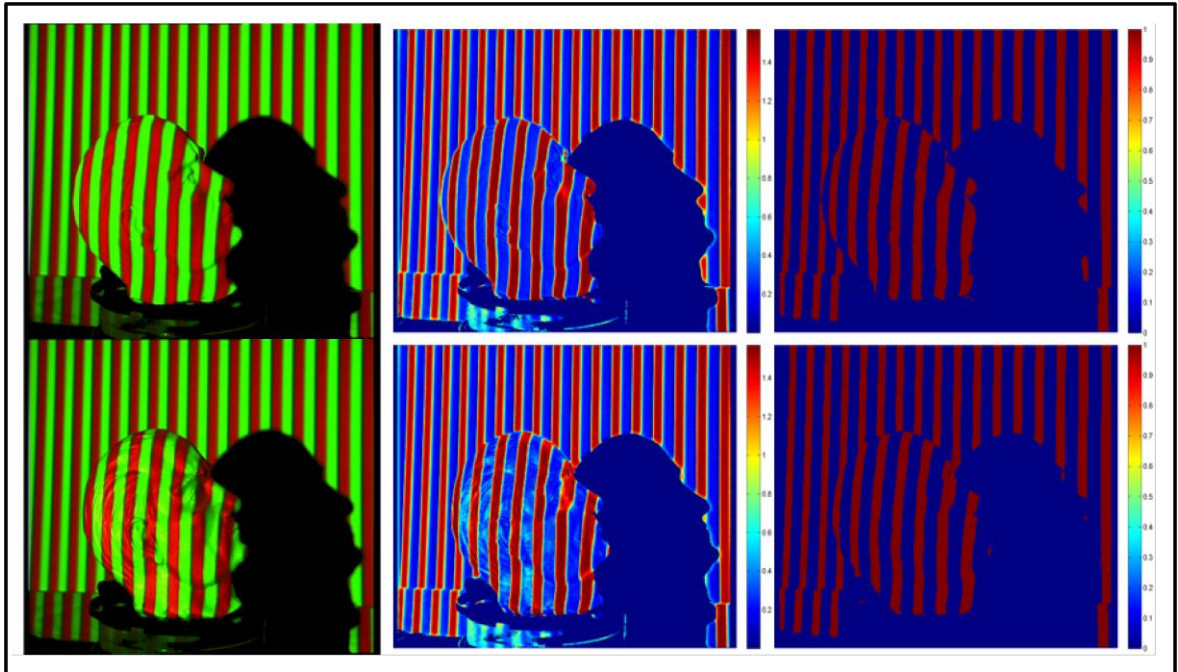


Figure 4.3. Binary pattern decoding results using c_1 . First column: matte and shiny Atatürk. Second column: c_1 color invariant images. Third column: binary patterns decoded (in color coded form)

As the next example, we apply the same procedure using c_3 in Figure 4.4. To use this invariant, we project blue-green colored stripes (pattern) on test objects. As in the previous example, we provide the coded images, color invariant versions, and the decoded patterns (in color coded form) in Figure 4.4. As can be seen, we can decode the single layer of binary patterns from both matte and shiny Atatürk objects in a reliable manner using the c_3 color invariant.

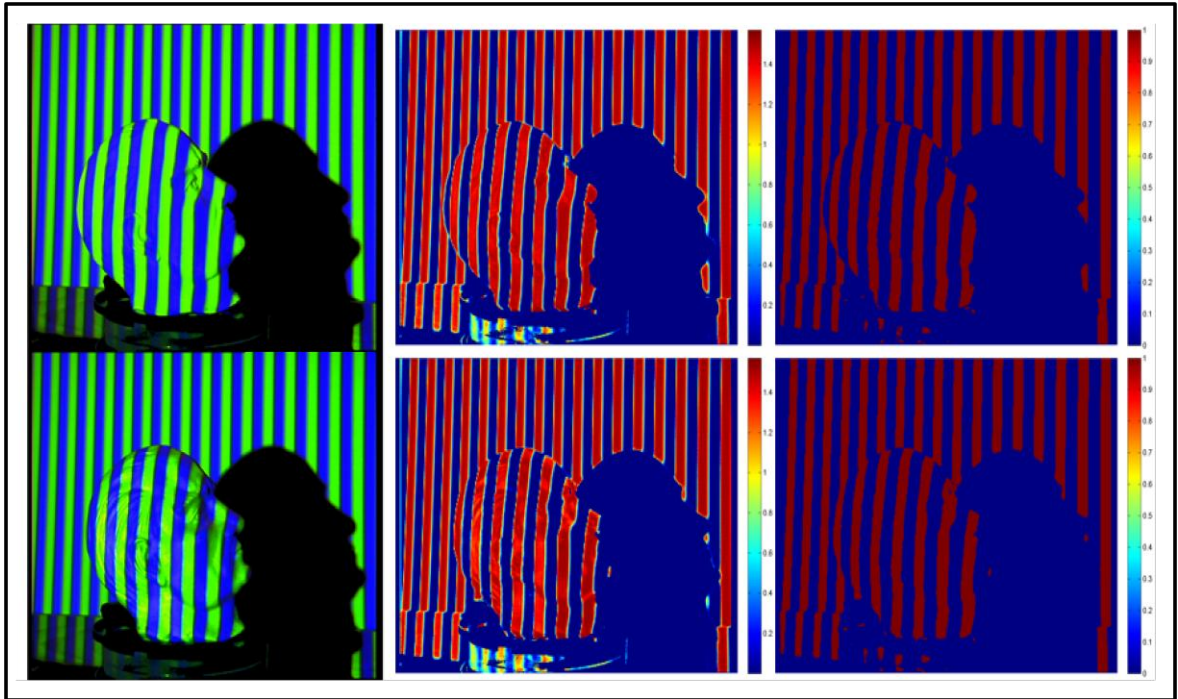


Figure 4.4. Binary pattern decoding results using c_3 . First column: matte and shiny Atatürk. Second column: c_3 color invariant images. Third column: binary patterns decoded (in color coded form)

As a comparison, we also provide the single layer of black and white pattern decoding example for the standard binary range scanner in Figure 4.5. In this figure, we provide the coded images of matte and shiny Atatürk objects and their decoding results. As can be seen, although the decoding results for the matte Atatürk object are fairly good, there are false decodings around the eye and the hair sections of the shiny Atatürk. These regions will lead to false range data after decoding all pattern levels.

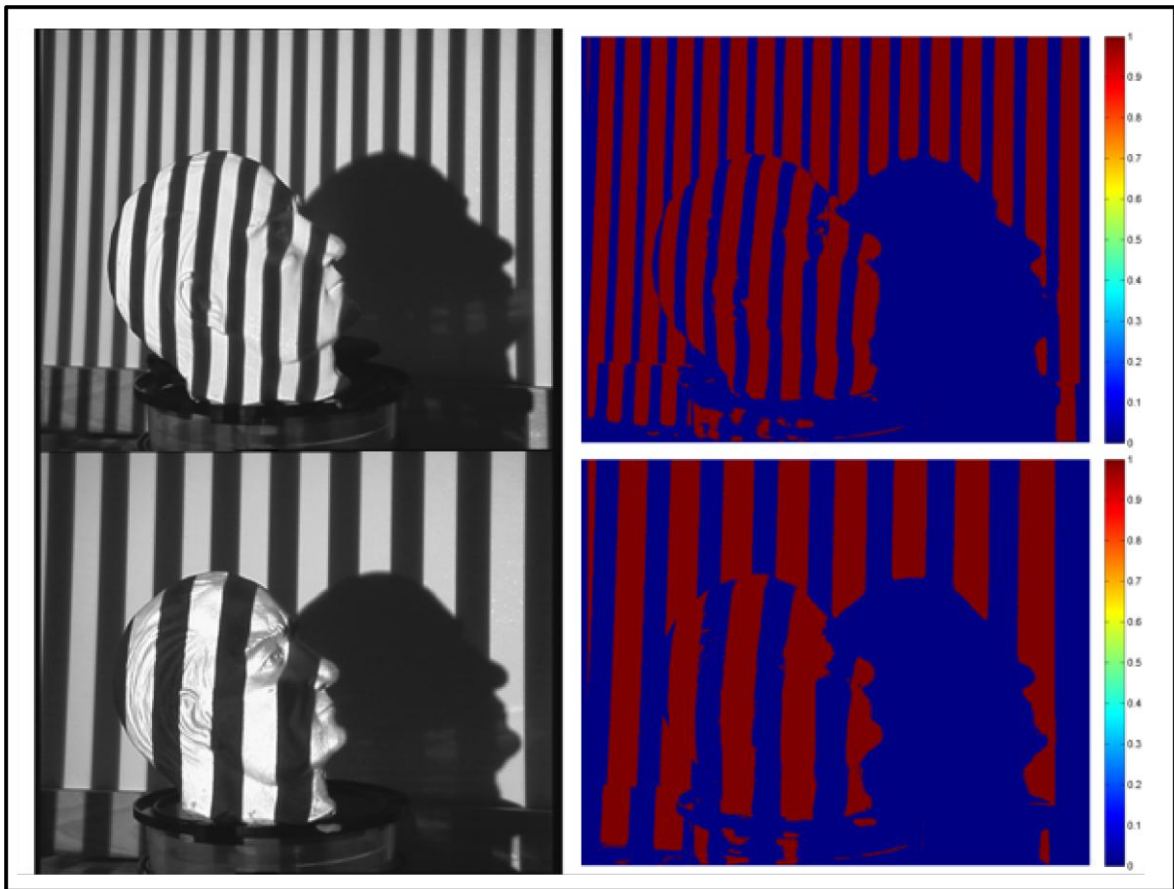


Figure 4.5. Binary pattern decoding results using black and white stripes. First column: matte and shiny Atatürk. Second column: binary patterns decoded (in color coded form)

4.4.2. Decoding Ternary Patterns

For the ternary coded patterns, we use red, green, and blue colors. To decode stripes corresponding to these colors in the patterns projected, we have similar assumptions as in the binary range scanner. For the red colored stripes (in the patterns), we have $r_p = 1$, $g_p = 0$, and $b_p = 0$. Using these values in Eqns. 4.24 and 4.25, we obtain

$$c_1 \approx \arctan\left(\frac{k_r+W}{W}\right) \quad c_3 \approx \arctan\left(\frac{W}{k_r+W}\right) \quad (4.28)$$

Similarly, for the green colored stripes (in the patterns), we have $r_p = 0$, $g_p = 1$, and $b_p = 0$. Using these values in Eqns. 4.24 and 4.25, we obtain

$$c_1 \approx \arctan\left(\frac{W}{k_g+W}\right) \quad c_3 \approx \arctan\left(\frac{W}{k_g+W}\right) \quad (4.29)$$

Finally, for the blue colored stripes (in the patterns), we have $r_p = 0$, $g_p = 0$, and $b_p = 1$. Using these values in Eqns. 4.24 and 4.25, we obtain

$$c_1 \approx \arctan\left(\frac{W}{k_b+W}\right) \quad c_3 \approx \arctan\left(\frac{k_b+W}{W}\right) \quad (4.30)$$

We can see that there is a nice symmetry in Eqns. 4.28, 4.29, and 4.30. As in the binary range scanner, we have $k_r + W > W$, $k_g + W > W$, and $k_b + W > W$ independent of the reflectance values and the ambient light level. Using the symmetry and the mentioned inequalities, we can define a new variable to decode ternary coded patterns as

$$s = c_1 - c_3 \quad (4.31)$$

For the red, green, and blue colored stripes, $s > 0$, $s \approx 0$, and $s < 0$ respectively. Therefore, we can use s to extract the red, green, and blue stripes from the grabbed image. In implementation, we divide the range of the s value to three and apply segmentation based on these. We apply this strategy to all red, green, and blue colored pattern projected object images. Then, we construct the decoded line stripes in the overall image.

As in the previous section, we provide the ternary pattern decoding example using s on one of the five patterns for our ternary range scanner in Figure 4.6. First, we provide the ternary coding images for matte and shiny Atatürk objects. In the same figure, we provide the extracted s images and the decoded patterns (in color coded form) for both objects. As can be seen, ternary pattern decoding results on matte and shiny Atatürk objects are fairly good using color coding and the s value. The binary and ternary scanner implementation using the ‘c’ color invariants is published in [43].

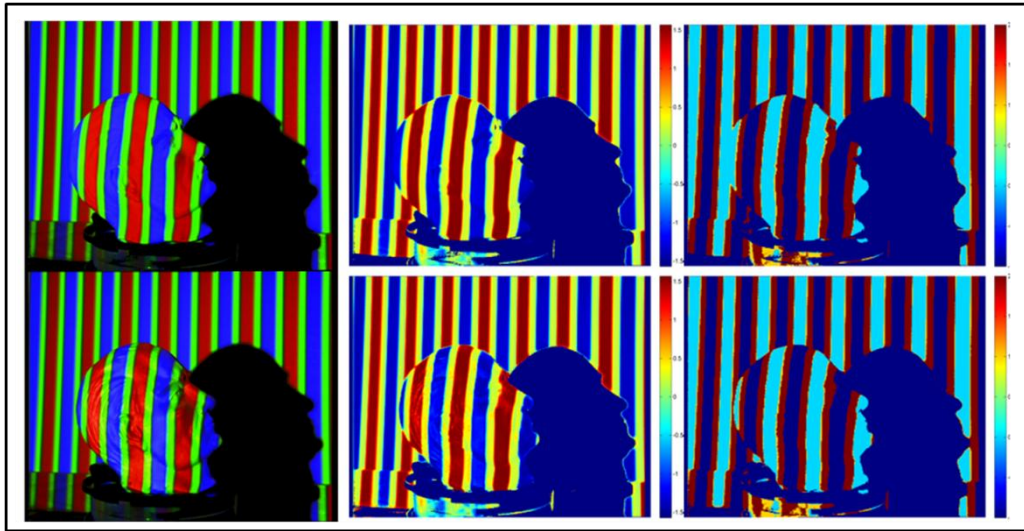


Figure 4.6. Ternary pattern decoding results using s . First column: matte and shiny Atatürk. Second column: s images. Third column: ternary patterns decoded (in color coded form)

4.4.3. Decoding Quaternary Patterns

To decode quaternary patterns, we apply a similar strategy as in the ternary pattern case. Therefore, we again use s in Eqn. 4.31. In quaternary coding, we have red, green, blue, and cyan color coded stripes in the patterns. For the red, green, and blue colored stripes, the derivations are the same as in the ternary coding case. To extract the cyan colored stripes, we have $r_p = 0$, $g_p = 1$, and $b_p = 1$. Using these values in Eqns. 4.24 and 4.25, we obtain

$$c_1 \approx \arctan\left(\frac{W}{k_g+W}\right) \quad (4.32)$$

$$c_3 \approx \arctan\left(\frac{k_b+W}{\max\{k_g, k_b\}+W}\right) \quad (4.33)$$

For the cyan colored stripes, s has a negative value such that $s \approx c_1 - 1$. Although there is no symmetry here, in implementation we experimentally observed that, we can extract four different colored stripes in the quaternary scanner by dividing the s range to four and applying segmentation based on these. We apply this strategy to all four red,

green, blue, and cyan colored patterns. Then, we construct the decoded line stripes in the overall image.

As in the previous section, we provide the quaternary pattern decoding example using s on one of the four patterns for our quaternary range scanner in Figure 4.7. First, we provide the coding images on both matte and shiny Atatürk objects. In the same figure, we provide the extracted s images and the decoded patterns in color coded form for both objects. As can be seen, the quaternary pattern decoding results on matte and shiny Atatürk objects are fairly good using color coding and the s value.

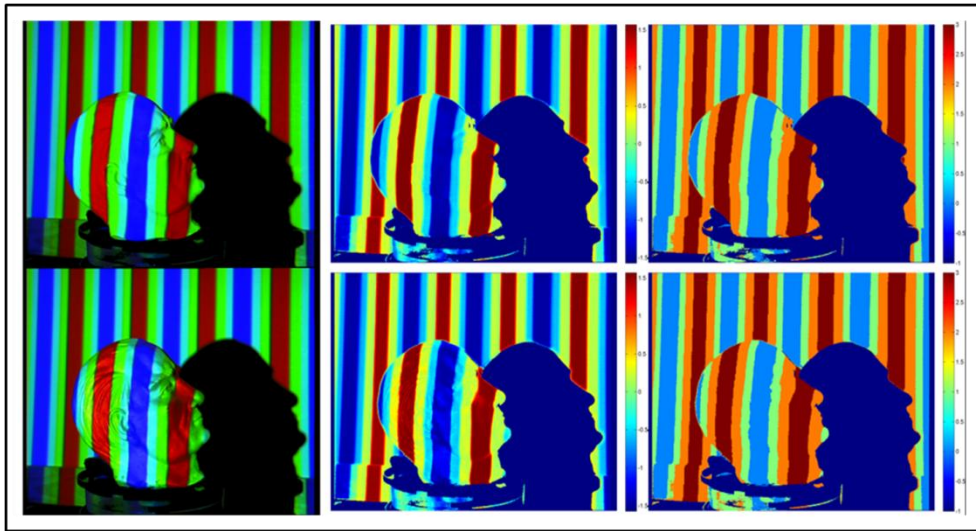


Figure 4.7. Quaternary pattern decoding results using s . First column: matte and shiny Atatürk. Second column: s images. Third column: quaternary patterns decoded (in color coded form)

5. EMBEDDED SYSTEM IMPLEMENTATION

To implement the mentioned range scanner system, we used a TI DM6437 EVM board as the main unit. Our scanner system includes all the methods explained in the previous chapter. Therefore, it gives the user an opportunity to choose the scanner type. Furthermore, our range scanner system can scan the target object from different viewing angles. To do so, we added a rotating table system controlled by the TI DM6437 EVM board. Hence, we can obtain the object model. The single stripe scanner implementation on the embedded system is published in [44]. We will give the details of model construction in the following chapter. Here, we focus on the the details of the hardware part of our scanner system.

