

AUTOMATED TOLERANCE INSPECTION OF FREE FORM OBJECTS

by
Çağkan EKİCİ

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

Yeditepe University
2012

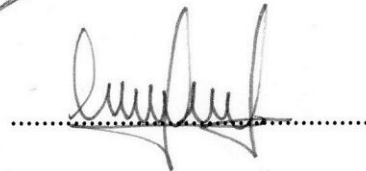
AUTOMATED TOLERANCE INSPECTION OF FREE FORM OBJECTS

APPROVED BY:

Assoc. Prof. Dr. Cem Ünsalan
(Supervisor)

A handwritten signature in black ink, appearing to be 'Cem Ünsalan', written over a horizontal dotted line.

Assoc. Prof. Dr. Duygun Erol Barkana

A handwritten signature in black ink, appearing to be 'Duygun Erol Barkana', written over a horizontal dotted line.

Assist. Prof. Dr. Dionysis Goularas

A handwritten signature in black ink, appearing to be 'Dionysis Goularas', written over a horizontal dotted line.

DATE OF APPROVAL: / /

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Assoc. Prof. Dr. Cem Ünsalan for his patient attention and guiding me throughout the thesis.

This work could not be possible without backing from my family, friends and loved ones and I express my thankfulness to them.

ABSTRACT

AUTOMATED TOLERANCE INSPECTION OF FREE FORM OBJECTS

In quality control applications it is an important task to determine if a manufactured object deviates from the design requirements. Inspection involves with measurement and tolerance check of geometric dimensions of manufactured products. There had been presented many inspection solutions for products with regular features, such as quadrangular and circular shapes. However for products with free-form surfaces manufactured by CNC machines, turn benches or plastic injection, the inspection process is still a big problem. By being end-product or sub-product, free-form surfaces are widely used in many fields such as automotive, aerospace, biomedical and machining industries. In this thesis, a vision-based inspection system is proposed for the purpose of improving the speed and the accuracy of the process. We will develop a vision based inspection system which will inspect geometrical and also free form objects within given tolerances. The system will alert if the inspected object dimensions are not in given tolerances, thus the object is faulty. The system can be used in all manufacturing environments, where tolerances of objects are important. Several matching methods were investigated and some of them were tested to major on single method. The Hausdorff distance method is found to be robust and works stable with different shape features, but the processing speed of the method is still a big problem. By considering robustness and promising features, Hausdorff Distance method was studied to improve the speed of the matching process with using different auxiliary techniques. Finally, the results of the applied techniques will be discussed to present the best solution.

ÖZET

SERBEST ŞEKİLLİ NESNELERİN OTOMATİK TOLERANS İNCELEMESİ

Kalite kontrol uygulamalarında, imal edilen üründe tasarım gereksinimlerinden sapma olup olmadığının belirlenmesi son derece önemlidir. İncelemede imal edilen ürünlerin geometrik boyutlarının ölçümü ve tolerans kontrolü gerçekleştirilir. Dörtgen veya dairesel şekilli ürünlere yönelik çok sayıda inceleme çözümü sunulmaktadır ancak CNC makineleri, torna tezgahları veya plastik enjeksiyon yoluyla üretilen serbest şekilli yüzeye sahip ürünlerin incelenmesindeki sorunlar henüz çözülmemiştir. Bu tezde ölçüm işlemlerinin daha hızlı ve daha yüksek doğrulukla gerçekleştirilmesine yönelik görüntü işleme temelli inceleme sistemi önerilmektedir. Geometrik nesnelere yanı sıra belirli tolerans aralığına sahip serbest şekilli nesnelere incelenmesine olanak tanıyan görüntü işleme temelli inceleme sistemi geliştirilecektir. Boyutları belirtilen tolerans aralığının dışında olan hatalı nesnelere için sistem uyarı verecektir. Bu sistem, nesne boyut toleranslarının önemli olduğu her türlü üretim ortamında kullanılabilir. Çeşitli eşleştirme yöntemleri incelenmiştir ve üzerinde yoğunlaşılacak yöntemin belirlenmesi amacıyla bunlardan bazıları test edilmiştir. Farklı şekil özelliklerine sahip nesnelere için Hausdorff mesafe yönteminin en sağlıklı ve kararlı çalışan yöntem olduğu gözlemlenmiştir. Bununla birlikte bu yöntemin işleme hızının düşük olması sorun oluşturmaktadır. Hausdorff mesafe yönteminin güvenilirliği ve ileride sunması beklenen özellikleri göz önünde tutularak, çeşitli yardımcı teknikler yoluyla bu yöntemdeki eşleştirme hızının artırılması üzerinde çalışılmıştır. Sonuç olarak en iyi çözümün sunulması için uygulanan tekniklerin sonuçları tartışılacaktır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	viii
LIST OF TABLES	x
LIST OF SYMBOLS AND ABBREVIATIONS	xii
1. INTRODUCTION	1
1.1 PROBLEM DEFINITION	1
1.1.1 Inspection of Free-form Objects	1
1.1.2 Automated Inspection	2
1.2. PREVIOUS WORK.....	3
1.2.1 Type of Inspection Systems	3
1.2.1.1. Non-Contact Inspection Systems.....	3
1.2.1.2. Contact Inspection Systems	4
1.2.2. Alignment Methods in Automated Inspection Systems	6
1.2.2.1. Principal Axis Method	6
1.2.2.2. Polar Transform Method.....	8
1.2.2.3. Iterative Closest Point Method	9
1.2.2.4. Hausdorff Distance Method.....	11
2. SOFTWARE OF THE PROPOSED SYSTEM.....	15
2.1. IMAGE PREPROCESSING.....	15
2.1.1. Image Acquisition.....	15
2.1.2. Color Space Conversion	15
2.1.3. Shape Feature Extraction	16
2.1.4. Image Enhancement.....	16
2.1.5. Boundary Detection	17
2.2. SHAPE ALIGNMENT	17
2.2.1. Hausdorff Distance	18
2.2.2. Improved Hausdorff Distance.....	18

2.3. INSPECTION	20
2.3.1. Dimensional Measurements.....	20
2.3.2. Shape Tolerance Inspection	20
3. HARDWARE OF THE PROPOSED SYSTEM	22
3.1. CAMERA	22
3.2. LENSE	25
3.3. ILLUMINATION	26
3.4. PROCESSOR.....	28
4. IMPLEMENTATION ISSUES	29
4.1. SYSTEM SETUP	29
4.2. TRAINING THE OBJECT.....	30
4.3. PROGRAM OPERATION FLOW	30
5. EXPERIMENTS	35
5.1. Tests on objects with no fault	37
5.2. Tests on faulty objects	39
5.3. Software Speed Tests.....	56
6. CONCLUSION.....	58
REFERENCES	59
APPENDIX A: ALGORITHMS	63

LIST OF FIGURES

Figure 1.1. A vision based inspection system.....	4
Figure 1.2. A Coordinate Measuring Machine	5
Figure 2.1. Laplacian Filter Mask.....	17
Figure 3.1. UI-1245LE camera	24
Figure 3.2. Specifications of UI-1245LE camera	24
Figure 3.3. 16mm Lens	25
Figure 3.4. Backlighting	27
Figure 3.5. System Setup	29
Figure 5.1. Object to be inspected.....	35
Figure 5.2. Reference Image and its edges	36
Figure 5.3. Inspection regions and measurements on test image1.....	36
Figure 5.4. Hausdorff distance between two shape points.....	37
Figure 5.5. Maximum distances and mean distances of good objects	39
Figure 5.6. Maximum distances and mean distances of faulty objects.....	43
Figure 5.7. Before fine tuning alignment.....	44