5.1. TI DM6437 EVM BOARD PROPERTIES

Structured light coding, decoding, pattern generation, and stripe extraction operations require a powerful computing platform. TI DM6437 EVM board is one of such platforms. It provides high computational power, as well as highly optimized software tools. Therefore, we decided to use it in our implementation. We provide its properties next.

5.1.1. The DSP Platform

TI DM6437 is a high performance, fixed point digital media processor build on C64x+ CPU with clock rates up to 700 MHz. The processor in the TI DM6437 EVM board has 600 MHz clock rate, corresponding to 4800 million instructions per second (MIPS). The DSP subsystem has 32 KB program and 80 KB data level one cache. The 128 KB level two cache provides flexible allocation to be used as RAM or cache. Besides, the internal memory of the TI DM6437 EVM board contains 32 MB NOR and 64 MB NAND flash memories used for boot loading, 2 MB SRAM for application debugging and 2x64 MB DDR2 SDRAM for program, data, and video storage.



Figure 5.1. The DM6437 EVM

5.1.2. Video Input/Output Peripherals

In the TI DM6437 EVM board, video input/output peripherals are managed by the Video Processing Subsystem (VPSS). VPSS involves two configurable video/imaging peripherals: one Video Processing Front-End (VPFE) input used for video capturing, one Video Processing Back-End (VPBE) output having a Video Encoder (VENC). The VENC provides four analog DACs that run at 54 MHz, providing a means for composite NTSC or PAL video, S-video, and/or component video output. VPFE has a CCD Controller (CCDC), a preview engine, histogram module, auto-exposure, white balance, focus module (H3A), and resizer. The CCDC is capable of interfacing to common video decoders, CMOS sensors, and CCDs. On the TI DM6437 EVM board, CCDC is interfaced with a TVP5146M2 video decoder.

The TVP5146M2 is a 10-bit, 30 million samples per second (MSPS) high quality single-chip digital video decoder that digitizes and decodes NTSC, PAL, SECAM, composite, and S-video into component YC_bC_r format [45]. The decoder is configured over the I2C host port interface. According to this configuration, it generates synchronization, blanking, field, active video window, horizontal and vertical syncs, clock, genlock (for downstream video encoder synchronization), host CPU interrupt, programmable logic I/O signals and 4:2:2 YC_bC_r video output signals.

5.1.3. Programming and Debugging Issues

TI has an integrated DSP development environment called Code Composer Studio (CCS). We used CCS V.4 to program and debug the TI DM6437 EVM board and the TI MSP430 microcontroller. A real-time multi-tasking kernel (mini-operating-system) created by TI for the TMS320 family of DSP's named DSP/BIOS is an integrated part of this development platform. The DSP/BIOS includes graphical kernel object viewer and real-time analysis tools specifically focused on debugging and tuning multitasking applications. Through a graphical configuration manager, DSP/BIOS manages device configurations, hardware and software interrupts, memory mappings, CPU and peripheral timings, and data exchange between the evaluation board and the CCS environment. In coordination with the DSP/BIOS, device initializations and controls are implemented through the Chip Support Libraries (CSL) and Board Support Libraries (BSL) by the C language callable functions. To have a fast scanning operation, we used TI's fixed point IQmath and FastRTS libraries in programming.

5.1.4. DSP Configuration for Image Input/Output

The first implementation step on the TI DM6437 EVM board is to setup the image input/output structure. To have an optimized hardware configuration, we used the video preview framework provided by TI Digital Video Software Development Kit (DVSDK). It uses C language callable functions of VPSS to capture and output the image. This framework generates video input and output buffer queues having three times the image size by memory allocation on the DDR2 SDRAM. These queues work in FIFO structure. The image to be processed is called from the video input queue. The processed image is placed to the video output queue. At the same time, a new image is captured and placed in the video input queue. Therefore, the data acquisition and processing runs in parallel. We provide this operation in a systematic layout form in Figure 5.2. We use the advantage of this structure to prevent the time loss in shadow image capturing (to be explained next).

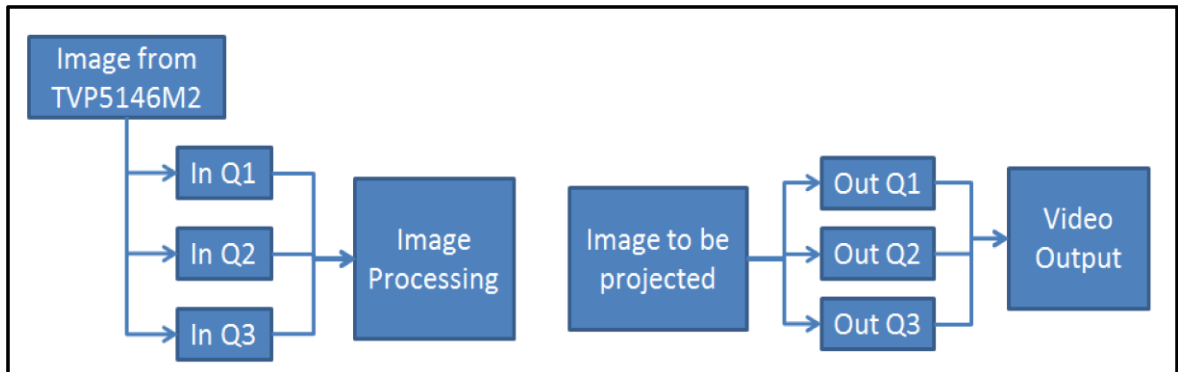


Figure 5.2. The image input output structure

5.1.5. Properties of the Rotating Table

To be able to scan the target object from different viewpoints, we implemented a rotating table system for our scanner setup. By controlling the rotation angle, we can scan the target object from any desired angle. Our rotating table system consists of a step motor, driving circuit and a TI MSP430 microcontroller.

5.1.5.1. Motor driving circuit

We used a step motor in the rotating table system to have an angular control. The step motor rotates in steps according to the magnetic field occurred by the current flowing through different directions on the four coils of the motor. We used an ULN2003 H-Bridge IC to control the current directions on the motor coils. This IC changes the current flow direction according to the states given by the microcontroller. The microcontroller gives eight states on four bits to control the motor on half step driving. We used optocouplers to isolate the microcontroller from the ULN2003 circuit. This way, we can protect the microcontroller from any undesired current flows. We also balanced the voltage levels of the IC and the microcontroller this way. The motor is fed by a 9V-3A DC power supply. The microcontroller circuit is supplied through the TI DM6437 EVM board by a 3.3V DC voltage. The schematic of the motor driving circuit is given in Figure 5.3.

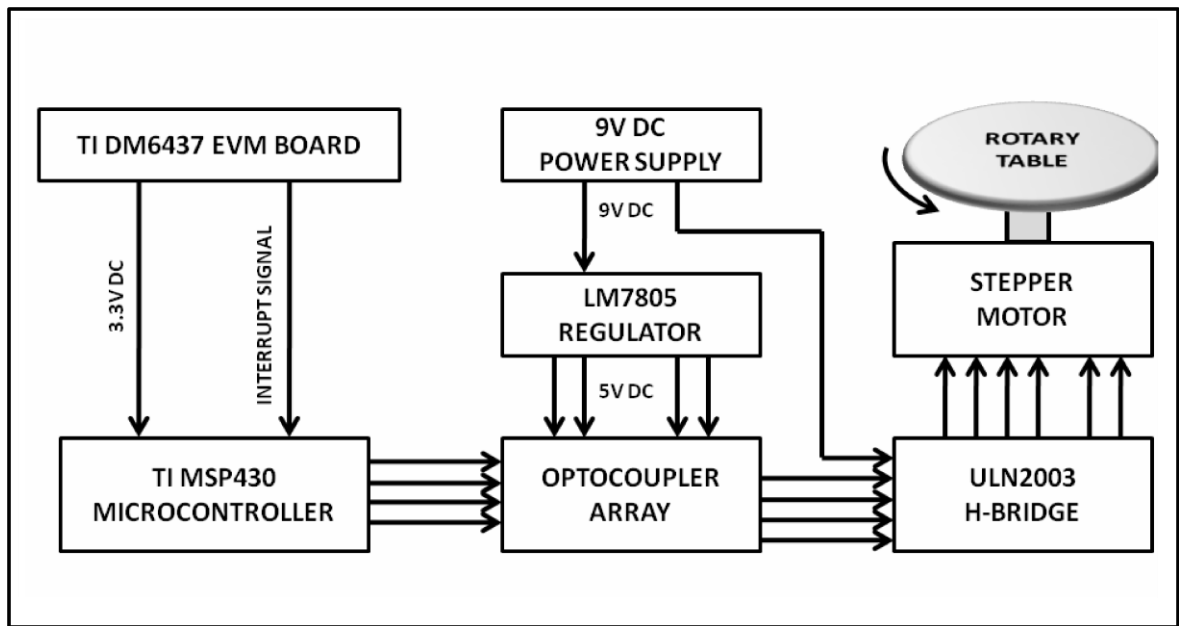


Figure 5.3. Rotating table motor driving schematic

5.1.5.2. The MSP430 microcontroller

To control the states of the motor driving circuitry, we used a TI MSP430F2274 microcontroller. The MSP430F2274 has a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. Port2 of the microcontroller is set to be used as the general purpose output. We provide the state output from this port to drive the motor to the next step. The first bit of port1 on the microcontroller is set to get an interrupt from the TI DM6437 EVM board. On each high to low change signal from the TI DM6437 EVM board, the microcontroller gets an interrupt. The program in the microcontroller branches to the interrupt service routine that places the next state value to port2. This way, according to the number of pulses given by the TI DM6437 EVM board, the microcontroller gets interrupts and rotates the motor to the desired angle. The pictures of the motor driving circuit, motor, and the complete rotating table structure is given in Figure 5.4.

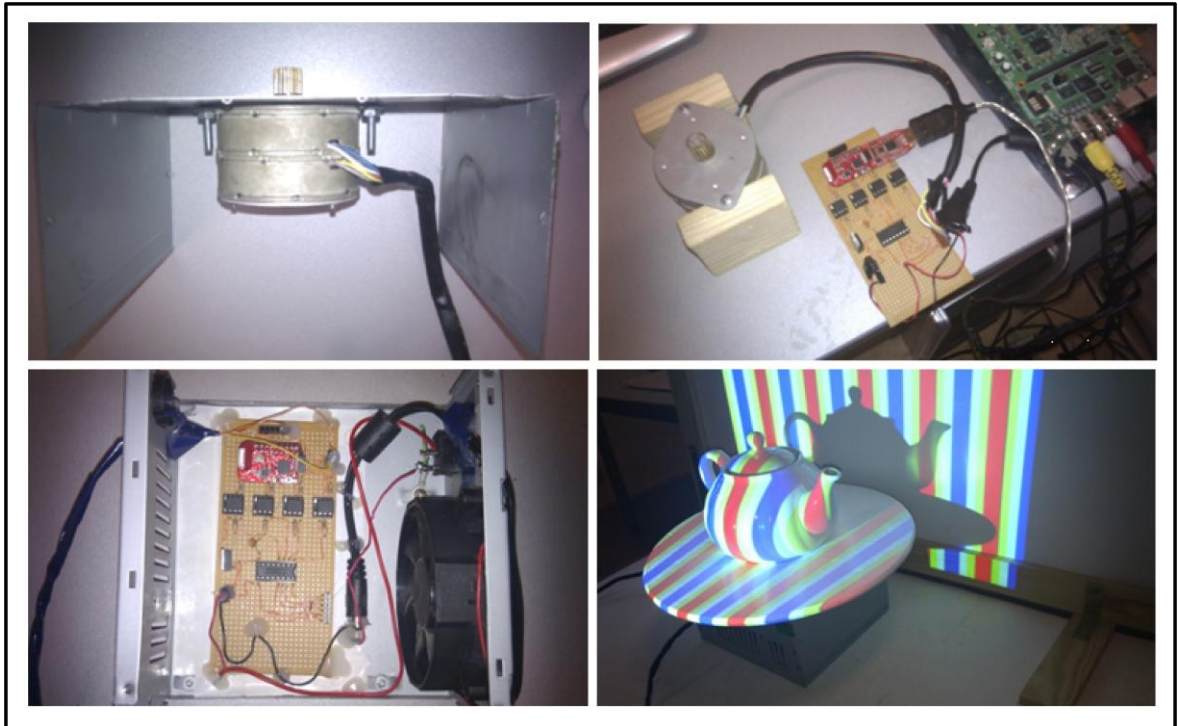


Figure 5.4. Left: The stepping motor used. Top right: The driver circuit. Bottom left: The MSP430 microcontroller. Bottom right: Rotating table during a scan process

5.2. THE SCANNER SOFTWARE

In this section, we focus on the software implementation issues of our range scanner in terms of hardware properties. For more advanced DSP implementations of the proposed methods in this study, we refer the reader to two excellent books [46,47]. We configured the CCS such that, when it is launched on the host computer, it connects to the target board and waits with an empty user interface. By using the GEL script language specified for the CCS environment, we designed a menu item called “3D Scanner” to the user interface. This menu item is given in Figure 5.5.

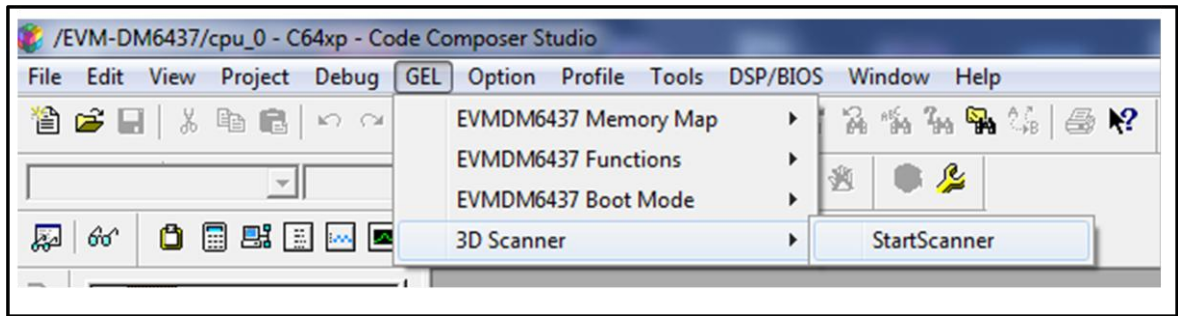


Figure 5.5. Starting the scanner from CCS

By clicking on the “Start Scanner” icon from this menu, the user loads and starts the program. At the beginning, the program asks the user to choose the scanning method. The user selects one of the binary, ternary, or quaternary scanning methods by entering the number corresponding to it. This window is shown in Fig. 29. According to the selection, the program calls the corresponding pattern generation function.

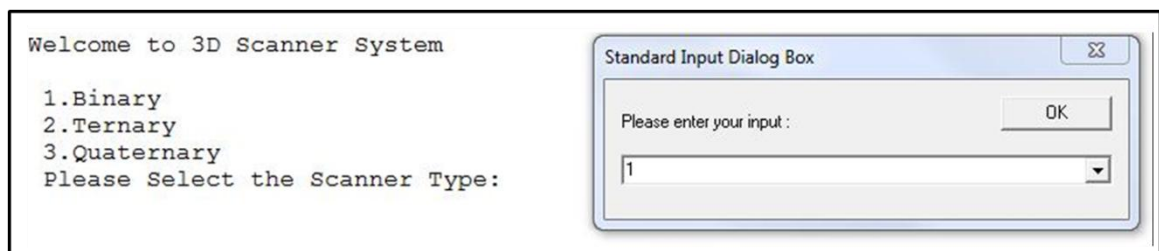


Figure 5.6. Selecting the scanning method by the user

5.2.1. Pattern Generation

In order to prevent delays based on pattern generation during the scanning operation, we prepare the patterns at the beginning of the program. To generate and store the patterns, we used a buffer structural element. For all scanning methods, we use vector header files that contain the pattern structure in a single line. The projection device projects a 576×720 pixel image. According to the selected scanning method, the single line pattern code is expanded to the entire image to be projected. Our scanner software has a queuing structure for parallel processing of image projection and capturing (to be explained in detail next). To benefit from this parallel structure, the program first copies the patterns to be projected

to the output buffer queue. The first pattern is projected by releasing the first output buffer. As the new frame is captured from the input buffer queue, the second pattern is automatically released from the output queue. While the second pattern is being projected, the color conversion operations (to be explained in the pattern decoding section next) are performed. This cycle continues sequentially for each pattern. After the last pattern in the frame is captured, a fully bright white colored image is released from the output buffer. The resulting image is captured to be used for shadow removal stage (to be explained next).

5.2.2. Shadow Removal

The shadow that is captured by the camera can disturb the color invariant calculations. Therefore, it should be removed. On the other hand, some dark colored objects may also be taken as shadow if the method works for all scenes. To prevent this confusion, the system asks the user whether the shadow should be removed or not. If the user selects to remove the shadow regions, the program takes an extra image with a pure white illumination. The Y value (in YC_bC_r) in the grabbed image is thresholded by 40. The pixels lower than this value are assumed to be from a shadow region. Let us remind that, we use a projection device that supports a controlled illumination. Therefore, we can use a fixed threshold. In the image processing stage of our program, we exclude the shadow pixels from further processing. This way, we reduce the unnecessary processor load.

After the shadow removal step, the program waits for the start command to begin the scanning operation. At the same time, the program places the first pattern to be projected from queue to the output. By the start command, the first pattern is projected onto the object. The user interface of this state is as given in Fig. 30.