Figure 5.8. After fine tuning alignment	45
Figure 5.9. Inspection and measurements	46
Figure 5.10. Reference images of other test objects	47
Figure 5.11. Mean distances of good objects (object2)	48
Figure 5.12. Maximum distances and mean distances of faulty objects (object2)	49
Figure 5.13. Mean distances of good objects (object3)	50
Figure 5.14. Maximum distances and mean distances of faulty objects (object3)	51
Figure 5.15. Mean distances of good objects (object4)	52
Figure 5.16. Maximum distances and mean distances of faulty objects (object4)	53
Figure 5.17. Mean distances of good objects (object5)	54
Figure 5.18. Maximum distances and mean distances of faulty objects (object5)	55

LIST OF TABLES

Table 5.1.	Results of alignment experiments for object1	38
Table 5.2.	Iterations and results for image11	40
Table 5.3.	Iterations and results for image12	40
Table 5.4.	Iterations and results for image13	41
Table 5.5.	Iterations and results for image14	42
Table 5.6.	Iterations and results for image15	42
Table 5.7.	Distance errors before and after fine tuning algorithm (object1)	43
Table 5.8.	Results of alignment experiments for object2	48
Table 5.9.	Distance errors before and after fine tuning algorithm (object2)	49
Table 5.10.	Results of alignment experiments for object3	50
Table 5.11.	Distance errors before and after fine tuning algorithm (object3)	51
Table 5.12.	Results of alignment experiments for object4	52
Table 5.13.	Distance errors before and after fine tuning algorithm (object4)	53
Table 5.14.	Results of alignment experiments for object5	54
Table 5.15.	Distance errors before and after fine tuning algorithm (object5)	55

Table 5.16. Time consumption of alignment process for 360 Angle Trials	56
Table 5.17. Time consumption of alignment process for 56 Angle Trials	57

LIST OF SYMBOLS AND ABBREVIATIONS

$d(\vec{p}_1, \vec{p}_2)$	Euclidian distance between two points
(g_x, g_y)	Centroid of the image
$\vec{h}(A, B)$	Directed Hausdorff distance from A to B.
$h_m(A, B)$	Mean Hausdorff distance
N	Number of points in the point set
P	Point set
R	Rotation Matrix
θ	Rotation Angle

1. INTRODUCTION

1.1. PROBLEM DEFINITION

1.1.1. Inspection of free-form objects

Free-form geometrical shapes have begun to be used frequently in many areas today. Because of the production difficulties of free form models, quality control process is a need to confirm if the object is produced in desired geometrical criteria. According to the material content that the free form object was made, production type of the object is either plastic injection forming or CNC machines. Free form models are used especially where aesthetics have great importance or advanced engineering designs are needed. Automotive sector and sculpturing are some examples of aesthetic based usage areas of free form objects while aerospace or defense industries are the examples of advanced engineering designs.

For products with free-form surfaces, such as marine propellers, the complex sculptured surfaces are produced with extremely high fidelity to the original design. As an example study of free-form surfaces, Jinkerson *et al.* proposed methods for the inspection and feature extraction of marine propellers [1]. The inspection of sculptured surfaces is essential since many products with sculptured surfaces are designed and manufactured with a requirement for high precision. Menq *et al.* also studied on free form surfaces that they presented method for precision measurement of surface profile [2].

It is an important task to ensure that the object is produced in its standard design parameters which the precision inspection of parts with free-form surfaces is becoming increasingly critical. Although there had been presented many techniques for tolerance inspection of geometrical objects, inspection of free-form objects is still a challenging topic regarding high speed and robustness prospects of manufacturers.

1.1.2. Automated Inspection

Today, with rising competition in global production market, time and cost parameters have gain great importance. Automation systems are the use of machines and information technologies together to improve manufacturing in the terms of time and cost optimization. Machine vision based inspection systems are the part of automation systems that take role in the quality control phase. As machine vision systems are fast and reliable systems, they are capable of processing much more products as compared with the classical human based inspection studies.

In the consideration of tolerance inspection studies, automated alignment of object is the main objective as manually inspection systems are time consuming. Newman and Jain performed a survey of automated visual inspection techniques [3]. For the system being automated, first, the object shape must be aligned to be ready for the inspection phase and then the target object shape must be compared with the reference object shape. Alignment plays the most important role in the automated inspection. Alignment is the determination of the position and orientation of an object according to reference object that provides rotation and translation invariant inspection that under all different position conditions the software enables the precise measurement.

There has been presented some methods like Principal axis, Iterative closest point, Hausdorff distance and Polar transform that deals with the alignment process of inspection application. However, process time and accuracy is still a big problem. While some methods provide good accuracy among different shape forms they suffer from lack of speed. Thus, position invariant inspection is still one of the big challenges in measurement and tolerance inspection studies.

1.2. PREVIOUS WORK

1.2.1. Type of Inspection Systems

As considering the technique of collecting data, the inspection systems divided into two types: Non-Contact Inspection Systems and Contact Inspection Systems. Non-contact inspection systems such as machine vision systems and scanning laser systems collect data by imaging technologies and reflecting of light photons, respectively. Contact inspection systems such as Coordinate Measuring Machines collect the data by touching probe with its own coordinate system.

1.2.1.1. Non-Contact Inspection Systems

There are two kinds of non-contact inspection systems: Machine vision and laser scanning. Machine vision deals with images or sequences of images with the objective of analyzing them for the industrial application manner. It describes the understanding of technically obtained images for controlling production processes. Machine vision technology is an interdisciplinary technology that combines electronics, optics and software engineering. One of the typical applications of machine vision technology is inspection systems. A machine vision inspection system is composed of camera, lighting, processor and appropriate software for purpose specific. Machine vision inspection system is based on analyzing the image of the object to be inspected. The system decides manufactured object is produced whether within given tolerances or not.

For an example of industrial developments on vision inspection systems, measurement instruments manufacturer company, KEYENCE, has newly presented a vision based dimensional inspection system IM6500.



Figure 1.1. A vision based inspection system

Other non-contact inspection system is laser scanning. Laser scanners work with the principle of reflecting light photons. Laser systems scan the predetermined path as sending light to the target object surface and receiving them back to generate measurement data. The data provided from the laser scanning system can be used for many applications such as tolerance inspection of free form objects. Compared with the vision systems, laser scanning can provide very accurate position measurements of the components however they are suffered from high cost and low speed.

1.2.1.2. Contact Inspection Systems

Coordinate Measuring Machine (CMM) is a measuring device that acquires surface geometric information by physically touching the parts using tactile sensors such as probes. The probe is the part of the CMM which is rigidly attached to a movable component of the CMM. When contact with the object to be inspected occurs, the coordinates of the contact point are computed.