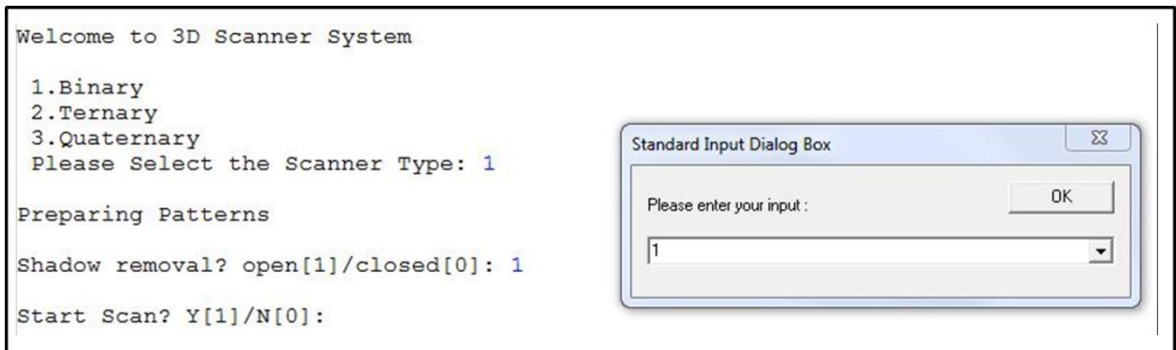


Figure 5.7. Starting the scanning process

5.2.3. Image Capturing

After the pattern is projected onto the object, its image is captured and placed to the video input queue. The frame to be processed is called from the queue to a buffer structural element. To optimize the bandwidth, the image in the buffer is subsampled in 4:2:2 YC_bC_r format by the TVP5146 video decoder of the TI DM6437 EVM board. Our program first decomposes the YC_bC_r data and calculates the corresponding red, green, and blue color values. In the next step of the program, we benefit from these color values in calculating the color invariants for pattern decoding.

Unfortunately, we have a delay in the projection device. This causes problems in capturing the correct pattern projected image from the object surface. To overcome this problem, the program renews the image capturing process three times while the same pattern is released from the output buffer. This causes a delay in the pattern projection and image capturing stages of our software. We will talk about this issue in Section 6.4.

5.2.4. Pattern Decoding

To decode patterns, we apply the color invariant on each pixel of the grabbed image. The resulting values are thresholded (calculated theoretically in previous sections). If the pixel's color invariant value is above the threshold, it is stored in a buffer with a code number related to the pattern projected. The program repeats the pattern projection, image

capturing, and stripe segmentation processes for each pattern. Finally, we obtain a buffer array that contains the decoded data of the object's scanned pose.

5.2.5. Three Dimensional Point Cloud Extraction

As the patterns are decoded in the previous step, we can extract each line stripe (coded by Nary representation) separately. Then, using the triangulation principle and the disparity information between the projected and decoded line stripe positions, we can obtain the depth information [5-7]. We explained the details of triangulation in Chapter 3. For a proper triangulation the system should be calibrated. We obtain the calibration data of our system once. We feed this calibration data to our program as a header file. We gave the details of calibration in Chapter 3.

5.2.6. Scanning Objects from Different Viewing Angles

When the scanning from one viewing angle is completed, the program calls the rotating table function to rotate the object. To prevent any time losses, we rotate the table in parallel with the point cloud calculation operation. To register the point clouds from different viewing angles in a robust manner, the object should be rotated in small angles. Therefore, we scan the object from 11 different angles covering the overall 360 degrees range.

5.2.7. Transfer of Point Cloud Data

At the end of the scanning process, the program informs the user by the “Scan Process Completed” message. The user selects the “Point Out” icon under the “3D Scanner” menu item to transfer the point cloud data to the host computer. The view of this operation is as in Figure 5.8.

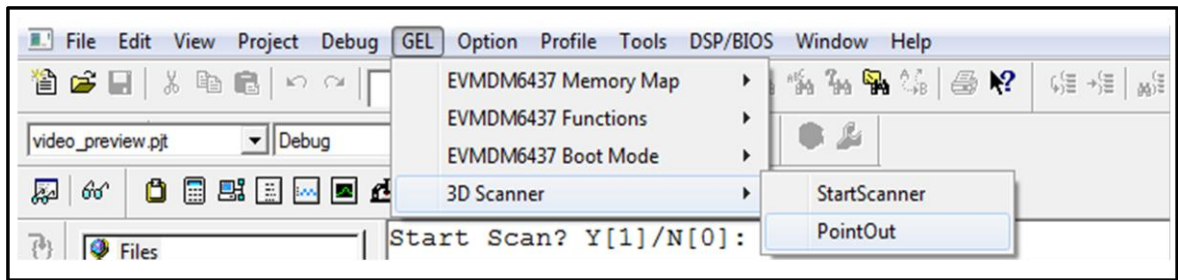


Figure 5.8. Transferring the point cloud data to the host computer

The GEL script we prepared saves the point cloud obtained from 11 different viewing angles in rectangular coordinates (x, y, z) . For each viewing angle, a specific text file is created with “.dat” extension. The registration software (that will be explained in the following chapter) can easily open these files and read the point cloud data.

6. OVERALL PERFORMANCE OF THE SCANNER SYSTEM

In this chapter, we measure the performance of the proposed range scanner system by: the quality of the range data, operation timing, and accuracy. We performed experiments on each range scanner with objects constituting different surface characteristics. We test the proposed range scanner system on these objects under ambient light. For comparison purposes, we first provide the range data extracted by the standard binary structured light based range scanner (using black and white patterns). Then, we provide the range data extracted by our scanners. We also performed timing experiments to show the improvement on scanning time by decreasing the number of patterns. Having the highest precision was not our primary objective. However, we also measured the accuracy of the scanners we proposed. Next, we give the extracted range data by the proposed system.

We provide the images of the first set of our test objects, besides the matte and shiny Atatürk objects (given in Figure 4.2) in Fig. 32. These objects are: shiny teapot, shiny concave fish, shiny carafe, shiny green cat, soft donkey, and hen. In total, five of these test objects have shiny surfaces. The remaining three have matte surfaces. The shiny concave fish has a concave shape. Also, these test objects have different colors on them. We provide the extracted range data of these test objects in terms of *point clouds* for each scanner separately next.

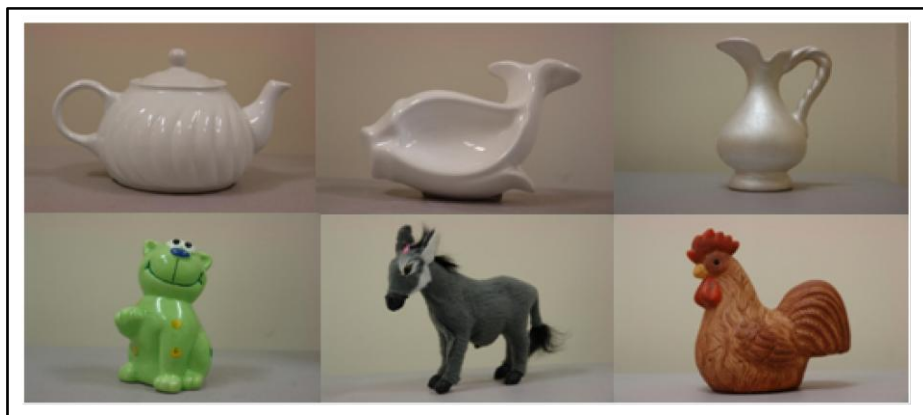


Figure 6.1. First set of test objects. First row: shiny teapot, shiny concave fish, shiny carafe. Second row: shiny green cat, soft donkey, hen

6.1. EXTRACTED RANGE DATA USING THE STANDARD BINARY SCANNER

We implemented the standard binary structured light based range scanner (using black and white color patterns) through our hardware as a benchmark. We provide the range data extracted by this scanner in Figure 6.2.

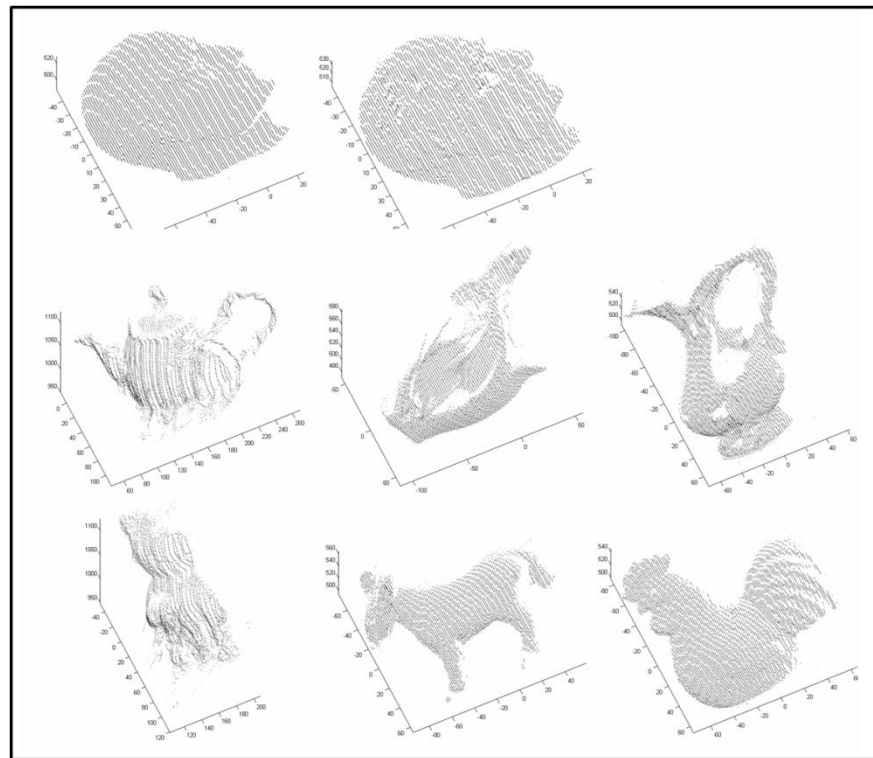


Figure 6.2. Point clouds of eight test objects using the standard binary range scanner. First row: matte Atatürk, shiny Atatürk. Second row: shiny teapot, shiny concave fish, shiny carafe. Third row: shiny green cat, soft donkey, hen

As can be seen in Figure 6.2, the standard binary range scanner gives good results on the matte Atatürk object. However, the chin and the hair parts of the shiny Atatürk object is problematic. As we have mentioned previously, this is due to the problem in the pattern decoding step. This problem is also evident for the remaining test objects. For the matte objects such as soft donkey and hen, the scan results are good. On all other shiny objects, this scanner gives poor results. Either some parts of the objects are missing or there are

some outliers in the extracted range data. The main reason for this poor performance is, as mentioned above, the pattern decoding step.

6.2. EXTRACTED RANGE DATA USING OUR SCANNERS

Since we proposed several scanners, we provide the range data extracted from them separately in this section. We start with binary range scanners based on color invariants c_1 and c_3 separately. Then, we proceed to the results of the ternary and quaternary range scanners.

6.2.1. Binary Range Scanners

We provide the range data extracted by our binary range scanner using c_1 in Figure 6.3. As can be seen in this figure, the range data extracted from all objects are fairly good.

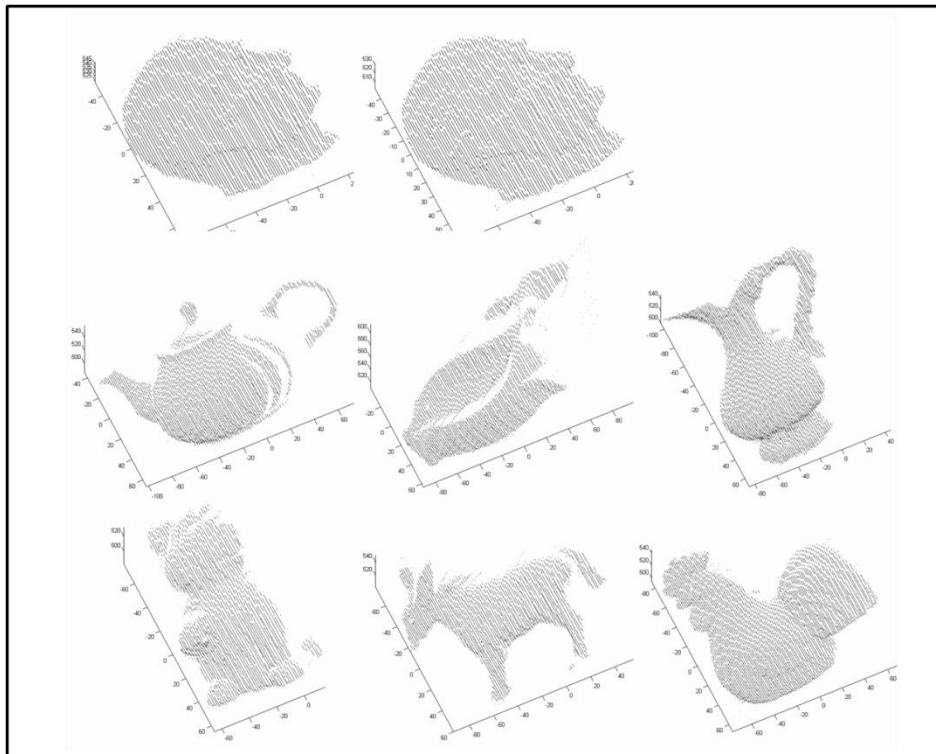


Figure 6.3. Point clouds of eight test objects using the binary range scanner (with c_1). First row: matte Atatürk, shiny Atatürk. Second row: shiny teapot, shiny concave fish, shiny carafe. Third row: shiny green cat, soft donkey, hen

Similarly, we provide the range data extracted by our binary range scanner using c_3 in Figure 6.4. As in the previous scanner, all results are fairly good. The main difference between this and the previous binary scanner is the used pattern colors (blue-green) in coding.

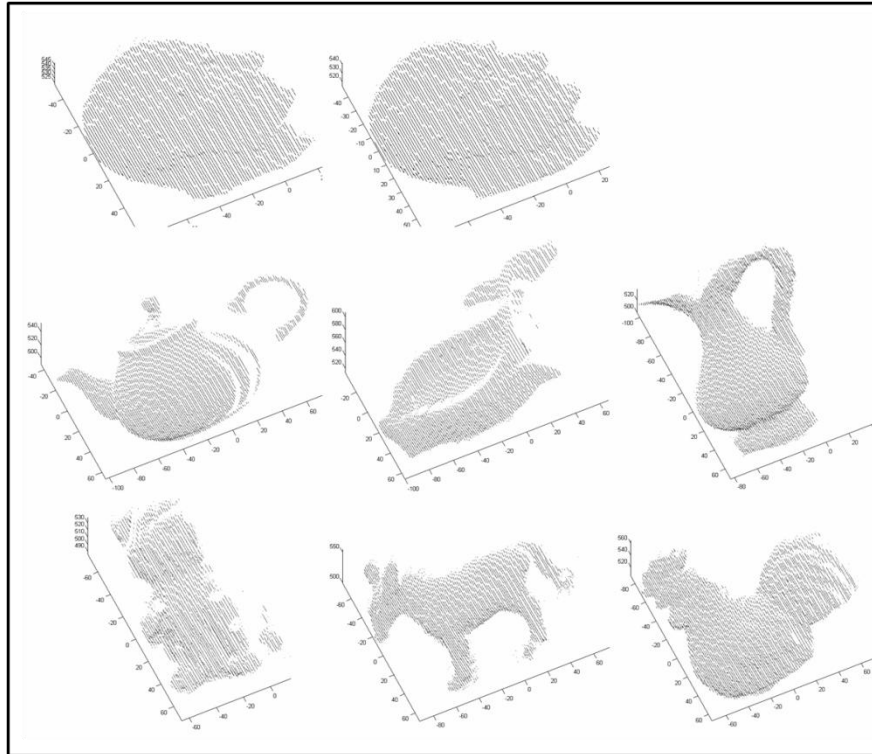


Figure 6.4. Point clouds of eight test objects using the binary range scanner (with c_3). First row: matte Atatürk, shiny Atatürk. Second row: shiny teapot, shiny concave fish, shiny carafe. Third row: shiny green cat, soft donkey, hen

6.2.2. The Ternary Range Scanner

We provide the range data extracted by our ternary range scanner in Figure 6.5. As can be seen in this figure, the range data extracted from all test objects using this scanner are also fairly good.

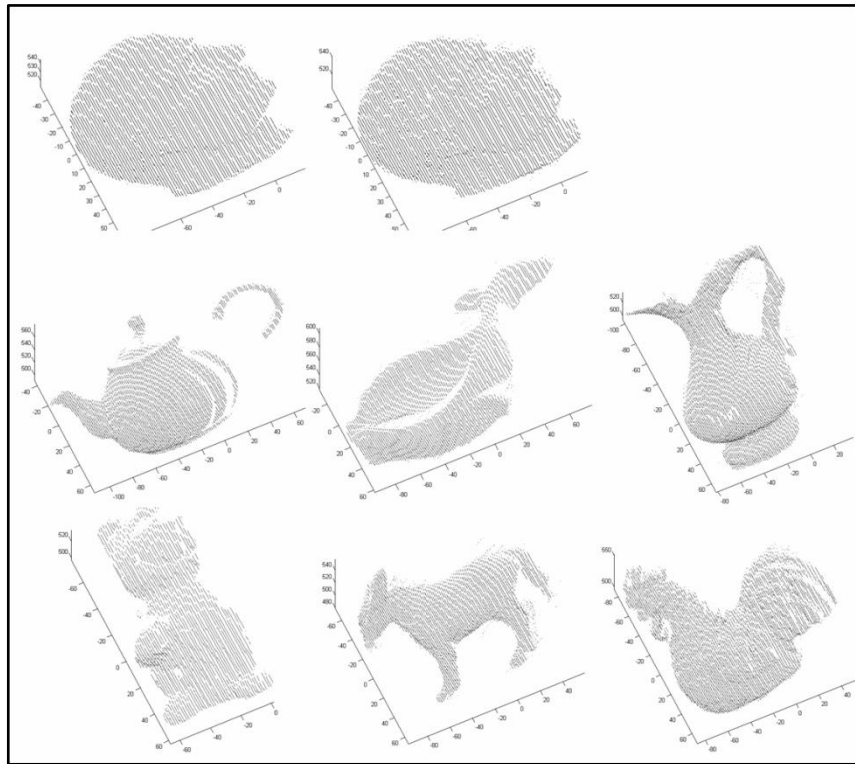


Figure 6.5. Point clouds of eight test objects using the ternary range scanner. First row: matte Atatürk, shiny Atatürk. Second row: shiny teapot, shiny concave fish, shiny carafe. Third row: shiny green cat, soft donkey, hen

6.2.3. The Quaternary Range Scanner

Finally, we provide the range data extracted by our quaternary range scanner in Figure 6.6. Although the results obtained with this scanner are better than the standard binary range scanner, for the green cat and hen objects, the range data is not as good as the ternary and binary range scanners. One possible explanation for these results is the decoding step. As we mentioned before, for decoding the cyan colored stripes, we could not obtain a symmetric relationship. This may have caused minor problems in the decoding step.

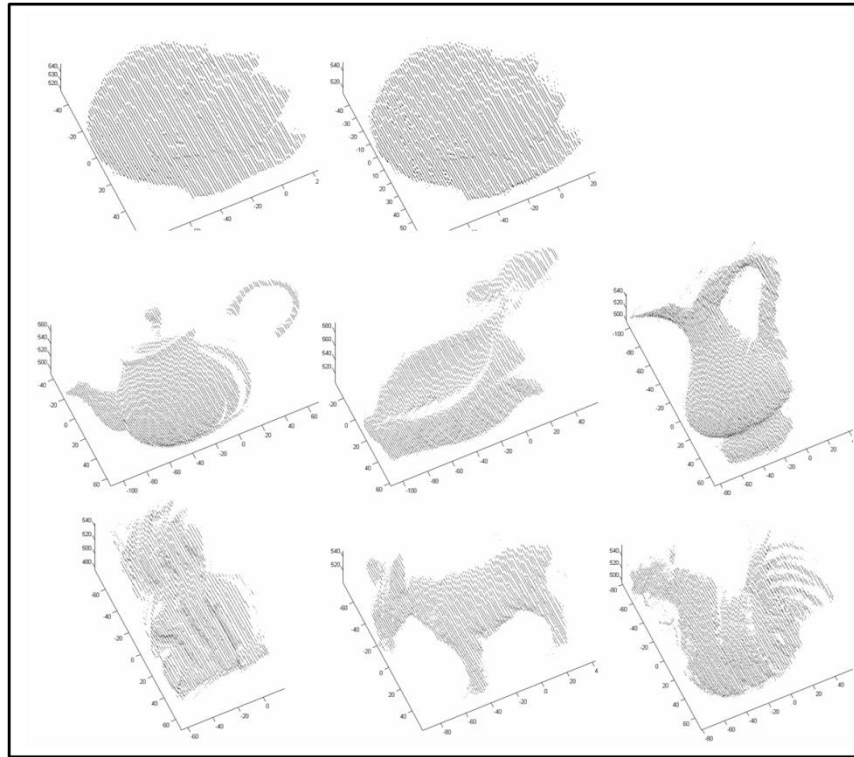


Figure 6.6. Point clouds of eight test objects using the quaternary range scanner. First row: matte Atatürk, shiny Atatürk. Second row: shiny teapot, shiny concave fish, shiny carafe. Third row: shiny green cat, soft donkey, hen

6.3. QUANTITATIVE COMPARISON OF THE SCANNER RESULTS

To have a quantitative comparison of the standard binary range scanner and the ones proposed in this study, we measured the percentage of the outliers and the missing points on the extracted range data. Table 6.1. Average percentage (%) of outliers for the scanners tabulates the percentage of the outlier points for each scanner. For a fair comparison, we divided our test objects as matte (matte Atatürk, soft donkey, hen) and shiny (shiny Atatürk, shiny teapot, shiny concave fish, shiny carafe, and shiny green cat). For each scanner type, we provide the average percentage of outliers separately (for matte and shiny objects) in this table.

Table 6.1. Average percentage (%) of outliers for the scanners

Scanner Type	Object Type	
	Matte	Shiny
Standard binary	0.34	2.01
Binary c_1	0.37	0.23
Binary c_3	0.70	0.55
Ternary	0.72	0.45
Quaternary	0.74	2.17

As can be seen in Table 6.1. Average percentage (%) of outliers for the scanners, for matte objects the standard binary scanner gives better results compared to others. However, the performance improvement is not significant. On the other hand, for the shiny objects the standard binary scanner has an average of 2.01% outlier. On the average, this scanner gives 38000 points for the five shiny objects. Therefore, the standard binary range scanner has an average of 763 outliers for each shiny object. This number is almost tenfold more than the binary range scanner using c_1 . In this table, it can be seen that, the quaternary scanner has a high number of outliers. However, this is not because of the shiny surface characteristics. It is based on the color of the surface. Especially, for the shiny green cat object, the effect of sensor crosstalk caused the number of the outliers to be higher. This increases the average percentage error for the quaternary scanner.

Not only the outliers but also missing points occur based on decodification errors. Therefore, we also measured the percentage of the missing points on the same matte and shiny test objects. Table 6.2. Average percentage (%) of missing points for the scannerstabulates the average percentage of missing points for each scanner.

Table 6.2. Average percentage (%) of missing points for the scanners

Scanner Type	Object Type	
	Matte	Shiny
Standard binary	0.11	3.35
Binary c_1	0.02	0.76
Binary c_3	0.02	0.24
Ternary	0.02	0.08
Quaternary	0.02	0.02

As can be seen in Table 6.2. Average percentage (%) of missing points for the scanners, for both matte and shiny objects the percentage of the missing points is higher for the standard binary scanner. Based on the calculations for outliers in the above paragraph, for a shiny object approximately 1273 points are missing on the average. However, in our scanners this number decreases to 288 in the worst case. This is a fivefold improvement. This improvement and the one obtained for the outliers in the previous paragraph clearly show that our range scanners provide better range data compared to the standard binary range scanner.

6.4. TIMING PERFORMANCE OF THE SCANNER SYSTEM

We also tested the timing performances of our range scanners. We picked the shiny carafe as the test object. We provide the time needed for each scanning step (from one viewing angle only) in Table 6.3 for our binary, ternary, and quaternary range scanners. In this table, ‘Initialization’ stands for the initialization of the TI DM6437 EVM board. ‘Pattern generation’ is the step to generate the pattern image to be projected. ‘Configuration’ stands for the configuration of the video processing subsystem. In the ‘Acquisition’ step, the pattern to be projected is filled to the buffer, projected, acquired, decomposed, and converted from $YCbCr$ 4:2:2 format to RGB . ‘Pattern decoding’ step involves the usage of color invariants to decode patterns. ‘Rotating the table’ step involves the time needed to control the rotating table for the next scan. Finally, ‘Calculations’ step

includes all the calculations to output real world coordinates of the scan result. We take the clock of the TI DM6437 EVM board as 600 MHz.

Table 6.3. Operation timings in milliseconds for the scanners

Step	Time (msec)		
	Binary	Ternary	Quaternary
Initialization	0.19	0.19	0.19
Pattern generation	415.87	307.67	283.38
Configuration	0.10	0.10	0.10
Acquisition	1778.51	1180.79	968.72
Pattern decoding	2393.45	1969.02	3477.81
Rotating the table	88.17	88.17	88.17
Calculations	1918.10	2649.32	728.94
TOTAL	6594.39	6195.26	5547.31

As can be seen in Table 6.3. Operation timings in milliseconds for the scanners, the total time needed to scan the shiny carafe object from one viewing angle is 6.59 sec, 6.19 sec, and 5.54 sec for the binary, ternary, and the quaternary range scanners respectively. However, the time needed just to acquire the pattern images (including all initialization steps) from the object surface is **2.19** sec, **1.49** sec, and **1.25** sec for the same scanners. Therefore, to scan an object from one viewing angle, it should stay in front of the binary scanner at most 2.19 sec. For the quaternary scanner, this time reduces to 1.25 sec. The rest of the operations can be done off line.

In Table 6.3, it is clearly seen that, the pattern decoding, acquisition, and calculations steps take most of the operation time. Since our method needs color information, these timings are unavoidable. Besides, in the calculations step, the timing between each scanner type is different since each extract different number of range points depending on their spatial resolution. This is because of the different line widths used for each scanner. In terms of total timings, the quaternary range scanner is the fastest of all, as expected.

Besides, the binary and ternary range scanners have fairly good operation speeds. These results are due to the TI DM6437 EVM board and using C programming with optimized fixed point coding libraries.

The proposed method implemented on the TI DM6437 EVM board may seem slow compared to fringe projection based methods such as [4]. This is because of three main reasons. First, due to budget constraints and availability we had to use a standard projection device in the setup. Therefore, delays in projecting the patterns and acquiring them became unavoidable. Besides, it was not possible to use an external trigger for this projector. There are projection devices specifically designed for range scanning applications. Using these will definitely shorten the operation timings. Second, our DSP platform was specifically designed to process video images. Although it was a good choice for academic purposes, developing a real time range scanner was not possible based on its characteristics. Third, using color information also decreased the timing performance of our system. On the other hand, our proposed method implemented on a TI DM6437 EVM board as suggested in this study is faster than standard PC based implementations. One can see timing comparisons of recent structured light based range scanning systems in the review paper [2].

6.5. ACCURACY OF THE SCANNER SYSTEM

Caspi *et al.* [9] used a staircase object to verify the accuracy of their system. Similarly, we tested the accuracy of our binary range scanner on a staircase object given in Figure 6.7. This object has three levels. We measured the actual depth of each staircase level from three different locations (left, middle, right) by a caliper. Then, we measured the depth of these locations by our range scanner. We provide the results in Table 6.4.

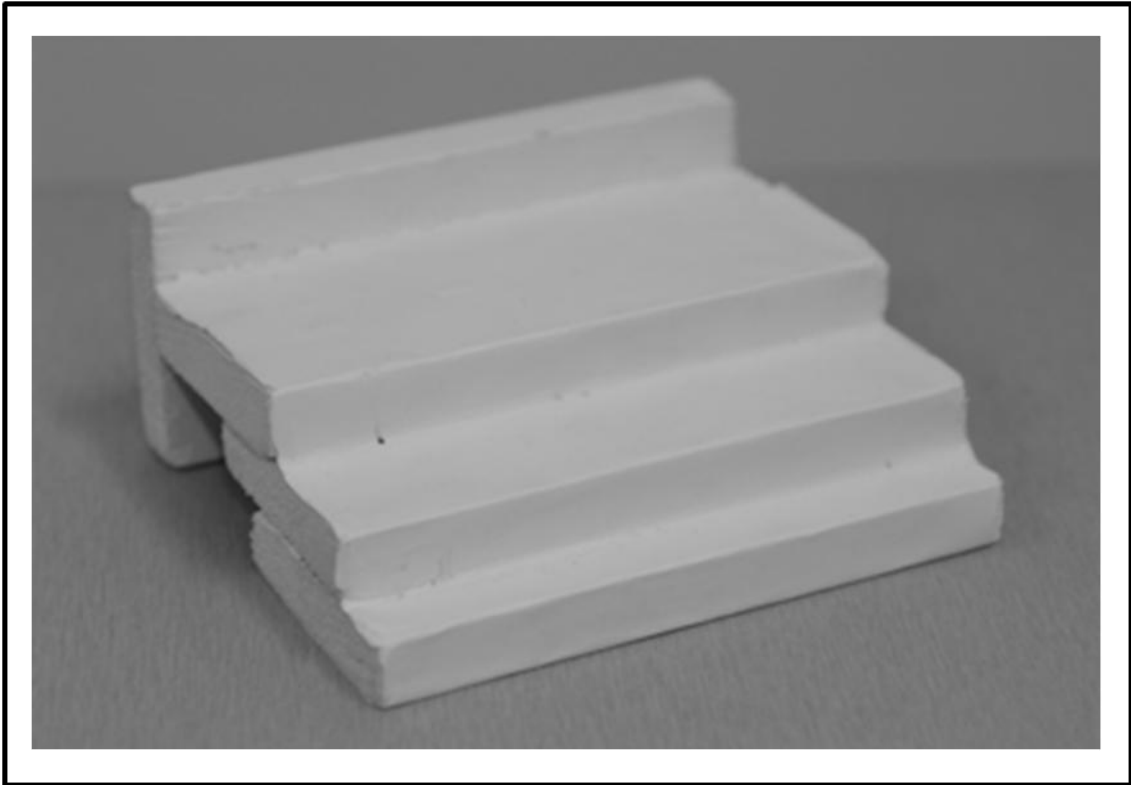


Figure 6.7. The staircase object for testing the accuracy of range scanners

Table 6.4. Comparison of the actual and the measured depth values (in millimeters) on the staircase test object

Step, location	Actual	Measured	Error
First step, left	39.90000	39.11473	0.78527
First step, middle	39.99000	40.03656	-0.04656
First step, right	39.80000	39.27202	0.52798
Second step, left	20.59000	19.20797	1.38203
Second step, middle	19.92000	19.31543	0.60457
Second step, right	20.34000	20.67780	-0.33780
Third step, left	10.29000	9.86059	0.42941
Third step, middle	9.62000	9.80263	-0.18263
Third step, right	9.55000	9.80600	-0.25600

As can be seen in Table 6.4, the maximum error is 1.38203 mm on the second step. The average error for the first, second, and third steps are 0.45327 mm, 0.77480 mm, 0.28935 mm respectively. The overall average error for the staircase test object is 0.50581 mm. Based on these tests, we can claim that the accuracy of our binary range scanner is acceptable. To note here, the accuracy was not the main target in this study. Using more advanced techniques and equipments, this accuracy can be improved further.

6.6. COMMENTS ON THE PERFORMANCE

Comparing all the range data extracted by five range scanners, we can summarize some key observations. First of all, the standard binary range scanner (using black and white patterns) is not a good choice for scanning shiny objects under ambient light. The binary range scanner based on c_1 gives good results on all test objects. The other binary range scanner based on c_3 also gives good results on all test objects. The range data extracted by the ternary range scanner is fairly good. Finally, the range data extracted by the quaternary range scanner from most test objects is fairly good. Only for some challenging objects, the extracted range data is not as good as the other scanners based on color invariants. The average value of outliers and the missing points justifies these claims quantitatively. We also compared the operation timings for these scanners in Section 4. As expected, the speed of the system increases as the number of patterns decrease. However, the user should decide on using whether a high speed scanning or a high resolution scanning. As a final comment, either fast or slow, with high or low resolution, our scanner system can scan shiny or matte surfaces reliably under ambient light.

7. 3D MODEL CONSTRUCTION

The final step in object scanning is the 3D model construction. The proposed scanner system gives 3D point cloud as an output. Then, we obtain the polygon meshes of the poses from different angles of the object. For obtaining the object model, these meshes should be registered. We use the iterative closest point (ICP) algorithm to register these patches [5]. Next, we will give a brief explanation of the ICP method. Then, we summarize the test objects used. Finally, we give the models of these test objects.

7.1. 3D MODEL CONSTRUCTION USING ICP

There are several 3D point set registration algorithms proposed in the literature. Rodrigues *et al.* [48] presented a survey on major registration algorithms. Iterative Closest Point (ICP) proposed by Besl and McKay [5] is the current state-of-art algorithm. Rusinkiewicz and Levoy [49] categorized and summarized variants of the ICP algorithm. Here we will explain the basic algorithm.

ICP is an algorithm introduced to register the two set of 3D points. Since the algorithm is very effective, it is commonly used to reconstruct the final 3D models of real objects from their range data. The algorithm works on a basis of iterative estimation of the Euclidean transformation (translation and rotation) between the two point sets. The algorithm requires an initial estimation of the transformation. Till satisfying the stopping criteria, the algorithm works iteratively. The output of the algorithm is the refined transformation.

Assume that we try to register the two set of range image points O and F . The algorithm calculates the 3D rotation matrix \mathbf{R} and the translation vector \bar{T} that minimizes the error as

$$f(R, \bar{T}) = \frac{1}{N_o} \sum_{k=1}^{N_o} \| \bar{x}_k - R \cdot \bar{o}_k - \bar{T} \| \quad (7.1)$$

where \bar{o}_k is the k^{th} point in the point set to be registered, \bar{x}_k is the k^{th} point in the reference point set to which O is registering. N_O is the number of points in the data set O .

The closest points are calculated using the Euclidian distance between each points. For two points given as $\bar{e}_1 = (x_1, y_1, z_1)$ and $\bar{e}_2 = (x_2, y_2, z_2)$ the Euclid distance is

$$d(\bar{e}_1, \bar{e}_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (7.2)$$

Let F be point set with N_f points denoted by $\bar{f}_k: O = \{\bar{f}_k\}$ for $k = \{1, \dots, N_k\}$. The distance between a point to be registered \bar{o} and the point set set F is

$$D(\bar{o}, F) = \min_{k \in \{1, \dots, N_f\}} d(\bar{o}, \bar{f}_k) \quad (7.3)$$

For every point o_k in the point set O , and the corresponding point in F is computed using Eqn. 7.3.