Figure 1.2. A Coordinate Measuring Machine

CMM has the advantages of high accuracy, repeatability and reliability that make it the main tool for part validation in manufacturing. The measured data is used with various algorithms to determine positions, orientations and dimensions of objects. CMM usually acquires data using a touch trigger probe that contacts individual points on a work-piece. It can be used to accurately measure objects with widely varying size and geometric configuration, and provide the relationship between the features of a work-piece. Thus, it does not require clean surfaces or special illumination, whereas a vision system always does. With the use of position free probes, CMM can inspect surfaces that a light beam cannot reach or a camera cannot acquire the appropriate image. However, CMM is a low speed method for inspection that makes it impossible to measure many points on the object. The part needs to be stationary and carefully placed and they have a slower measuring speed than optical systems. One other disadvantage of CMM is the programming because it is a manual system and needs a highly trained operator usage.

Vision Guided Coordinate Measuring Machines is a hybrid system that combines both vision system and CMM functionalities to improve the process speed and accuracy. The system is based on CMM that image processing algorithms only provide an enhancement through working capabilities of CMM.

There are some applications on integrating multiple sensors and vision probes with CMM in order to achieve high measuring quality and speed [4], [5], [6]. Global information generated by the vision systems was used to guide the movement of the touch probe. Vision provided information about the positions of part features of interest, and then the probe was guided to the features to make actual measurements.

1.2.2. Alignment Methods in Automated Inspection Systems

In order to implement feature based matching, the image features must initially be extracted. After the features are extracted, the attributes of the features are compared between two images. The feature pair having the attributes with the best fit is recognized as a match. Researchers have proposed several techniques based on Principal Axis Method, Polar Transform Method, Iterative Closest Point Method and Hausdorff Distance Method to improve the speed and robustness of the matching utility. Matching of two shapes is the main part of the inspection process thus some researchers who deals with matching studies also deals with inspection studies. There have been presented many tolerance inspection techniques in recent years [7]. The methods based on ICP and Hausdorff distance can also handle inspection process as they are also used in alignment phase [8]. Unlike these methods, implicit polynomials were also used in tolerance inspection process [9] with prior alignment constraint.

1.2.2.1. Principal Axis Method

Principal axes of a given shape can be uniquely defined as the two segments of lines that cross each other orthogonally in the centroid of the shape and represent the directions with zero cross-correlation. Ellipses are generally used with principal axis method as they provide a useful representation of objects. Since they are more convenient to manipulate than the corresponding sequences of straight lines needed to represent the curve, and their detection is reasonably simple and reliable. Thus they are often used by computer vision systems for model matching.

Over the years much attention has been paid to fitting ellipses to data samples, and many variations of the standard method for finding the least squares (LS) solution exist. Gander *et al.* [10] surveyed the Gauss-Newton method to solve the nonlinear least squares

problem. Their experiments resulted that all algorithms are prohibitively expensive compared to the simple algebraic solution. If the problem is well posed, and the accuracy of the result should be high, the Newton method applied to the parameterized algorithm is the most efficient. The odr algorithm purpose optimizing scheme is competitive with algorithms specifically written for the ellipse fitting problem. If one takes into consideration further, that we didn't use a highly optimized odr procedure, the method of solution is surprisingly simple and efficient. The varpro algorithm seems to be the most expensive. Reasons for its inefficiency are that most parameters are non-linear and that the algorithm does not make use of the special matrix structure for this problem.

Stojmenovic and Nayak [11] dealt with ellipse fitting and measuring shape ellipticity. They proposed a method to measure how elliptical a finite set of point is. Most other ellipticity measures are area-based therefore are linked to closed curve. Their algorithm has the edge on works on both open and closed curves. This method can also be guaranteed to return an ellipse, work with open and closed curves, and meaningful number in the interval.

Rosin [12] proposed a method which fit ellipse to curve data. This technique used for accumulate ellipse hypotheses as minimal subset method. Regrettably this method has some imperfections as involving the treatment of circular parameters, statistical efficiency and correlation between the five parameters. He introduced solutions to these problems and describes some variations on the theme of robust ellipse fitting. He presented certain subjects related the sampling of points to form the minimal subsets. Essential parameter set used to generate the contaminated data sets to calculate deviation in the parameter estimates. In addition to this many of the fits which produced low scores according to this criterion still represent the data decently.

Yu *et al.* [13] proposed the ellipse fitting problem is formulated and significant algorithms are surveyed. They introduced an objective function based on the geometric definition of ellipse is performed and it is amplified to three ellipse fitting algorithms. They used Penalized Objective Function, Axial Guided Ellipse Fitting and Weighted Objective Function results for a spheroid fitting algorithm. They defined a series of experiments which synthetic data has been used for the simulations in different settings to demonstrate the efficacy of the algorithm.

Principal axis method is used in industrial applications frequently owing to its speed and implementation advantages. Principal Axis methods like ellipse fitting are very fast but may be unstable with objects of unfavorable proportions.

1.2.2.2. Polar Transform Method

Polar coordinate system is a two dimensional coordinate system in which a point in two dimensional space is described by distance and angle values according to the origin. A point that is described in Cartesian coordinate system x and y can be converted to polar coordinates r and θ with $r \geq 0$ and θ in the interval $(-\pi, \pi]$ by:

$$r = \sqrt{y^2 + x^2} \quad (1.1)$$

$$\theta = \text{atan2}(y, x) \quad (1.2)$$

Log-polar coordinates is a coordinate system in two dimensions, where a point is identified by two numbers, one for the logarithm of the distance to a certain point, and one for an angle. Log-polar coordinates are closely connected to polar coordinates, which are usually used to describe domains in the plane with some sort of rotational symmetry. An invariant shape representation can be formed using the log-polar mapping.

There have been proposed some techniques based on Polar Transform method, one of them is Adaptive Polar Transform, the translation parameter between the two images is determined. Fourier phase correlation is used to fix the translation before calculating the log-polar matching [14, 15] in the frequency domain.

Koroutchev and Korutcheva [16] introduced a method which has criteria to choose figures suitable for coding and easy recognition are formulated. They analyzed complexity of the criteria. Their experiments show that the information can be coded using the orientation of the specially designed printed figures, based on the first geometrical moments of the figures. They calculated the errors in the scanning and decoding of the figures can be kept within reasonable limits. They achieved coding of the coordinates by using random coding of the printed pattern. They optimized the code thus the length can be just one figure more than the optimal coding length if the scanning is errorless.

Matungka *et al.* [17] designed an algorithm that registers two images to occlusion and alteration in addition to scale, rotation and translation. They introduced a technique based on Adaptive Polar Transform (APT) in the spatial domain that samples the image. They used the projection transform to the transformed image to reduce the image 2-D to 1-D vector. They designed a new algorithm that uses the scale and rotation invariant feature point to eliminate the detailed search for all the possible translation of the model image. Their algorithm works the image comparison scheme in the projection domains that is designed for locating the areas that are subjected to occlusions and alterations in the image. Their method uses the innovative projection transform to reduce the dimensions and sampling the image in the Cartesian coordinates.

Pan *et al.* [18] proposed a new method which for calculating both the polar and the log-polar Fourier transforms in two dimensions. The algorithm is also usable for higher dimensions. They named the algorithm Multilayer Fractional Fourier Transform (MLFFT). Their algorithm has an interpolation process from a multilayer method to the real polar or log-polar grid. MLFFT has advantages over the pseudo polar-based image registration as high accuracy in recovering large scale factors and large rotation angles, adaptability, for different precision requirements, working well with both the log-polar and the polar transforms, easy implementation and fast and parallel able computing with just serial fractional FFT algorithms.

1.2.2.3. Iterative Closest Point Method

Iterative Closest Point (ICP) is an algorithm that is used for registration process of two images which is first proposed by Besl and McKay [19] ICP algorithm iteratively updates the translation and rotation parameters needed to minimize the distance between the two point clouds. ICP is one of the well-known algorithms for alignment methods as there are many modified techniques in literature which are based on ICP algorithm [20].

The goal of the ICP algorithm is to find the transformation parameters, for which the error (mostly least squares) between the transformed data shape points and the closest points of the model shape gets minimal. This characteristic can be divided to six stages:

1. Selection of some set of points in one or both meshes.

2. Matching these points to samples in the other mesh.
3. Weighting the corresponding pairs appropriately.
4. Rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs.
5. Assigning an error metric based on the point pairs.
6. Minimizing the error metric.

The primary advantages of most ICP based methods are simplicity and relatively quick performance when implemented with kd-trees for closest-point look up. However initialization is the critical issue that most of the ICP algorithms we have searched were time consuming. The drawbacks include the implicit assumption of full overlap of the shapes being matched. One other disadvantage is the theoretical requirement that the points are taken from a known geometric surface rather than measured.

Zinßer *et al.* [21] dealt with to estimate the scale factor within the ICP algorithm. Their method based on to find correct estimate of the scale factor in a correct registration. In addition their method allows the simultaneous use of wide range of other extensions to the ICP algorithm. They introduced a solution for simultaneous estimation of rotation, translation and scale factor. Their algorithm used for aligning two differently scaled 3-D point sets in every iteration successfully.

Kaneko *et al.* [22] proposed a method based on the iterative closest point algorithm. Their algorithm extended by M-estimation. They focused on the problem of robustly matching three dimensional contours of rigid bodies with no additive measurement but only depth data. They introduced the improved ICP algorithm. Their method used of the real contour data with ill-conditions in comparison with the original ICP method.

Yang *et al.* [23] proposed Random Sample Consensus (RANSAC) matching algorithm utilizing a multi-scale representation of range image. Their algorithm solves the problem of registration and segmentation of range image. They introduced a method which takes into account data association uncertainty simultaneously in the RANSAC paradigm. They used the algorithm to overcome a range of limitations possessed by least squares approaches and poor degradation to outliers.

Gelfand *et al.* [24] focused on a technique for identifying whether a pair of meshes will be unstable in the ICP algorithm by estimating the covariance matrix from a sparse uniform sampling of the input. They used this technique for minimizing instability by drawing a new set of sample points primarily stable areas of the input meshes. They dealt with translational and rotational uncertainties in registration. They introduced a method which uses a point selection technique that improves geometric stability of the ICP algorithm. They achieved to provide the best convergence of the algorithm to the correct pose by using sample of the input meshes.

1.2.2.4. Hausdorff Distance Method

In two dimensional Euclidean plane, if $\vec{p}_1 = (x_1, y_1)$ and $\vec{p}_2 = (x_2, y_2)$ then the distance between two points is given by:

$$d(\vec{p}_1, \vec{p}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1.3)$$

The Hausdorff distance is a measure of the maximum of the minimum distances between two sets of objects. For a set $A = \{a_1, \dots, a_p\}$ and $B = \{b_1, \dots, b_p\}$ Hausdorff distance can be defined as:

$$H(A, B) = \max(h(A, B), h(B, A)) \quad (1.4)$$

with the directed Hausdorff distance defined as:

$$\vec{h}(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \quad (1.5)$$

This distance proved to be an efficient and robust measure of similarity between two shapes. Its robustness to noisy and incomplete objects makes it suitable for using it for inspection of faulty objects. With regarding these attributes, The Hausdorff distance is commonly used in similarity determination and registration of two shapes however the computation of classical Hausdorff distance is very time consuming. Some techniques were proposed to improve the speed of the algorithm.

Huttenlocher *et al.* [25] proposed algorithms for computing Hausdorff distance between all relative positions of a binary image and a model. They focused on matching process of two images. They studied to improve the techniques to rigid motion. The method is quite tolerant of small position errors. The algorithms they have proposed was not tested on shape features, thus algorithm efficiency may become unsatisfactory for inspection studies.

Rotter *et al.* [26] focused on simplifying the computation of the Hausdorff distance. They proposed a method which allows for a given set of pixels to check whether it is sufficient to compute the Hausdorff distance using only the boundary pixels. They also present a method to eliminate a part of the contour for improving the speed of the algorithm.

Rucklidge [27] proposed a method for efficiently searching a space of transformations of a model to find transformations that minimize the Hausdorff distance between the transformed model and an image. Hausdorff distance was used for locating an affine transformation of a model in an image. He proposed a method to locate all transformations of the model that satisfy two quality criteria that can also efficiently locate only the best transformation. The experiments were performed for matching an image part to the original image. He presented a hierarchical search method that is guaranteed to produce the same results as an exhaustive search. The search techniques which can be performed in parallel used to locate the best transformation at high speeds.

Chetverikov and Khenokh [28] proposed a fast and robust method for a shape defect detection problem. The method is applied to inspection of ferrite cores. A modified mean Hausdorff distance was used for determining the error. Target shape was positioned on the reference shape by minimizing the error. Then measurement and inspection algorithms applied to check whether the object is faulty or not. Distance transform method was used for speeding up the process computation time.

Alt *et al.* [29] Proposed algorithms for computing Hausdorff algorithm which geometric objects are represented by finite collections of k -dimensional simplices in d -dimensional space. More efficient algorithms for special cases like sets of points, line segments or triangulated surfaces in three dimensions were also presented.

Nutanong *et al.* [30] proposed three algorithms which utilize hierarchical indexes and the branch and bound search principal. They introduced a method that to compute Hausdorff distance between two point sets and browses trajectories in increasing order of Hausdorff distance. They analyzed a method which proposed a baseline based and two basic branch and bound algorithms. They compared their proposed method with these three algorithms. Consequentially their method exceed in terms of the traversal cost, priority queue maintenance cost, distance calculation cost and the total execution time.

Agarwal *et al.* [31] proposed a method which adopt Hausdorff distance and extend it to sets of non-point objects and apply it to several variants of the shape matching problem, with and without constraints on the allowed transformations. Their method related to minimizing Hausdorff distance between sets of points, disks and balls. They studied two main topics, one of them to compute exactly or approximately the smallest Hausdorff distance over all possible rigid motions and the other one to approximate efficiently the best Hausdorff distance under certain transformations when partial matching is allowed.

Tang *et al.* [32] proposed a novel algorithm which to compute the Hausdorff distance between complicated polygonal models at interactive rates in real-time. Their algorithm approximates the distance within a user specified error bound. The algorithm based on to calculate tight upper and lower bounds to the exact Hausdorff distance value and then it refines these bound by polygon subdivision until the error bound is obtained. They proved inclusion properties related to Hausdorff distance measures, and utilized these properties to perform efficient bounding volume hierarchy (BVH) culling on the input models. Thus, the algorithm is able to calculate Hausdorff distance for polygon-soup models consisting of tens of thousands of triangles in real-time. In addition their algorithm is able to calculate a similarity between polygonal models of shape analysis and also the algorithm is able to compute penetration depth (PD) efficiently for physically-based animation.