The ICP algorithm iteratively minimizes the error function. It starts with an initial rotation matrix \mathbf{R} and the translation vector \bar{T} that transforms the data set to be registered and calculates the registration error. Then it calculates the rotation matrix \mathbf{R} and the translation vector \bar{T} again by minimizing the distance. Until the error reaches to the required level, the process continues iteratively. This iterative process is guaranteed to converge to a local minimum for any starting value of O when it is a subset of F . However, there are some limitations of the ICP algorithm when used in range data registration. First, range data are not subsets of each other. Instead, they partially overlap with each other depending on the viewpoint. Therefore, the algorithm requires the detection of outliers that comes through the non-overlapping regions. Second, the algorithm requires a good initial estimation close to the global minimum in order to avoid any local minimum.

We give an example of ICP algorithm implementation for the hen test object in Figure 7.1 On the left side, the two patches are given with blue and red colored points. The middle figure shows the rough registration using ICP algorithm. The right figure shows the fine registration of the two point clouds.

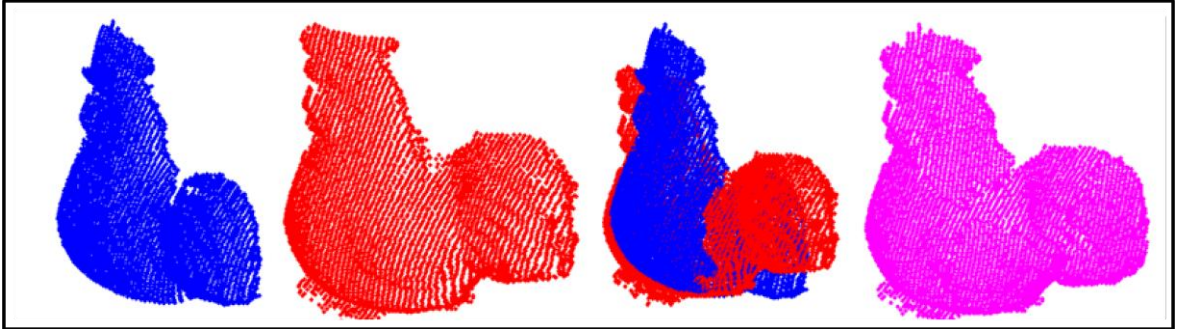


Figure 7.1. ICP implementation example under MATLAB

The ICP algorithm has an average complexity of $Q(N^2)$, where N is the number of points in the range image. One needs to compute the corresponding point pairs in every iteration. This increases the complexity and time consumption of the algorithm. For a faster algorithm, Sertel and Ünsalan [50] proposed using the edge information that makes the rough registration based on the edges of the patches. Although we can implement this, because of the time complexity, we used a commercial software having an optimized and fast ICP implementation. To note here, other registration methods may also be used taking the rotating table properties into account.

7.2. PROPERTIES OF THE TEST OBJECTS

To test the overall system, from range data extraction to model formation, we picked 28 test objects given in Figure 4.2, Figure 6.1 (given in the previous chapters) and Figure 7.2 (given below). These test objects have diverse surface characteristics. We provide the dimensions of these test objects in Table 7.1 As can be seen in this table, the dimensions of our test objects are also diverse.



Figure 7.2. Second set of test objects. First row: circular carafe, column vase, bunny, clay pot, armed vase, shoe. Second row: elephant, brown shoe, dove and yellow shoe. Third row: hedgehog, bird, cornered vase, Venus. Fourth row: Alexander, moon, shiny fish, shiny

Table 7.1. Dimensions of test objects (in millimeters) used in experiments

Object	Length	Width	Depth
Atatürk	110	120	10
Shiny teapot	110	118	113
Shiny concave fish	110	195	45
Shiny carafe	170	140	110
Shiny green cat	140	90	80
Soft donkey	110	145	45
Hen	150	150	80
Circular carafe	150	125	40
Column vase	160	100	100
Bunny	85	55	60
Clay pot	185	120	120
Armed vase	190	110	85
Shoe	80	125	40
Elephant	65	95	70
Brown shoe	90	135	65
Dove	140	185	100
Yellow shoe	80	85	40
Hedgehog	50	80	60
Bird	80	145	70
Cornered vase	185	75	75
Venus	160	170	125
Alexander	260	230	160
Moon	120	40	5
Shiny fish	70	120	25
Shiny stork	90	100	10
Cow	75	65	20
Shiny star	120	120	25
Shiny rose	85	65	30

7.3. OBJECT MODELS EXTRACTED BY THE PROPOSED SCANNER SYSTEM

In this section, we provide the models of our test objects using our binary range scanner (using c_1). For each object, we provide the object model from three different directions. Only the final eight objects are imaged from only one direction, since they have flat surfaces. In Figure 7.3, we provide the models of the shiny teapot, shiny carafe, shiny green cat, soft donkey and hen objects.

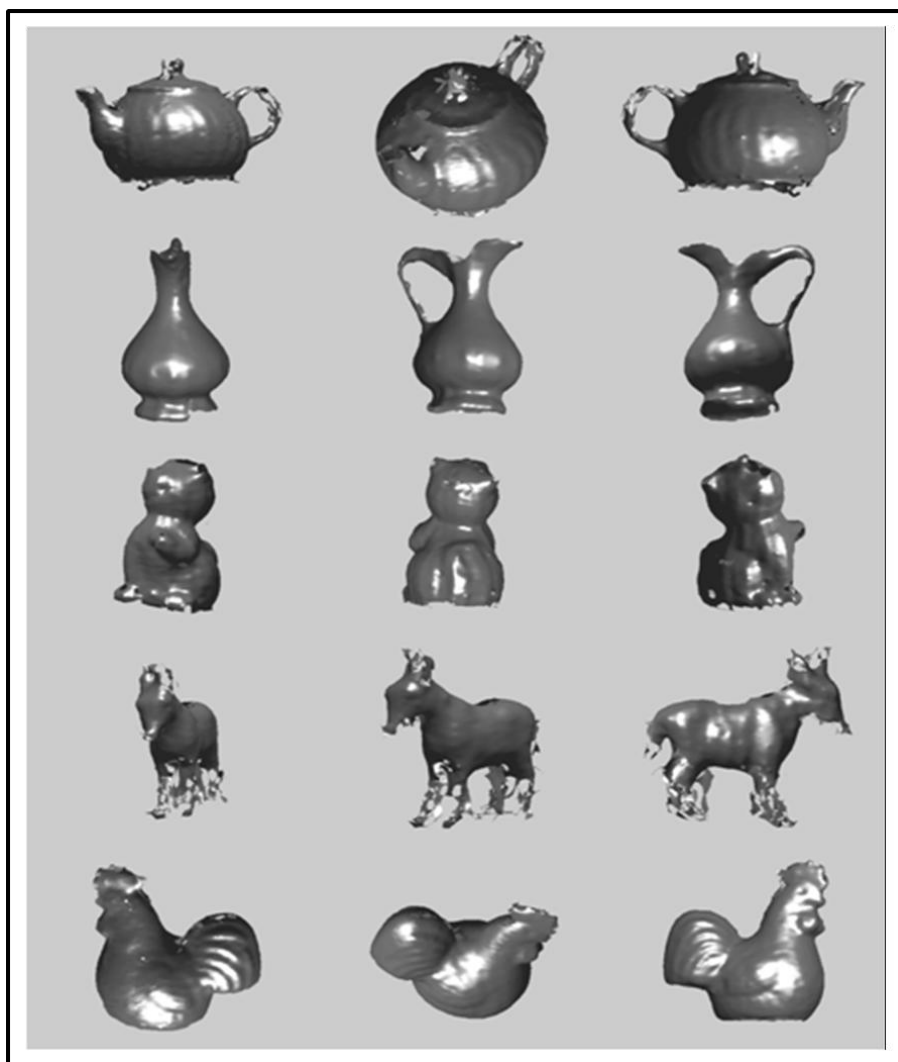


Figure 7.3. Models of shiny teapot, shiny carafe, shiny green cat, soft donkey, and hen objects

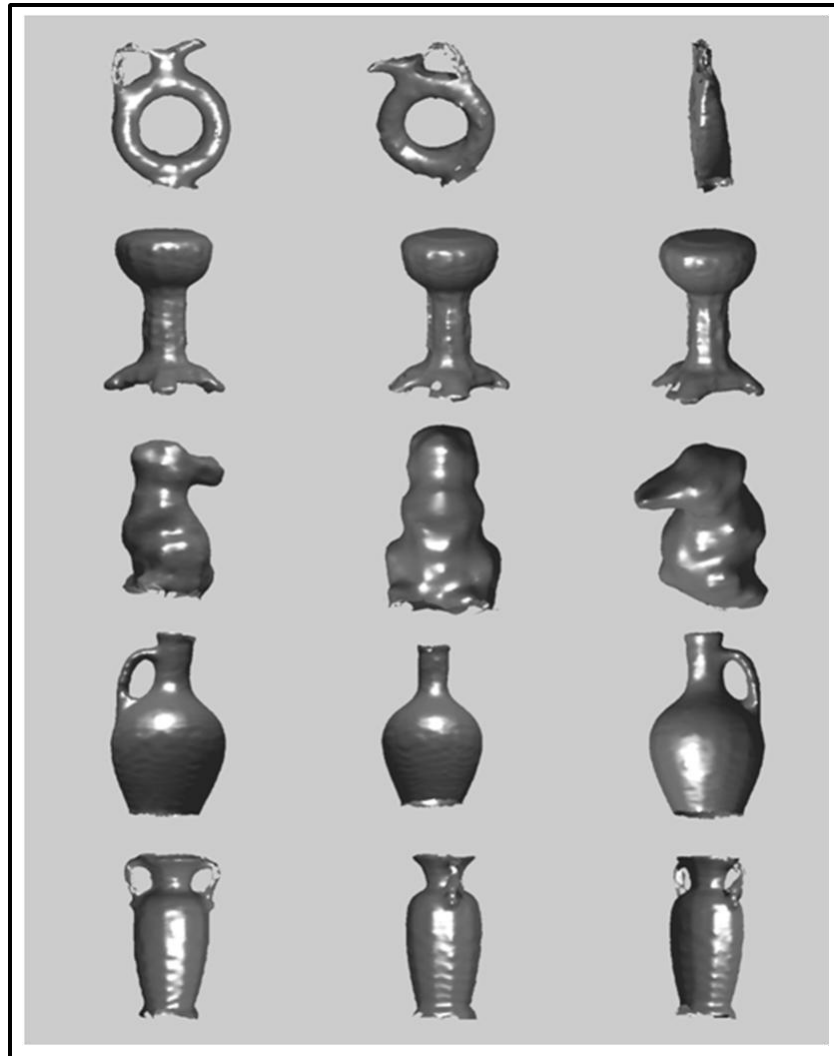


Figure 7.4. Models of the circular carafe, column vase, bunny, clay pot, and armed vase object

Figure 7.4 holds the models of the circular carafe, column vase, bunny, clay pot, and armed vase object. Similarly, in Figure 7.5, we provide the models of shoe, elephant, brown shoe, dove, and yellow shoe objects. In Figure 7.6, we provide the models of the hedgehog, bird, cornered vase, Venus, and the Alexander object.

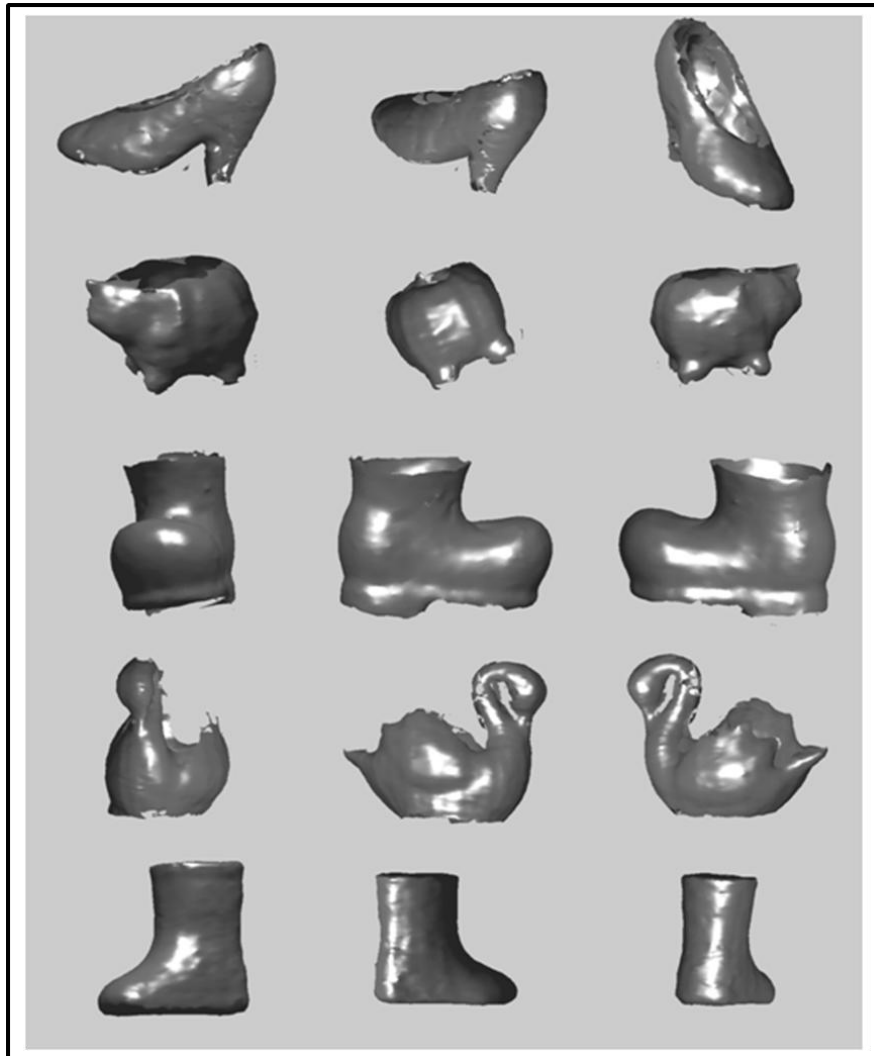


Figure 7.5. Models of the shoe, elephant, brown shoe, dove, and yellow shoe objects

We provide the models of flat objects shiny concave fish, shiny Atatürk, moon, shiny fish, shiny stork, shiny rose, cow and shiny star in Figure 7.7. These are small sized and highly detailed objects. As can be seen, all object models are reliably extracted by the proposed range scanner system. These results are fairly good. We provide our face scan results by our system in Figure 7.8. We also provide our texture mapped face scans in the same figure. Finally, we provide our body and hand scans in Figure 7.9. As can be seen, our face and body scan results are also fairly good.

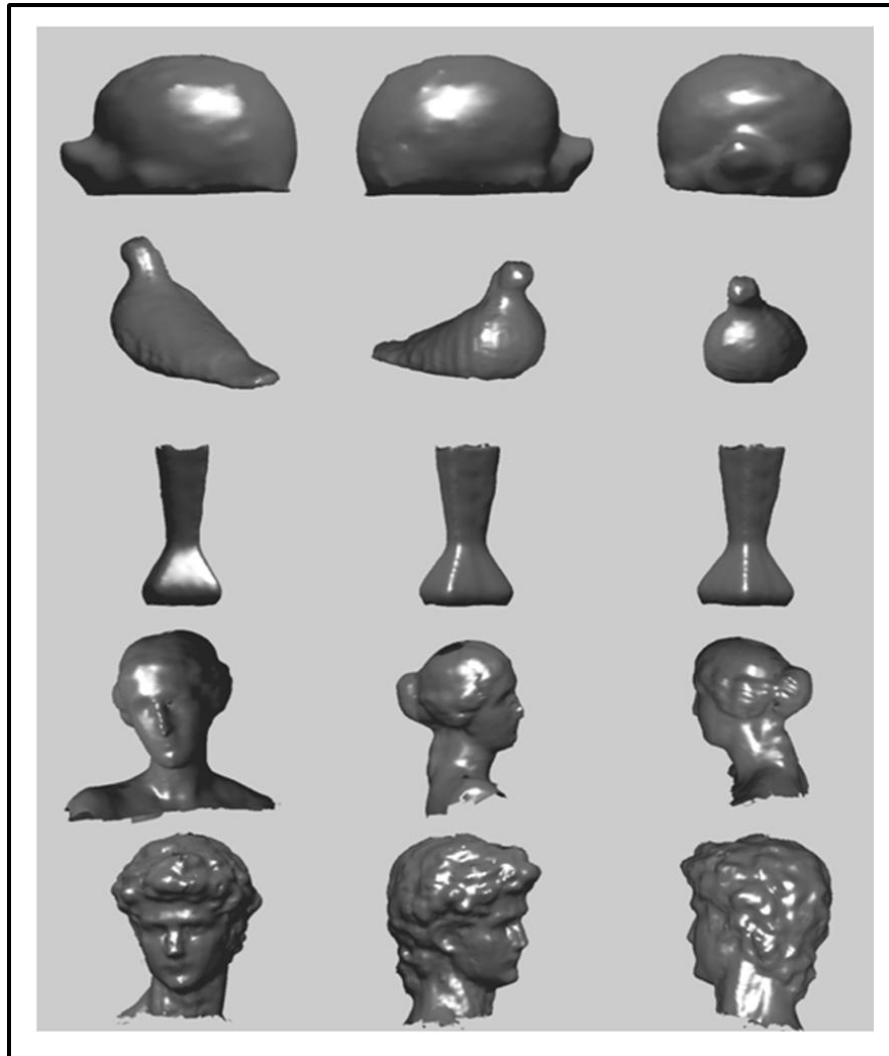


Figure 7.6. Models of the hedgehog, bird, cornered vase, Venus, and Alexander objects

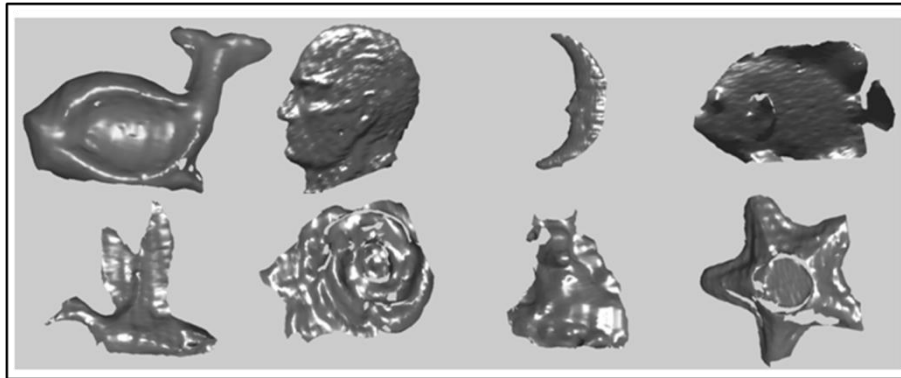


Figure 7.7. Models of the flat objects. First row: shiny concave fish, shiny Atatürk, moon, shiny fish. Second row: shiny stork, shiny rose, cow, shiny star

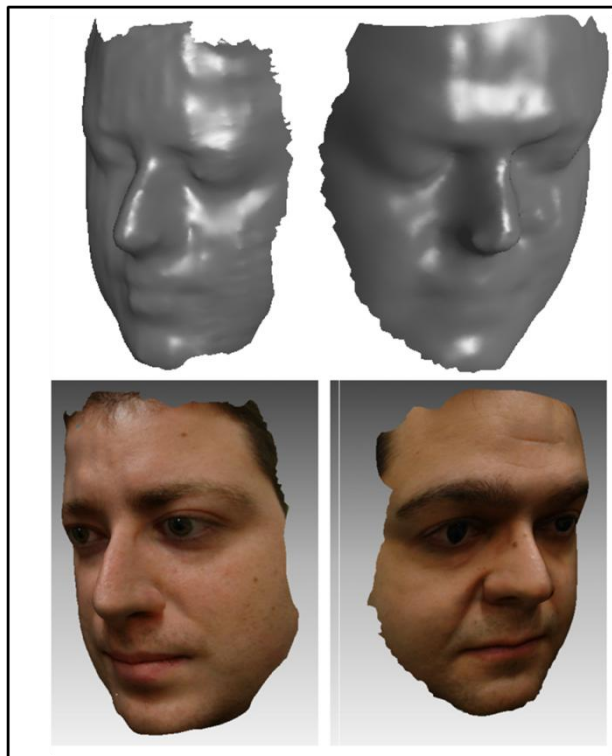


Figure 7.8. Our face scans and their texture mapped versions

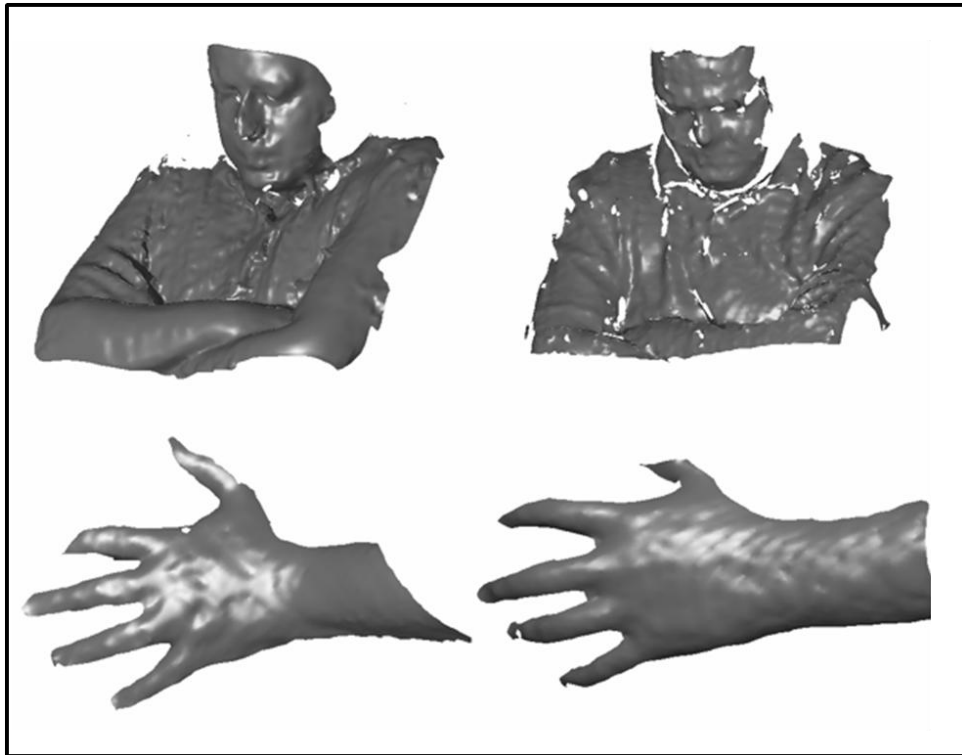


Figure 7.9. Our body and hand scans

8. CONCLUSIONS

In this study, we designed and implemented a novel range scanner system both in hardware and software. Our aim was to solve the problem of range scanning of shiny objects under ambient light. We benefit from color pattern projection and color invariants for this purpose. This is our main contribution since the color invariants are used for the first time for this application. We developed five range scanning methods based on single stripe, binary, ternary, and quaternary color coded patterns. Although the standard binary coded structured light based range scanner could not extract the range data of shiny objects, our color invariant based range scanners were able to extract the same range data in a reliable manner. The increase of the number of colors used in the scanner system directly affect the speed of the system. However, the resolution is decreased related to the coding strategy. If the main aspect of the user is speed, than the user can use the ternary or quaternary scanner. If the aim is higher precision, then the user can select the scanner with less number of colors.

Another important objective of this study is to implement this system on an embedded processor. This way, the overall processing speed is increased and the system became less dependent on a computer. Therefore, we implemented our range scanners on a TI DM6437 EVM board. All the range data extraction software works on this board with an optimized, fast, and reliable structure. Only the 3D model formation part of our range scanner system works on the host computer. By a GEL script file, we also implemented a basic interface for the user. The system gives the entire complete object model in an acceptable time and quality.

The hardware limits us on developing range scanners with less number of patterns. Since the EVM board we use does not have a component RGB input, we have a sensor cross talk problem on the image captured from the camera. Also, the resolution of the camera limits us on the minimum width of the stripe pixels that we can project. Another hardware limitation of the system is the projection device. Based on the delays of the projection device, we had to slow down the system in the pattern image acquisition step. With a triggered system, that works synchronized with the camera, we may have a faster

operation. Obtaining very high resolution range data was not the main target of our study. Therefore, we did not design modules for reaching subpixel accuracies. On the other hand, we have adequate quality and timing from the proposed range scanner system. As a general conclusion, we can claim that our color invariant based range scanner system can be used to scan shiny and matte objects under ambient light in an acceptable operation time and quality.

APPENDIX A: OTHER INVARIANTS FOR THE SCANNER SYSTEM

Here, we summarize other color invariants that we used in our experiments. These are: Hue, normalized RGB, YC_bC_r , and some other invariants that were proposed by Gevers and Smeulders for content based image retrieval. Some of these invariants give good results for a specific type of scanner. However, for the completeness of the system, we choose the invariant that works on all scanners in a reliable manner. In comparing these color invariants, we picked the metallic plate object given in Figure A.1. We also provide the segmentation results in color coded form in the following sections

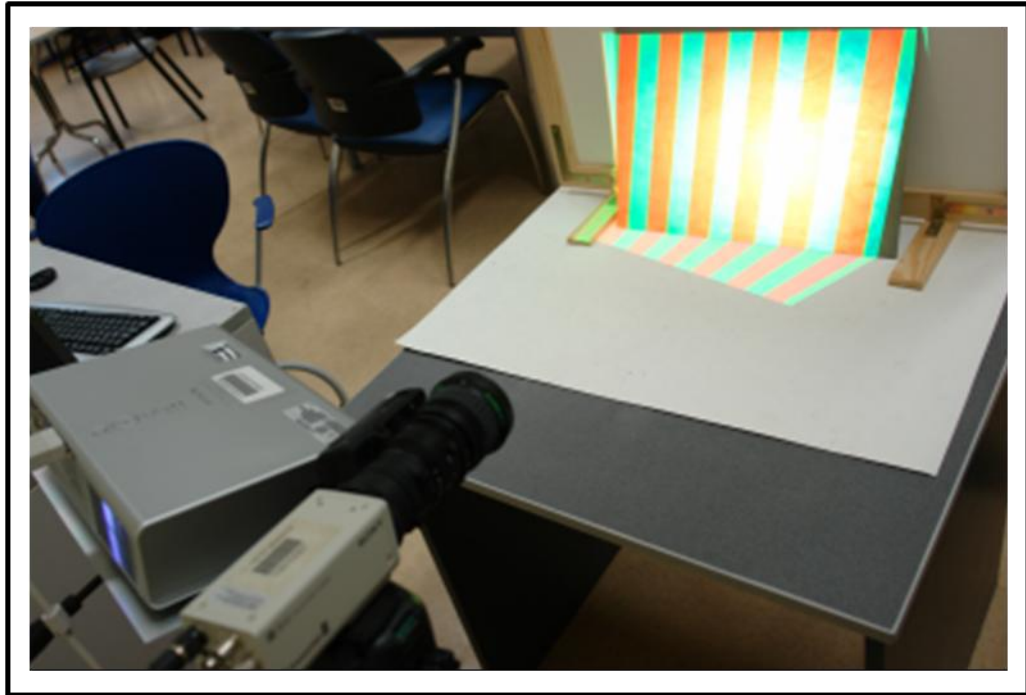


Figure A.1. The metal plate object used as a benchmark

A.1. INVARIANTS TESTED FOR THE BINARY CODED STRUCTURED LIGHT SCANNER

First, we performed experiments on Hue and normalized RGB color components (R_n, G_n, B_n). Hue is the the quality of a color as determined by its dominant wavelength. Normalized RGB values are obtained by the ratio of the base color to the sum of the base colors of the pixel. These invariants are calculated as

$$Hue = \arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \quad (A.1)$$

$$\begin{aligned} R_n &= \frac{R}{R+G+B} \\ G_n &= \frac{G}{R+G+B} \\ B_n &= \frac{B}{R+G+B} \end{aligned} \quad (A.2)$$

We provide the segmentation results for these scanners in Figure A.2. As can be seen, Hue provides a result that can be thresholded without being effected from the highlights. In R_n and G_n , the effect of the highlights will create a problem in thresholding. Since B_n covers the blue color, we do not expect a result for the red and green colored pattern.

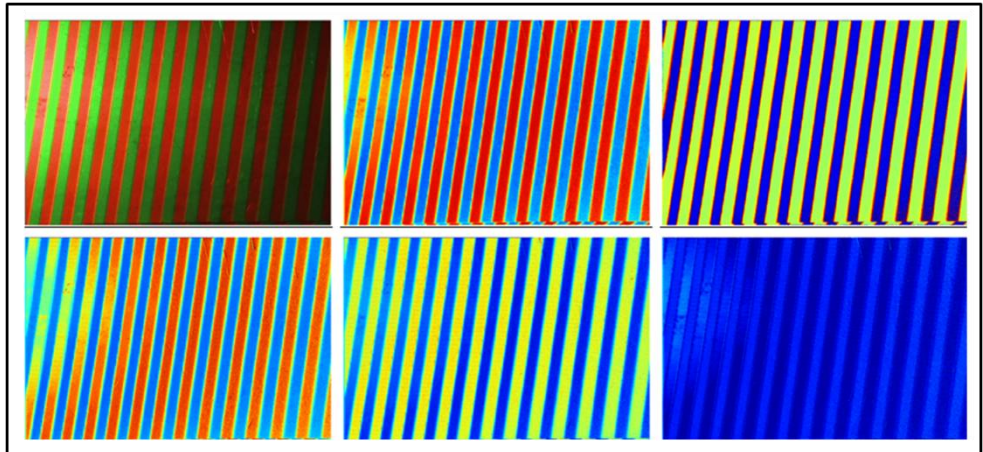


Figure A.2. Hue and normalized color results for the binary scanner. First row: from left to right shiny metal plate with red and green pattern, Hue result. Second row: from left to right normalized red and blue results

Another set of invariants that we tested is the c_4, c_5, c_6 invariants proposed by Gevers and Smeulders. They are defined as

$$\begin{aligned}
 c_4 &= \frac{R-G}{R+G} \\
 c_5 &= \frac{R-B}{R+B} \\
 c_6 &= \frac{G-B}{G+B}
 \end{aligned} \tag{A.3}$$

We give the results of these invariants together with the c_1, c_2, c_3 invariants in Figure A.3. Here, c_1 and c_2 has similar results that can be thresholded without being effected by the highlights. The effect of shiny surface can be seen more on the invariant c_4 . We should not expect a useful result from the c_3, c_5 and c_6 for a red and green colored pattern.

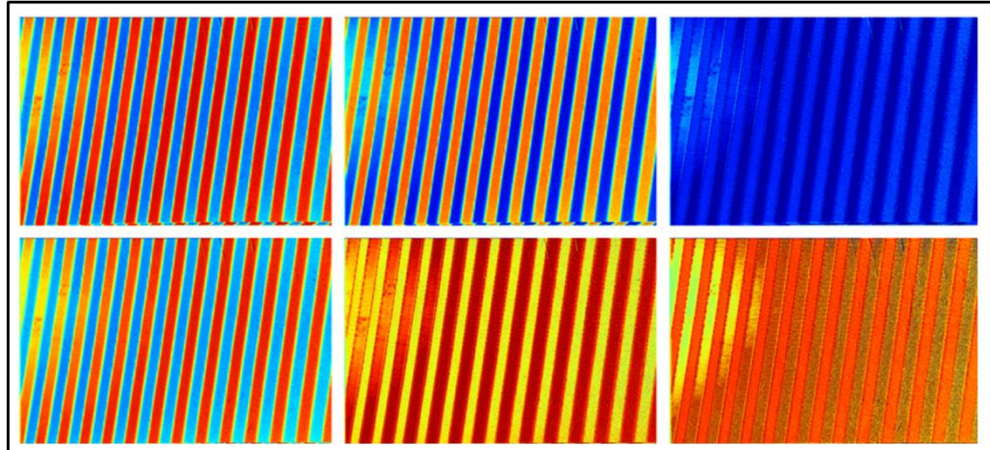


Figure A.3. The c color invariant results for the binary pattern. First row: from left to right c_1, c_2, c_3 results. Second row: from left to right c_4, c_5, c_6 results

The last set of invariants we tested is $l_1, l_2, l_3, l_4, l_5, l_6$ proposed by Gevers and Smeulders. These invariants are

$$\begin{aligned}
 l_1 &= \frac{(R-G)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2} \\
 l_2 &= \frac{(R-B)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2} \\
 l_3 &= \frac{(G-B)^2}{(R-G)^2 + (R-B)^2 + (G-B)^2} \\
 l_4 &= \frac{|R-G|}{|R-G| + |B-R| + |G-B|} \\
 l_5 &= \frac{|R-B|}{|R-G| + |B-R| + |G-B|} \\
 l_6 &= \frac{|G-B|}{|R-G| + |B-R| + |G-B|}
 \end{aligned} \tag{A.4}$$

Figure A.4 shows the results of the l color invariants. $l_2, l_3, l_5,$ and l_6 color invariants take the ratio of the difference of blue from red and green to the difference of all colors. Therefore, these color invariants does not give successful results for this pattern.

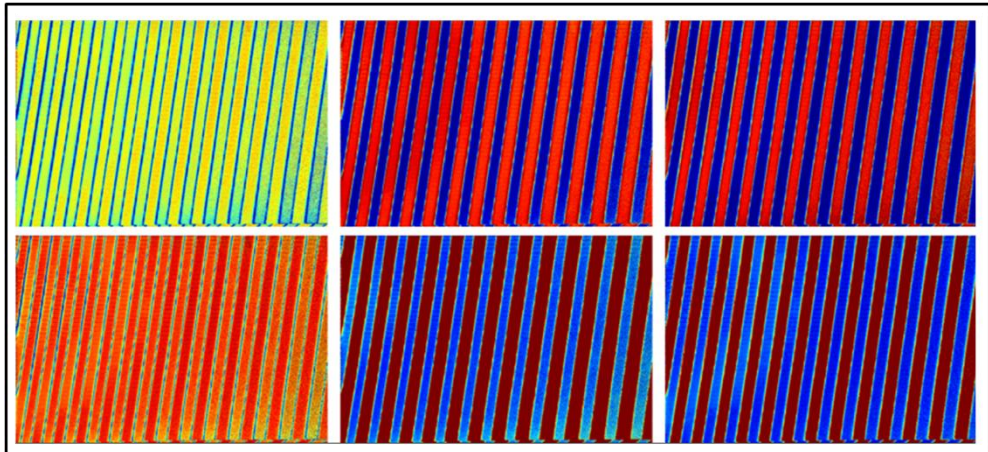


Figure A.4. The l color invariant results for binary pattern. First row: from left to right l_1 , l_2 , l_3 results. Second row: from left to right l_4 , l_5 , l_6 results

Other than these color invariants, we tested different color spaces (Lab, XYZ and xyY) as invariants [51]. The results on these color spaces are given in Figure A.5. As can be seen, Lab and XYZ are highly effected by the shiny surface and do not give a successful result. The 'y' component of the xyY space is successful on color segmentation of red and green colored pattern stripes.