Aspert *et al.* [33] introduced a method to evaluate the distance between 3D models, similar to Metro. They proposed Hausdorff distance application to distance measurements between 3D models have been introduced. In addition they studied an efficient implementation of the Hausdorff distance for triangular meshes. They compared the method with Metro. Consequentially, Mesh is fast, memory efficient and provides stable distance measures.

Alt and Scharf [34] introduced an algorithm for the computation of the Hausdorff distance between sets of plane algebraic rational parametric curves. They studied on general curve sets, including the parametric curves up to the fourth degree. The computation accuracy of the implemented software depends on the underlying algebra system. They examined the appropriate detection mechanisms and handling procedures.

2. SOFTWARE OF THE PROPOSED SYSTEM

For a typical gauging application, there are three phases to achieve the solution. First, image must be captured and preprocessed for to be ready for the core processes. Then, in the alignment phase, object must be re-positioned with according to position and orientation of reference object. After alignment was done, in the third phase, the target object dimensions are compared with the reference object's as if it was manufactured in the tolerance interval.

2.1. IMAGE PREPROCESSING

2.1.1. Image Acquisition

The first stage of any image processing software is the image acquisition stage. After the image has been obtained into the physical memory of processor, various processing techniques can be applied. In this study, image acquisition process was done with OpenCV functions. USB interface camera was used in the system setup thus no external image acquisition hardware was needed. When the target object was placed on the inspection area, software was triggered manually by pressing an assigned button. The video frame was captured as a Bitmap or JPEG image and saved to a pre-created directory. Then the image was loaded to the software as an `IplImage` structure, thus ready to be processed.

2.1.2. Color Space Conversion

In image processing, a color image is encoded in memory with three layers: red, green, and blue (RGB). RGB images store color information using 8 bits each for the red, green, and blue planes. Before processing the feature extraction algorithms, image must be converted to grayscale image format.

A grayscale image is composed of a single plane of pixels. Each pixel is encoded using one of the following single numbers:

- An 8-bit unsigned integer representing grayscale values between 0 and 255
- A 16-bit signed integer representing grayscale values between $-32,768$ and $+32,767$

In this study, OpenCV function `cvCvtColor` was used for grayscale conversion of the image. The grayscale image encoded with 8 bit unsigned integer. Thus, image pixel values differ between 0 and 255 that 0 is representing black and 255 is representing white.

2.1.3. Shape Feature Extraction

Shape feature extraction plays an important role in shape alignment and registration processes. Thresholding is one of the methods that segment an image into two regions, as object region and background region. Thresholding works by setting 0 (zero) to all pixels below a gray-level value which is called threshold value, and setting all other pixels in the image to 255 if the image is encoded with 8 bits. Then, background region of the image appear to be white, and foreground (object) is black.

In this study, regarding the system illumination, threshold value was selected to be 80, thus the algorithm sets 0 to all pixels below the value 80 and all other pixels to 255.

2.1.4. Image Enhancement

Image enhancement is the pre-process application that improves the quality of the image by manipulating it with appropriate software algorithms. There are many enhancement methods that differ along the usage of the application. In this study, it is used for eliminating the noise that could occur because of the dust particles or scratches on the inspection area. In the other words, it is used for cleaning the unwanted pixels from the background region. The algorithm was developed in the manner of recognizing fewer neighbor pixels than a pre-assigned value; it removes them by converting them to the same color as background. For this occasion, if there are fewer than 10 black pixels bonded with each other, algorithm converts them to white pixels.

2.1.5. Boundary Detection

After the processes of feature extraction and enhancement, shape of the object was interpreted as an area of many pixels. It is unnecessary to process all the pixels of the object because object contours are sufficient to process the shape alignment algorithms. Then a sharpening filter mask (Laplacian Filter Mask) was applied to the image for detecting the boundaries of the object. Therefore, a highly improvement on process time was achieved by constructing a shape with fewer elements (pixels) which it is still similar to the original.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figure 2.1. Laplacian Filter Mask

2.2. SHAPE ALIGNMENT

Shape alignment is the process of rotating and translating one shape to another to obtain best match. In many tolerance inspection applications, the object to be inspected may be at different locations in the image. The main attribute of the system is being automated, thus, position and rotation invariantly system will be capable to do the inspection process. Rotation and translation parameters will be calculated precisely in the alignment phase.

Hausdorff method is found to be a robust method for inspection however the speed of the algorithm is unsatisfactory. By considering robustness and promising features, Hausdorff Distance method was studied deeply to improve the speed of the matching process with using different auxiliary techniques. Also, a more precise alignment algorithm was proposed for inspection of faulty objects.

2.2.1. Hausdorff distance

Firstly, classical Hausdorff Distance (HD) algorithm code was developed and processed to have benchmarking information.

The classical Hausdorff distance between two finite sets of points, A and B, is defined as:

$$\vec{h}(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \quad (2.1)$$

Where $\vec{h}(A, B)$ is the directed Hausdorff distance from A to B.

The main steps of the Hausdorff matching method are as follows:

- Step1: Find the edges of reference object
- Step2: Find the edges of target object
- Step3: Rotate and translate the target object, for each relative pose compute the HD
- Step4: Select the rotation degree and translation value that yields minimum value
- Step5: Transform the target object

2.2.2. Improved Hausdorff distance

In this study, classical Hausdorff distance algorithm for alignment process was improved in two manners: Precision and time consumption.

For the improvement on precision, Hausdorff distances were calculated and the mean Hausdorff distances were determined for each trial of rotation angle variations. Using classical Hausdorff distance for alignment might cause unstable results in which object to be inspected was a faulty object, a modified approach “mean Hausdorff distance” was used.

The mean Hausdorff distance between two finite sets of points, A and B, is defined as:

$$h_m(A, B) = \max \frac{1}{N} \sum_{a \in A} \min_{b \in B} \|a - b\| \quad (2.2)$$

Where N is the number of points in A.

As we were using superimposing the centroids for translation process, we considered that faulty regions on target image might affect centroids and could cause translation errors. Thus, an iterative algorithm that calculates mean distance values was developed for best match.

Using classical Hausdorff method for alignment process iteratively for all angle values is found to be a time consuming method, thus new techniques were applied for speeding up the software computation time. When we reviewed computational costs of the inspection software, we found that the calculation of Hausdorff distance was the major function that consumed process time. Thus, methods for speeding-up the computation of Hausdorff distance between two shape models were studied.

Translation and rotation are the two transformation process that must be applied for the alignment of two objects. For translation, we used superimposing the centroids for the initial positioning which was very fast. Rotation angle was computed as the minimum of the mean Hausdorff distance value was found for all angle trials.

When we considered that the shape of the object was rotated by one degree in each iteration process, there were 360 iterations. Thus, an algorithm was developed for decreasing the number of angle trials to make less computation. First, rotation angles were tried ten by ten and minimum error was obtained. Then iterations were processed on the angle value (± 10) that included the minimum error. Thus, the trial number of overall process was decreased to 56 instead of 360.