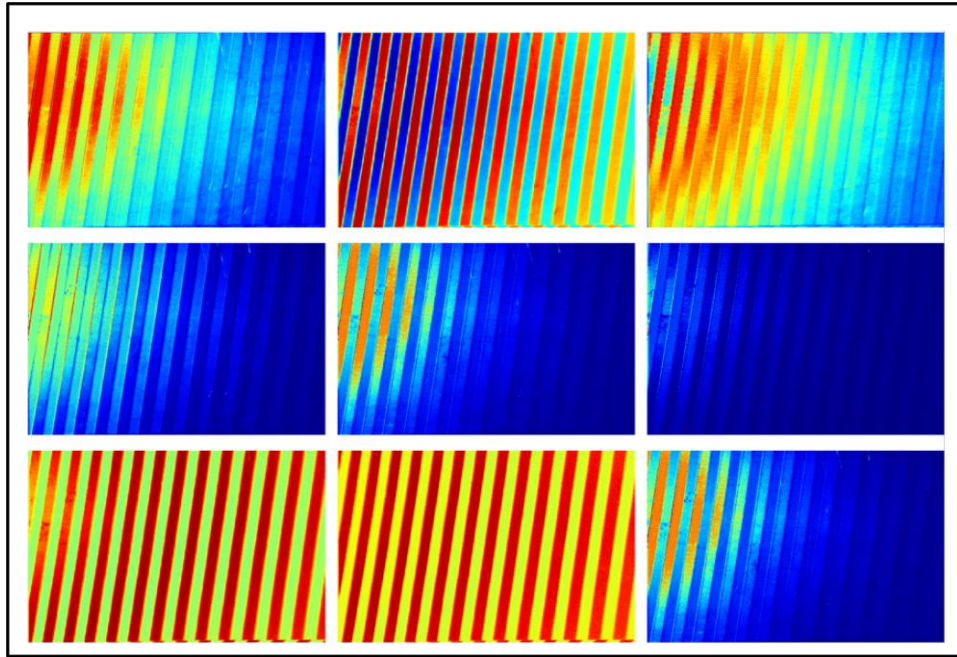


Figure A.5. Lab, XYZ and xyY color space results. First row: Lab components. Second row: XYZ components. Third row: xyY components

A.2. INVARIANTS TESTED FOR THE TERNARY CODED STRUCTURED LIGHT SCANNER

As we did in binary patterns, we tried different color invariants for the ternary coded structured light scanner. The first invariant we tried is Hue. We projected a ternary pattern with red, green, and blue colored stripes on to the metal plate object. Figure A.6 shows the pattern projected image and the Hue result for the ternary pattern.

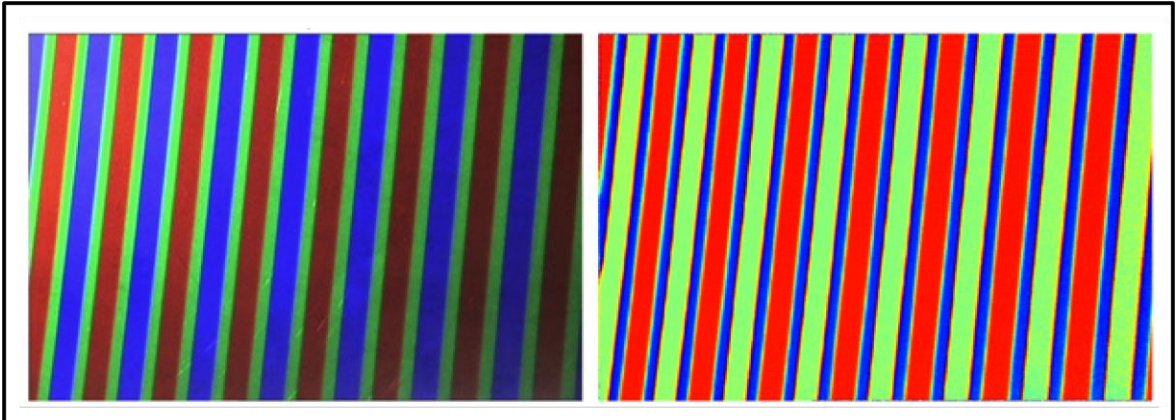


Figure A.6. Projected pattern and the Hue result

Hue gives successful result on segmentation of these three colors. Another set we tried is the normalized colors R_n , G_n , and B_n . The results for normalized colors is given in Figure A.7. Normalized colors are successful on color segmentation. However, it seems that there will be problems in thresholding on shiny surfaces.

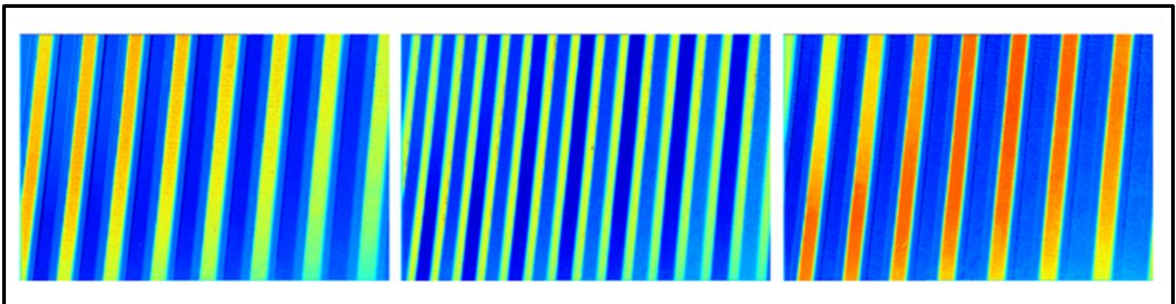


Figure A.7. Normalized color results for the ternary pattern. From left to right R_n , G_n , and B_n

As we did in the binary scanner, the second invariant set we tested is the 'c' color invariants. The c_1 , c_2 , and c_3 give successful result on segmentation especially without being effected by the shiny surface. However, we cannot say the same thing for the c_4 , c_5 , and c_6 invariants.

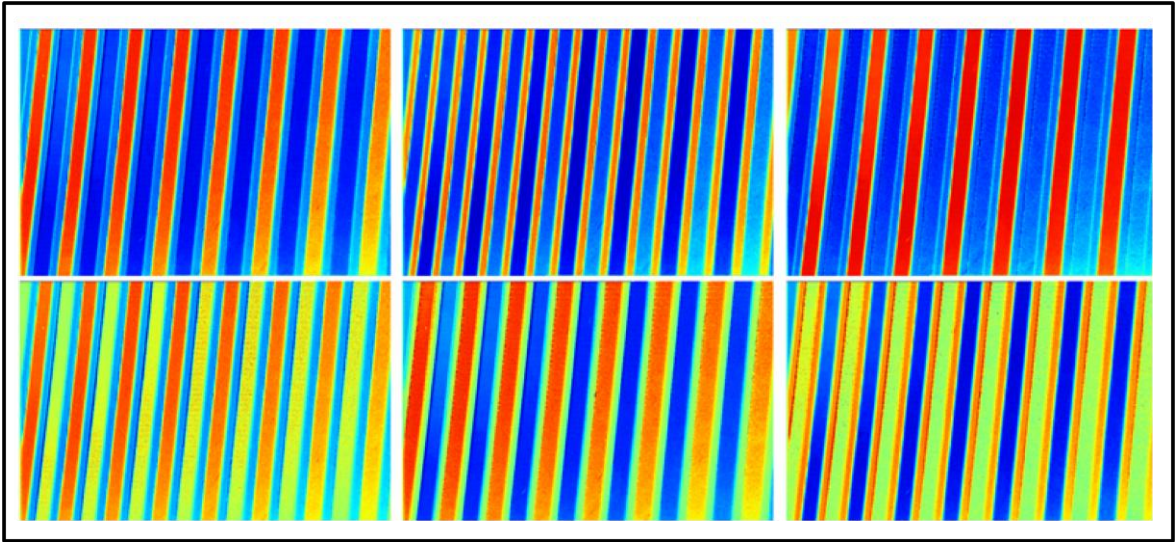


Figure A.8. The 'c' color invariant results for ternary pattern. First row: from left to right c_1 , c_2 , and c_3 . Second row: from left to right c_4 , c_5 , c_6 results

The last color invariant set we tried for ternary patterns is the 'l' color invariants. The results for the $l_1, l_2, l_3, l_4, l_5, l_6$ are given in Fig. 56. These color invariants take the ratio of the difference of two colors with respect to the sum of the difference of the all colors. Therefore, we should not expect to have a successful result for three colors by these invariants.

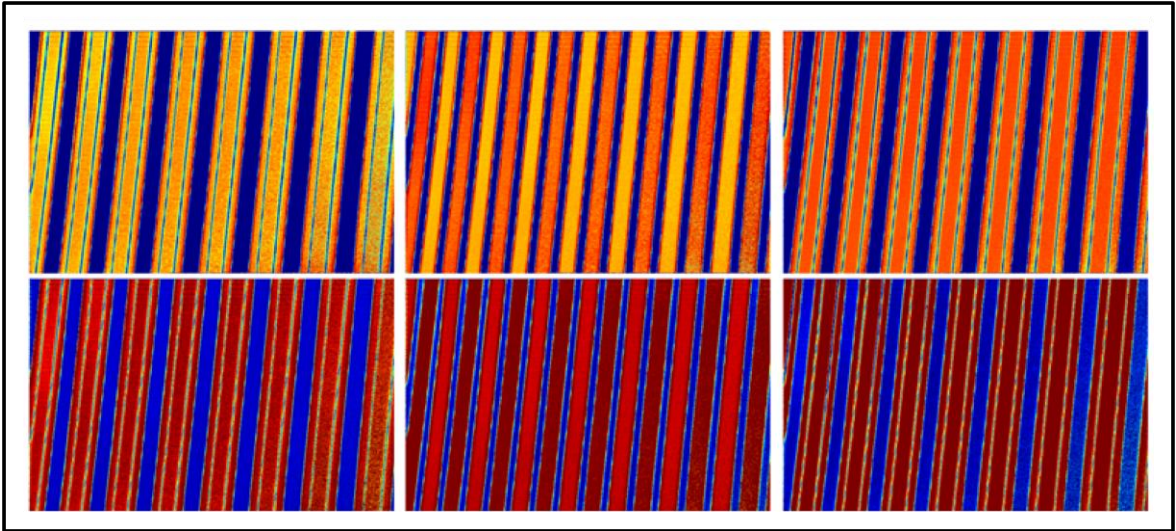


Figure A.9. The l color invariant results for the ternary pattern. First row: from left to right l_1 , l_2 , and l_3 . Second row: from left to right l_4 , l_5 , l_6 results

Finally, we give the results of other color spaces in Figure A.10. ‘L’ carries the intensity in the Lab space. Therefore its directly effected by the illumination. For the ‘a’ and ‘b’ components, we cannot separate the three colors. XYZ color space is effected by the illumination and could not give a robust threshold result. The ‘y’ component in the xyY color space can be used for the segmentation of the three colors. Although there is a short range between red and green color separation, these colors can be separated by thresholding.

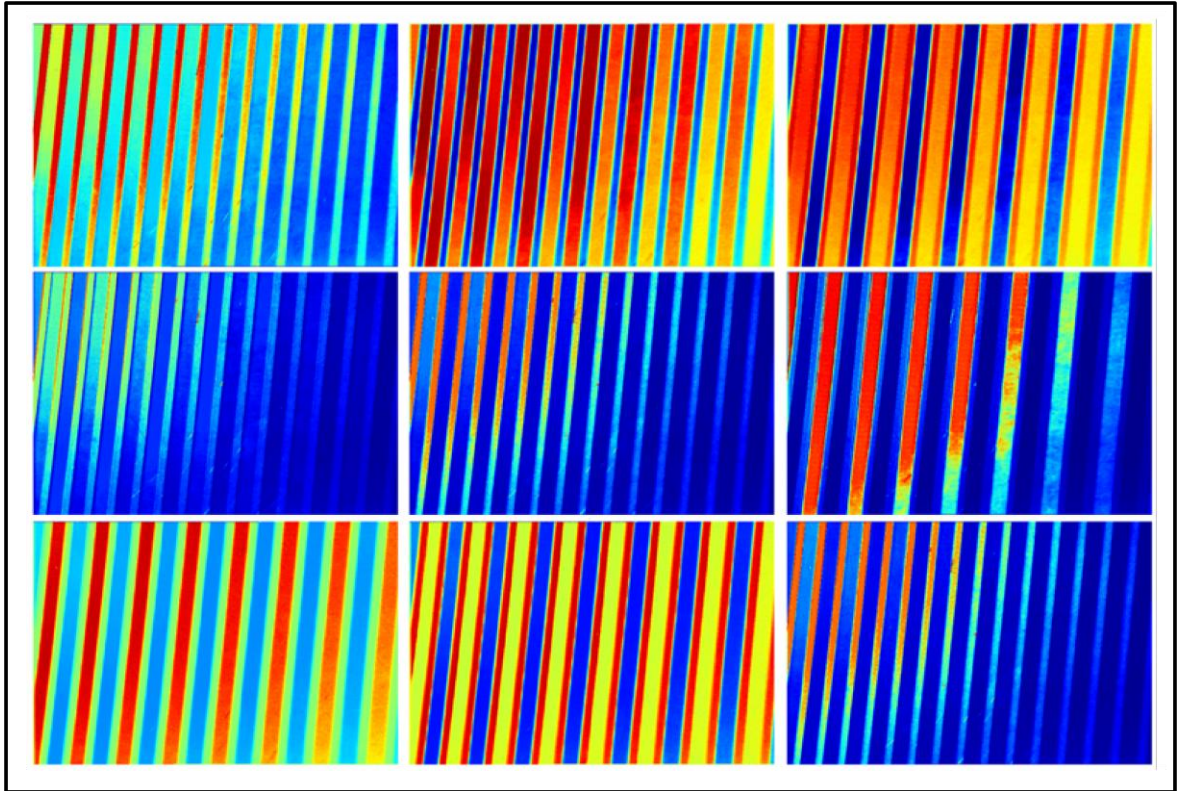


Figure A.10. Lab, XYZ and xyY color space results. First row: Lab components; second row: XYZ components; third row: xyY components

According to these experiments, we decided to use the c_1 and c_3 color invariants for our system. Since these invariants also give successful results for quaternary patterns, we did not try other invariants for quaternary patterns.

A.3. INVARIANTS TESTED FOR THE SENARY CODED STRUCTURED LIGHT SCANNER

To decrease the number of patterns to be projected to three, we should project six colors. This means we should segment out six different colors. Neither ‘c’ nor the other color invariants that we tested can segment six colors reliably. Therefore, we proposed a new color invariant based on opponent color theory for this purpose [52,53,54]. We also designed a new pattern structure, that can work reliably with the proposed color invariant.

Next, we give a brief explanation of the opponent color theory and the designed scanner. Then, we will provide the range data extracted by this method.

A.3.1. Opponent Color Theory

The opponent color space is based on human visual system that interprets the color from cones and rods. According to this theory, human visual system does not see a yellowish blue or reddish green. Therefore, the visual system interprets the color in two sets of hues yellow-blue (YB) and red-green (RG). These opponent color channels can be derived from trichromatic channels (RGB) as

$$\begin{aligned} RG &= R - G \\ YB &= 2B - R - G \end{aligned} \quad (\text{A.5})$$

Figure A.11 is a schematic diagram that shows how the cone responses are encoded into luminance (I) and the two opponent color channels.

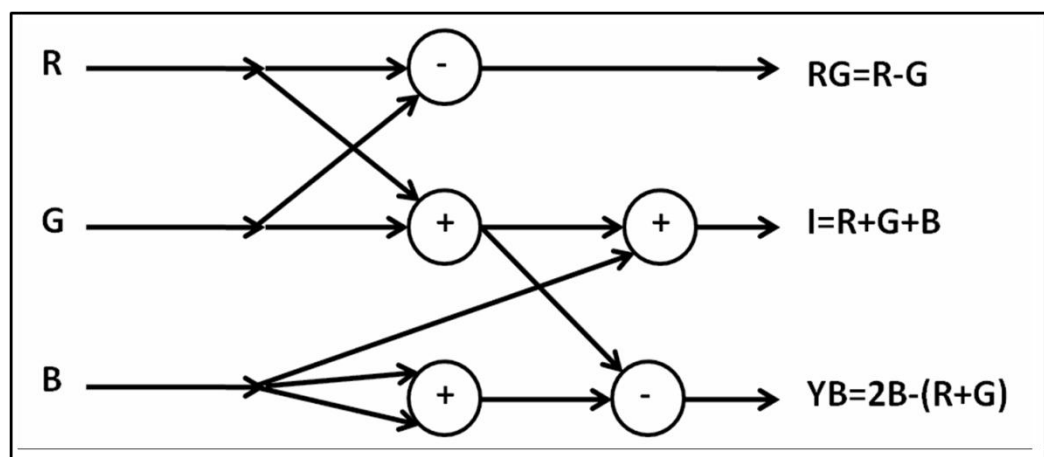


Figure A.11. Opponent-color encoding schematic

In the opponent color space, Hue can be coded in a circular format ranging through blue, green, yellow, red and black to white on two axis of RG and YB. Therefore, opponent colors space is more suitable for modeling the perceived color than RGB.

A.3.2. The Designed Pattern for Senary Coding

In order to segment out six colors, we designed the most suitable pattern to the result of opponent color space. Our pattern colors are formed by three sinusoidal signals that are shifted by $\pi/3$ radians for each channel of (RGB) colors as

$$\begin{aligned} R &= \sin(2\pi n) \\ G &= \sin\left(2\pi n + \frac{2\pi}{3}\right) \\ B &= \sin\left(2\pi n + \frac{4\pi}{3}\right) \end{aligned} \quad (\text{A.6})$$

We add these colors and divide them into six levels that give six different color values for our patterns. As in the previous scanners, we place the colored stripes by Gray coding to prevent ambiguities in cross sections of the patterns.

A.3.3. Color Segmentation for Senary Patterns

As we mentioned above, we designed our patterns to segment the six colors by using opponent color theory. If we place the three color channels of the designed pattern into the Eqns 52 these equations become

$$\begin{aligned} RG &= \sin(2\pi n) - \sin\left(2\pi n + \frac{2\pi}{3}\right) \\ YB &= 2\sin\left(2\pi n + \frac{4\pi}{3}\right) \end{aligned} \quad (\text{A.7})$$

Simplifying these, we obtain

$$\begin{aligned} RG &= -\cos\left(2\pi n + \frac{\pi}{6}\right) + \sin(2\pi n) \\ YB &= -\frac{3}{2}(\sqrt{3}\cos(2\pi n)) + \sin(2\pi n) \end{aligned} \quad (\text{A.8})$$

These are the two axes of an ellipse. In segmentation, we divide the ellipse into six regions by the threshold levels obtained experimentally. For an ideal camera image, the colors should be aligned according to the center of the image. However according to the camera color properties, these colors are shifted from the center of the ellipse.

Furthermore, pixels that correspond to shadow or highlights give a response close to the center of this shifted ellipsoid shape. Therefore, in segmentation we remove the pixels having values close to the center of the ellipse. The remaining pixels are thresholded taking the shifting into account. Figure A.12 shows the segmented colors on the matte Atatürk object.

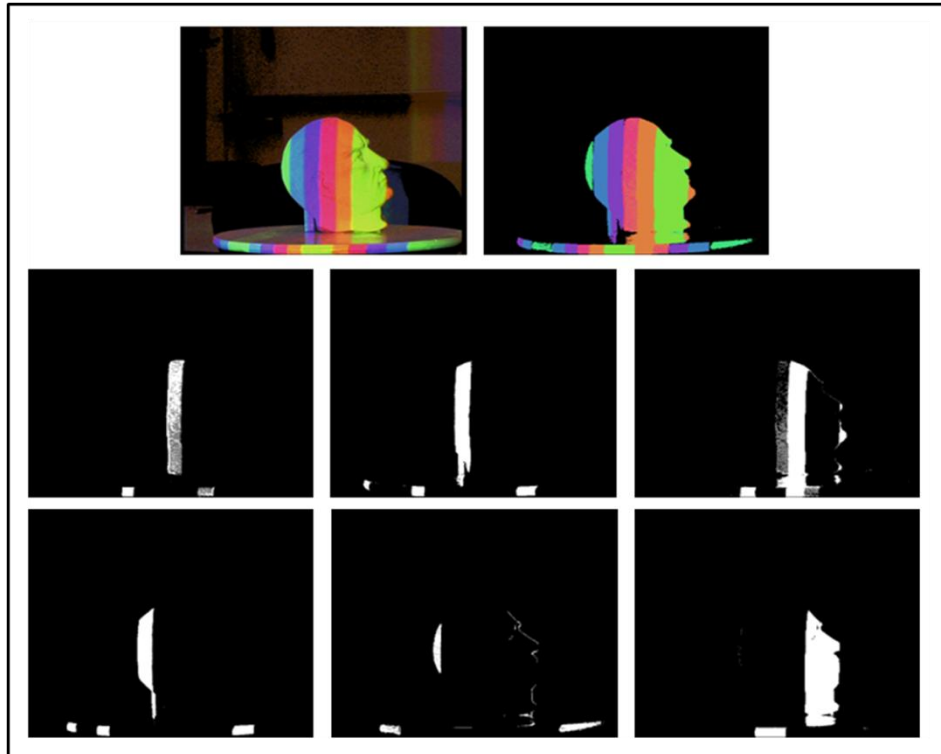


Figure A.12. Segmented six colors in the senary pattern

A.3.4. Extracted Range Data using Senary Coding

By increasing the number of colors, we had to decrease the resolution of the patterns. Therefore, by increasing the color pattern to have six colors, the resolution of the senary scanner is decreased (compared to the previous scanners). In Figure A.13, we provide the extracted range data for the senary scanner.

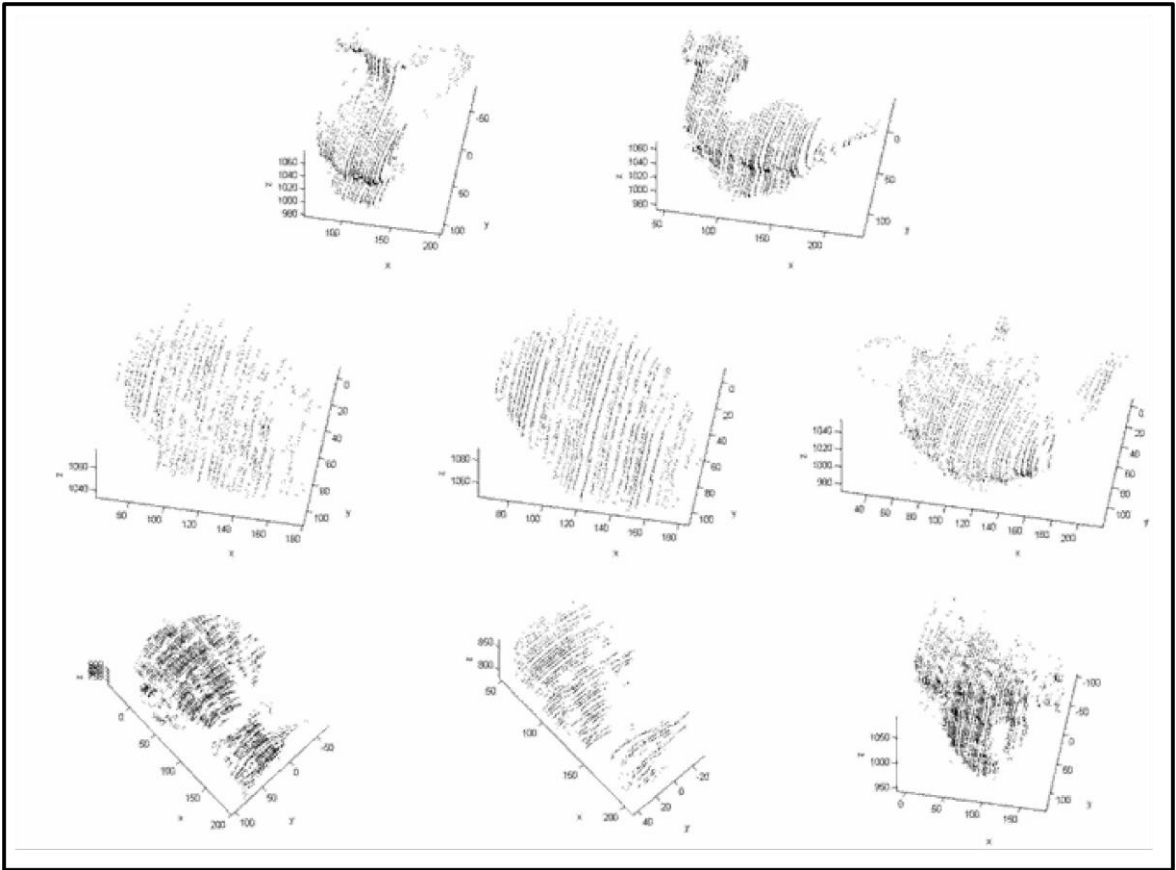


Figure A.13. Range data extracted by the senary scanner. First row: matte carafe and dove objects. Second row: shiny and matte Atatürk, and the teapot objects. Third row: Venus, Alexander, and the statue objects

Unfortunately, the resolution of the senary range scanner limited us using other test objects. Therefore, we did not include it as another scanning option in our range scanner system. However, we believe that using a high resolution camera, the resolution of this scanner may be increased and better range data can be obtained.

REFERENCES

1. Salvi, J., J. Pages and J. Batlle, “*Pattern codification strategies in structured light systems*”, *Pattern Recognition*, Vol. 37, pp. 827 – 849, 2004.
2. Salvi, J., S. Fernandez, T. Pribanic and X. Llado, “*A state of the art in structured light patterns for surface profilometry*”, *Pattern Recognition*, Vol. 43, pp. 2666–2680, 2010.
3. Zhang, S. and P. S. Huang, “*High-resolution, real-time three-dimensional shape measurement*”, *Optical Engineering*, Vol. 45, pp. 1–8, 2006.
4. Karpinsky, N. and S. Zhang, “*High-resolution, real-time 3D imaging with fringe analysis*”, *Journal of Real-Time Image Processing*, pp. 1–12, 2010.
5. Besl, P. J., “*Active, optical range imaging sensors*”, *Machine Vision and Applications*, Vol. 1, pp. 127–152, 1988.
6. Klette, R., K. Schlüns and A. Koschan, *Computer Vision: Three-Dimensional Data from Images*, Springer, 1998.
7. Sonka, M., V. Hlavac and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Thomson-Engineering, 3 edn., 2007.
8. Inokuchi, S., K. Sato and F. Matsuda, “*Range imaging system for 3-D object recognition*”, *Proceedings of International Conference on Pattern Recognition 1984*, pp. 806–808, 1984.
9. Caspi, D., N. Kiryati and J. Shamir, “*Range Imaging With Adaptive Color Structured Light*”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, pp. 470–480, 1998.
10. Bernardini, F. and H. Rushmeier, “*The 3D Model Acquisition Pipeline*”, *Computer Graphics Forum*, Vol. 21, pp. 149–172, 2002.

11. Boehnen, C. and P. J. Flynn, “Accuracy of 3D scanning technologies in a face scanning scenario”, *Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pp. 310–317, 2005.
12. Blais, F., M. Picard and G. Godin, “Accurate 3D acquisition of freely moving objects”, *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, 2004.
13. Godin, G., F. Blais, L. Cournoyer, J. A. Beraldin, J. Domey, J. Taylor, M. Rioux and S. El-Hakim, “Laser Range Imaging in Archaeology: Issue and Results”, *IEEE CVPR Workshop on Applications of Computer Vision to Archaeology ACVA’03*, p. 11, 2003.
14. Guidi, G., A. Beraldin and C. Atzeni, “High-accuracy 3-D modeling of cultural heritage: The digitizing of Donatello’s Maddalena”, *IEEE Transactions on Image Processing*, Vol. 13, pp. 370–380, 2004.
15. Park, J. and A. C. Kak, “3D modeling of optically challenging objects”, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 14, pp. 246–262, 2008.
16. Umasuthan, M. and A. M. Wallace, “Outlier removal and discontinuity preserving smoothing of range data”, *IEEE Proceedings of Vision Image and Signal Processing*, Vol. 143, pp. 191–200, 1996.
17. Elgazzar, S., R. Liscano, F. Blais and A. Miles, “Active range sensing for indoor environment modeling”, *IEEE Transactions on Instrumentation and Measurement*, Vol. 47, pp. 260–264, 1998.
18. Levoy, M., P. Brunet and R. Scopigno, “The Digital Michelangelo Project”, *EUROGRAPHICS*, 1999.

19. Fisher, R. B., D. K. Naidu and D. Singhal, “*Rejection of spurious reflections in structured illumination range finders*”, *Proceedings on 2nd Conference on Optical 3-D Measurement Techniques*, pp. 467–474, 1993.
20. Forest, J., J. Salvi, E. Cabruja and C. Pous, “*Laser stripe peak detector for 3D scanners, A FIR filter approach*”, *Proceedings of the 17th International Conference on Pattern Recognition ICPR*, 2004.
21. Koninckx, T. P. and L. Van Gool, “*Real-Time Range Acquisition by Adaptive Structured Light*”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, pp. 432–445, 2006.
22. Skocaj, D. and A. Leonardis, “*Range Image Acquisition of Objects with Non-uniform Albedo Using Structured Light Range Sensor*”, *Proceedings of International Conference on Pattern Recognition 2000*, pp. 778–781, 2000.
23. Trobina, M., *Error Model of a Coded-Light Range Sensor*, Technical report biwi-tr-164, ETH Zurich, Switzerland, 1995.
24. Zhang, S. and S. T. Yau, “*High dynamic range scanning technique*”, *Optical Engineering*, Vol. 48, pp. 1–7, 2009.
25. Xu, Y. and D. Aliaga, “*An Adaptive Correspondence Algorithm for Modeling Scenes with Strong Interreflections*”, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, pp. 465–480, 2009.
26. Chen, C. S., Y. P. Hung, C. C. Chiang and J. L. Wu, “*Range data acquisition using color structured lighting and stereo vision*”, *Image and Vision Computing*, Vol. 15, pp. 445–456, 1997.
27. Rocchini, C., P. Cignoni, C. Montani, P. Pingi and R. Scopigno, “*A low cost 3D scanner based on structured light*”, *Proceedings of Eurographics*, pp. 299–308, 2001.

28. Gevers, T. and A. W. M. Smeulders, “*Color based Object Recognition*”, *Pattern Recognition*, Vol. 32, pp. 453–464, 1999.
29. Gevers, T. and A. W. M. Smeulders, “*Content-Based Image Retrieval by Viewpoint-Invariant Color Indexing*”, *Image and Vision Computing*, Vol. 17, pp. 475–488, 1999.
30. Gevers, T. and A. W. M. Smeulders, “*PictoSeek: Combining Color and Shape Invariant Features for Image Retrieval*”, *IEEE Transactions on Image Processing*, Vol. 9, pp. 102–119, 2000.
31. Forsyth, D. and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2003.
32. Heikkila, J. and O. Silven, “*Calibration procedure for short focal length off-the-shelf CCD cameras*”, *Proceedings of the 13th International Conference on Pattern Recognition 1996*, Vol. 1, pp. 166–170 vol.1, Aug 1996.
33. Tsai, R., “*A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*”, *IEEE Journal of Robotics and Automation*, Vol. 3, No. 4, pp. 323–344, Aug 1987.
34. Zhang, Z., “*Flexible camera calibration by viewing a plane from unknown orientations*”, *Proceedings of International Conference on Computer Vision ICCV 1999*, pp. 666–673, 1999.
35. Zhang, Z., “*A flexible new technique for camera calibration*”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330–1334, Nov 2000.
36. Bouguet, J. Y., *Complete Camera Calibration Toolbox for Matlab*.
37. Unsalan, C. and K. Boyer, “*Linearized vegetation indices based on a formal statistical framework*”, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 42, No. 7, pp. 1575 – 1585, July 2004.

38. Jolliffe, I. T., *Principal Component Analysis*, Springer, 2 edn., 2002.
39. Benveniste, R. and C. Ünsalan, “*Single stripe projection based range scanning of shiny objects under ambient light*”, *24th International Symposium on Computer and Information Sciences, 2009. ISCIS 2009.*, pp. 1–6, sept. 2009.
40. Benveniste, R. and C. Ünsalan, “*A color invariant for line stripe based range scanners*”, *The Computer Journal*, Vol. 54, No. 5, pp. 738–753, 2011.
41. Benveniste, R. and C. Ünsalan, “*A binary coded structured light system to scan shiny surfaces*”, *IEEE 18th Signal Processing and Communications Applications Conference (SIU), 2010*, pp. 292 –295, april 2010.
42. Benveniste, R. and C. Ünsalan, “*A Color Invariant Based Binary Coded Structured LightRange Scanner for Shiny Objects*”, *20th International Conference on Pattern Recognition (ICPR), 2010*, pp. 798 –801, aug. 2010.
43. Benveniste, R. and C. Ünsalan, “*Binary and ternary coded structured light 3D scanner for shiny objects*”, *25th International Symposium on Computer and Information Sciences, 2010. ISCIS 2010.*, 2010.
44. Benveniste, R. and C. Ünsalan, “*Single stripe projection based range scanner implementation on TI DaVinci DM6437 EVM*”, *4th European DSP Education and Research Conference (EDERC 2010)*, Nice, France, December 2010 2010.
45. TI, *TV P5146 NTSC/PAL/SECAM 4x10 Bit Digital Video Decoder Datasheet*.
46. Kehtarnavaz, N., *Real-Time Digital Signal Processing: Based on the TMS320C6000*, Newnes Publishing, 2004.
47. Qureshi, S., *Embedded Image Processing on the TMS320C6000 DSP: Examples in Code Composer Studio and MATLAB*, Springer, 2005.

48. Rodrigues, M., R. Fisher and Y. Liu, “*On the Representation of Rigid Body Transformations for Accurate Registration of Free-Form Shapes*”, *Computer Vision and Image Understanding*, Vol. 87, pp. 1–7, 2002.
49. Rusinkiewicz, S. and M. Levoy, “*Efficient Variants of the ICP Algorithm*”, *Proceedings of Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.
50. Sertel, C., O. and Ünsalan, “*Range image registration with edge detection in spherical coordinates*”, *Lecture Notes in Computer Science*, no: 4105, pp. 745–752, 2006.
51. Poynton, C., *Digital Video and HDTV Algorithms and Interfaces*, Morgan Kaufmann Publishing, 2003.
52. Plataniotis, K. N. and A. N. Venetsanopoulos, *Color Image Processing and Applications*, Springer, 2000.
53. Thyagarajan, K. S., *Digital Image Processing with Application to Digital Cinema*, Focal, 2006.
54. Ebner, M., *Color Constancy*, John Wiley, 2007