The other technique we implemented was to construct the shape with fewer elements (pixels) as it was still similar to the original. The robustness attribute of Hausdorff distance to sampled images was used in this process. Considering the system we have built (1 pixel corresponds to 13 micron), the shape of the object was represented by many points which

were unnecessary for the alignment process. Thus we used an optimized interval of sampled data to speed up the process time.

2.3. INSPECTION

2.3.1. Dimensional Measurements

Dimensional measurement and inspection of product parameters such as length, distance and diameter is one of the most common processes in quality control applications.

The process of determination if the product under inspection is manufactured correctly by dimensional manner is also called gauging. Depending on whether the gauged parameters fall inside or outside of the user-defined tolerance limits, the component or part is either classified or rejected.

Inspection of length along the axis can be divided into two as x-axis length inspections and y-axis length inspections. While x-axis length inspections are used for to measure the distance between two sides of the object which are along the width of the image, y-axis length inspections are used for to measure the distance between two sides of the object which are along the height of the image. If the object has circular figures, dimensions special to circles like diameter, radius and perimeter can also be calculated. To set where to inspect, predefined inspection regions (ROIs) which is defined in the training mode are used for inspection.

2.3.2. Shape Tolerance Inspection

Nowadays, shape tolerance inspection of parts is a necessity for several manufacturing industries. It's essential to inspect free form objects that the unique method is to inspect from their shape descriptors. One another usage area of shape tolerance inspection is to check if a manufactured part has the faulty effects like chips and burrs. To verify the acceptance of a manufactured surface, one needs to compare the measured data with the design model to determine if the manufactured surface falls in the designed tolerance zone.

When the best alignment occurs, tolerance values of shape features were obtained automatically as the distance values from reference object points to target object points were preserved in distance error arrays. Thus, the proposed system did not need a different inspection algorithm for inspection processes that improvement on the overall computation time had gained.

3. HARDWARE OF THE PROPOSED SYSTEM

3.1. CAMERA

An image sensor is a device that converts an optical image into an electronic signal. It is used mostly in digital cameras, camera modules and other imaging devices. There are three main types of camera: Vidicons, charge coupled devices (CCDs) and, more recently, CMOS cameras (Complementary Metal Oxide Silicon – now the dominant technology for logic circuit implementation). Vidicons are the older (analogue) technology, which though cheap (mainly by virtue of longevity in production) are now being replaced by the newer CCD and CMOS digital technologies. The digital technologies, currently CCDs, now dominate much of the camera market because they are lightweight and cheap (with other advantages) and are therefore used in the domestic video market.

Today, most digital still cameras use either a CCD image sensor or a CMOS sensor. Both types of sensor accomplish the same task of capturing light and converting it into electrical signals.

A CCD image sensor is an analog device. When light strikes the chip it is held as a small electrical charge in each photo sensor. The charges are converted to voltage one pixel at a time as they are read from the chip. Additional circuitry in the camera converts the voltage into digital information.

A CMOS imaging chip is a type of active pixel sensor made using the CMOS semiconductor process. Extra circuitry next to each photo sensor converts the light energy to a voltage. Additional circuitry on the chip may be included to convert the voltage to digital data. Neither technology has a clear advantage in image quality. On one hand, CCD sensors are more susceptible to vertical smear from bright light sources when the sensor is overloaded; high-end frame transfer CCDs in turn do not suffer from this problem. On the other hand, CMOS sensors are susceptible to undesired effects that come as a result of rolling shutter.

CMOS can potentially be implemented with fewer components, use less power, and/or provide faster readout than CCDs. CCD is a more mature technology and is in most respects the equal of CMOS. CMOS sensors are less expensive to manufacture than CCD sensors.

Another hybrid CCD/CMOS architecture, sold under the name "sCMOS", consists of CMOS readout integrated circuits (ROICs) that are bump bonded to a CCD imaging substrate – a technology that was developed for infrared staring arrays and now adapted to silicon-based detector technology. Another approach is to utilize the very fine dimensions available in modern CMOS technology to implement a CCD like structure entirely in CMOS technology.

This can be achieved by separating individual poly-silicon gates by a very small gap. These hybrid sensors are still in the research phase, and can potentially harness the benefits of both the CCDs and the CMOS imagers.

Choosing the camera and its lens is related with each other seriously. The cameras can be chosen as monochrome (RS-170), composite (Y/C), RGB or Line Scan according to the application. In addition to this, for taking the required data, sensor resolution should be high adequately. Triggering and integration control specifications may also be a requirement. The last important point is to protect the camera against the soil, dust and heat that they should be produced very qualified and strong. To consider those specifications, using the industrial cameras would be the best choice.

In machine vision systems, Line Scan and array cameras are typically used. Conventionally, an array camera, takes the picture that has a shape of a square or a rectangle in one time. However a Line Scan camera has linearly sequenced detectors that scan the image as a line. Assuming all those criteria, the IDS UI-1245LE camera has been chosen as the image sensor of the system.



Figure 3.1. UI-1245LE camera

The UI-1245LE is an extremely compact camera with modern e2v CMOS sensor in 1.3 Megapixel resolution (1280x1024 pixels). Through the use of the widespread USB 2.0 technology the camera can be interfaced with a vast variety of systems without problems. The light-weight housing of the UI-1245LE features a C/CS lens mount with adjustable flange back distance.

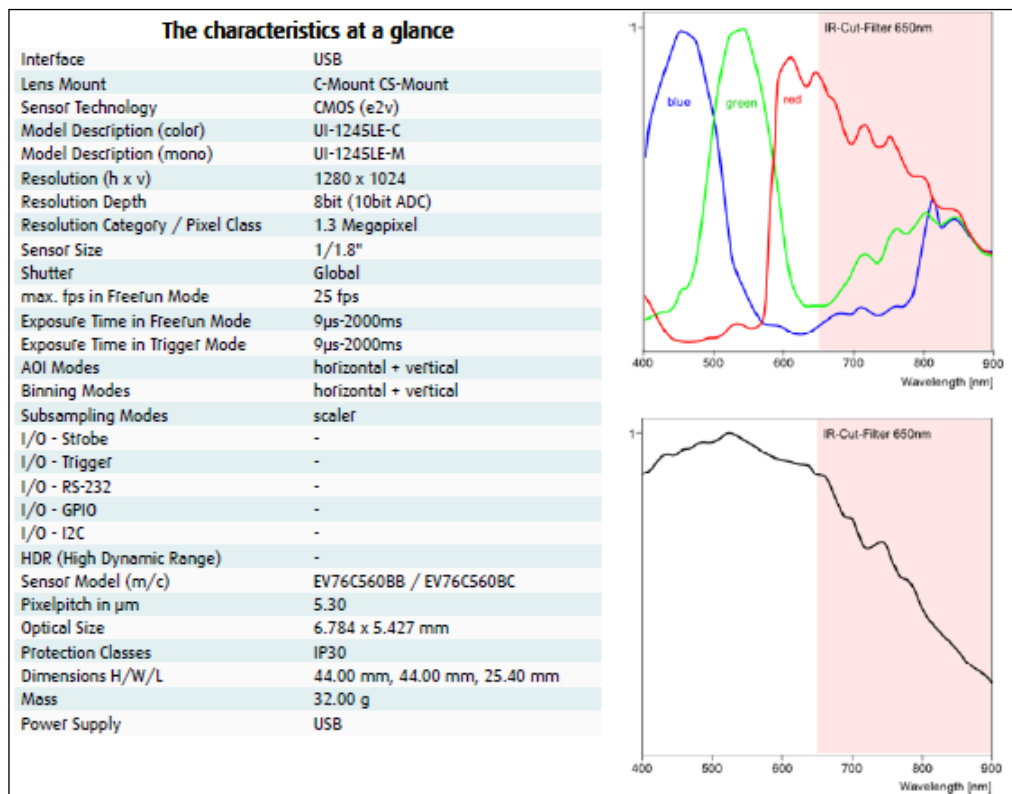


Figure 3.2. Specifications of UI-1245LE camera

3.2. LENSE

Industrial machine vision cameras come without lenses that also a machine vision lens should be chosen for the system setup. To maintain the high performance machine vision camera, it needs to be matched with appropriate lens.

There are three important factors that contribute in the selection process:

1. Field Of View (FOV)
2. Working distance
3. Sensor size of the camera

Magnification of the image acquired is (Sensor Size of the camera)/(Field of View). To estimate the required focal length for the application: Focal Length is (Magnification)*(Working Distance)/(1+Magnification)

The lens that is used in the system is a 16mm fixed focal length lens from Azure Optics which is suitable for the use with 1.3 mega-pixel color and monochrome cameras.



Figure 3.3. 16mm Lens

3.3. ILLUMINATION

Illumination is the one of the most important aspects in a machine vision system. If illumination method is not well-suited for the specific purpose, then undesirable results may occur. Performance of a vision inspection system is affected by illumination method directly. The illumination components are just as important as image quality and electronics to specifying the best system for the application.

Subject vision illumination sources and spectral content should take into consideration of two major issues which are the environmental structure for immediate inspection necessity and interaction of sample with light with regards to ambient contribution. Thus for choosing a more efficient decisive actions have to be improved in the manner of following paradigms and much of them could be answered before and through the process;

1. Surface condition: composed of flat, slights with bump, matte, shiny aspects
2. Object condition: being flat or curved
3. Color: range of the marks and details
4. Mobility: moving or stationary positions of parts
5. Intensity: combining of diffuse backlights, LEDs (towards other emitters like fiber optic lights, fluorescent, quartz halogen, metal halide, Xenon and high pressure sodium), telecentric illuminators, mounting accessories, large area and linear auxiliary lights, edge to edge forms as light guide adapters might be needed to mate with fiber optic illuminators in case of design factors.

Backlighting as a prerequisite is a machine vision lighting method that provides a high contrast silhouette of a part or parts' segments. For instance the object appears black against a uniform white background. Backlights can also be used with color filters, placed at the illuminator input end, to improve the contrast of colored components.

Backlight technique is a main requirement for the object is lit from behind. Utilization helps to maintain the silhouette of opaque objects or for imaging through transparent objects. High contrast for edge detection is a very advantageous facility. The negative

reaction that comes from elimination of surface detail is derived and minimized by enough powerful LED lighting used for homogenous lighting distribution.

In this study, a backlighting system which is composed of white power led lighting were designed and created. Illumination of the system provided high contrast for detection of edges.



Figure 3.4. Backlighting

3.4. PROCESSOR

In this study, a PC based vision system was developed. The machine vision camera that was used for acquiring the images had USB interface, thus we did not need a frame grabber board to retrieve the images. The experiments were processed under Visual Studio 2008 environment and C++ language, on a PC with Intel Core i7 2GHz clock speed and 6GB RAM.

4. IMPLEMENTATION ISSUES

4.1. SYSTEM SETUP

The imaging system set up must be designed before processing by giving importance to five vision concepts which are field of view, working distance, resolution, depth of field, and sensor size. The smallest feature size of the object that can be distinguished by the imaging system is resolution. Tolerance of the inspection system is defined by the resolution. Field of view, the viewable area of the object under inspection is another concept that the system must be configured as the whole part of the object must be acquired by the camera. The field of view of our system was (133 mm height)*(166 mm width).



Figure 3.5. System Setup

The distance from the front of the camera lens to the object under inspection is working distance. The size of a camera sensor's active area parameter is important in determining the proper lens magnification required to obtain a desired field of view. For the tolerance inspection systems, camera must be positioned perpendicular to the object to reduce perspective errors. In our setup, according to our calibration measurements 1 pixel corresponds 0.13 mm that means 13 microns precise measurements can be performed.

4.2. TRAINING THE OBJECT

Training is the important part of the inspection process if the operation of the system is based on the comparison of an arbitrarily positioned object to a reference object. We developed a training program that image information was gathered and the inspection regions were determined.

Reference image must be loaded prior to the inspection program was started, thus a reference object image had to be previously acquired in the training mode. After triggering the program with user interface button, camera captures the frame and saved to the predefined computer directory. Color image was loaded and converted to grayscale image with the OpenCV functions. Then we applied binary thresholding to extract the object from the image. There could be some noises on the image because of the dust on the environment so image enhancement algorithms were processed to clear these pixels which are not related with the object. Inspection regions could be acquired by mouse clicking.

4.3. PROGRAM OPERATION FLOW

The operation of the system is based on the comparison of an arbitrarily positioned object to a reference object generated from the standard dimensions set. For this purpose, alignment parameters must be calculated first, and then the object must be translated and rotated to normalize the position.

When the program has started, it automatically loads the predefined values of the target object. The software is triggered with pressing the defined button. Then it captures the frame and loads it to the software as `IplImage` structure to be processed. `IplImage` is one of

the main structures of Intel Image Processing Library that holds image information like width, height, channels and pixel values of data arrays. After shape extraction processes, the data of image matrixes were gathered and stored in two arrays, row and column coordinates.

As we are using superimposing the centroids technique for translation, center of gravity of shapes must be found first. The center of gravity which is also called centroid is the coordinates of average distribution points of the shape extracted from the image.

Where the general function $f(x, y)$ is:

$$f(x, y) = \begin{cases} 1, & \text{if } (x, y) \in D \\ 0, & \text{otherwise} \end{cases} \quad (4.1)$$

Where D is the domain of the binary shape, its centroid (g_x, g_y) is:

$$\begin{cases} g_x = \frac{1}{N} \sum_{i=1}^N x_i \\ g_y = \frac{1}{N} \sum_{i=1}^N y_i \end{cases} \quad (4.2)$$

Where N is the number of point in the shape, $(x_i, y_i) \in \{(x_i, y_i) | f(x_i, y_i) = 1\}$

After centroids of reference object and target object was found, translation parameters were calculated by taking the difference of centroids. Rotation process of the target shape also changes the centroid of the target object, thus we calculated the translation parameters by applying the rotation operations.

Two dimensional Rotation Matrix (R) is defined by:

$$[R] = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (4.3)$$

The new coordinates of a point can be calculated by using matrix operations:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (4.4)$$

Rotated coordinates (x', y') of the point (x, y) after rotation are as follows:

$$x' = x \cos \theta - y \sin \theta \quad (4.5)$$

$$y' = x \sin \theta + y \cos \theta \quad (4.6)$$

By using the equations above, we calculated translation parameters of row and column for each angle θ :

$$t_x = g_x - (x * \cos \theta - y * \sin \theta) \quad (4.7)$$

$$t_y = g_y - (x * \sin \theta + y * \cos \theta) \quad (4.8)$$

Where t_x is translation parameter of row coordinate, t_y is translation parameter of column coordinate, g_x is reference shape centroid row value, g_y is reference shape centroid column value, x is target shape centroid row value and y is target shape centroid row value.

As all elements (points) of the target shape was stored in row array and column array, the new coordinates of the target object after transformation was calculated by:

$$x' = x * \cos \theta - y * \sin \theta + t_x \quad (4.9)$$

$$y' = x * \sin \theta + y * \cos \theta + t_y \quad (4.10)$$

Where x' is target shape row values after transformation, y' is target shape column values after transformation, x is target shape row values before transformation and y is target shape column values before transformation.

The rotation angle was found by iterative trials of different angle values. In each trial, target shape was transformed to its new position and the error of the matching process was calculated. The error we had defined was mean Hausdorff distance, thus we searched for the minimum of mean Hausdorff distances.

To find the minimum of mean Hausdorff distances, first, distances of two point sets were calculated with using the formula below.

$$\text{distanceH} = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)} \quad (4.11)$$

Where *distanceH* is the Hausdorff distance between two set of points, x_1 is reference shape row values, x_2 is target shape row values after transformation, y_1 is reference shape row values, y_2 is target shape row values after transformation.

Then, for each trial of rotation, mean Hausdorff distances were calculated and stored in an array. The minimum of the mean Hausdorff distance value and the angle that matched with the trial were found. The angle was determined as correct angle of rotation.

After determination of the correct angle of rotation, transformation phase was started. The same shapes that were placed on the image with different angles could be interpreted with different number of pixels. Target object boundaries could be affected with the missing pixels problem thus inspection of the missing part might become impossible. To avoid the missing pixels problem, the target object was rotated in binary mode and then edge of the object was found by boundary detection.

After rotation process, target shape was positioned to the reference shape by superimposing the centroids. If the target object was not a faulty object, the best match would be achieved. However, if the target object was a faulty object, superimposing the centroids might not give the best match of two objects. The reason of that incident was the faulty parts of the target object might affect the centroid position, thus, a fine tuning algorithm was applied which calculated the error (mean Hausdorff distance) in each relative pose iteratively. This algorithm shifted the image in +x, -x, +y, -y coordinates

system iteratively and calculated error in every shifting. The iteration which gave the lowest error value was determined as the best match.

After successful alignment of the reference shape and the target shape, inspection algorithms were processed. Length and distance measurements of the shape were calculated by taking distances between the shape boundaries that placed on predefined regions. Dimensions special to circles like diameter, radius and perimeter calculated by founding the center of the circle first. After center of the circle was found, radius was found by taking the distances of pixel coordinates of center and the nearest circle boundary. By using the formulas above, diameter and perimeter were also calculated.

$$\text{Perimeter} = 2 * \pi * r \quad (4.12)$$

$$\text{Diameter} = 2 * r \quad (4.13)$$

The results that were calculated as pixel values, then converted to millimeter with predefined calibration parameters. According to the calibration parameters which was used in our system 1 pixel corresponds to 0.13 mm, thus real world measurements were found by multiplying calculated pixel value with 0.13.

5. EXPERIMENTS

The object to be inspected that was chosen for the experimental studies was a mechanical part which was manufactured in turn benches. It is used as connection apparatus in machining industry. Its raw material is aluminum that makes it easy to be formed however aluminum material causes a shiny surface that makes it hard to inspect with machine vision applications.

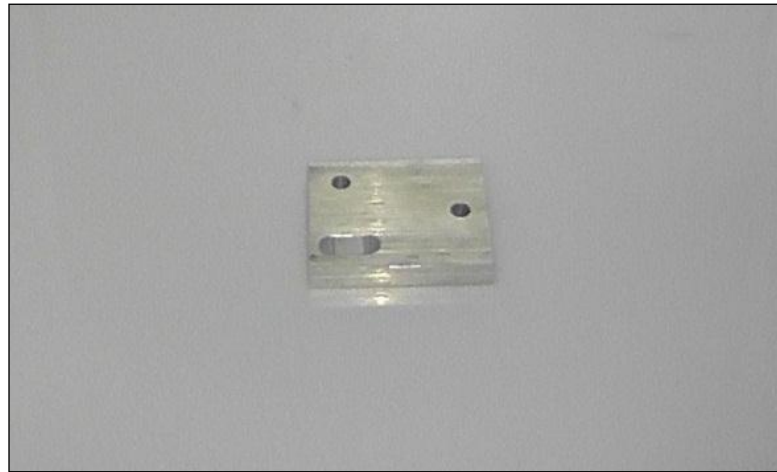


Figure 5.1. Object to be inspected

As mentioned in the third chapter before, backlighting illumination was used for neglecting shiny effects of the part and acquiring the best shape image for inspection. The part was chosen for having holes on it thus not only the outer boundaries but also inner boundaries would be important for alignment process.

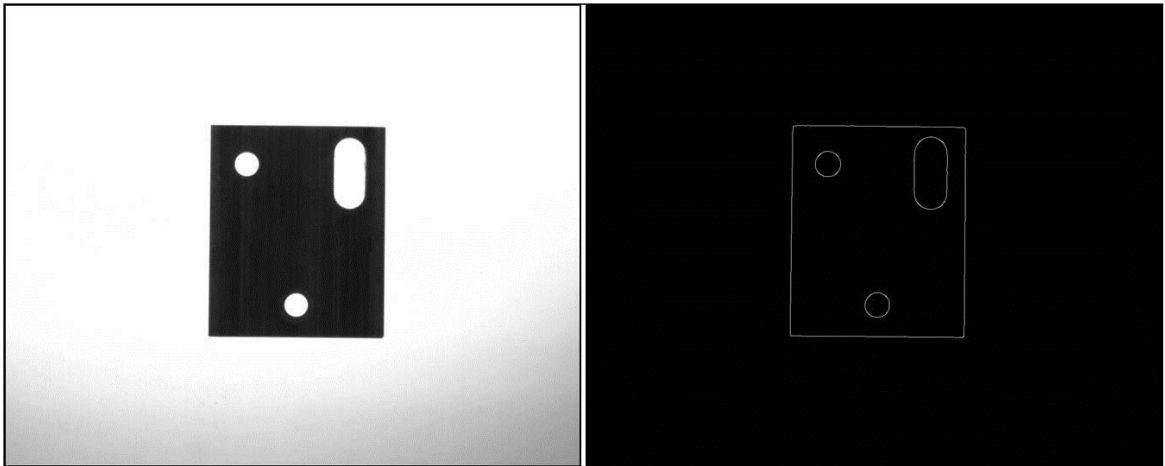


Figure 5.2. Reference Image and its edges

The object to be inspected was measured with a caliper to extract its dimensional data. These values were used for the benchmarking of the visual inspection results. Measurement data that is measured by caliper were also used for verification of the system calibration parameters. Six different region of the object were chosen for inspection and measurement process. These inspection regions are height (M1), width (M2), two circles (M5-M6), width measurement of the ellipsoid (M3) and height measurement the ellipsoid (M4). The inspection regions and their measurements are also shown for test image1, in Figure 5.3.

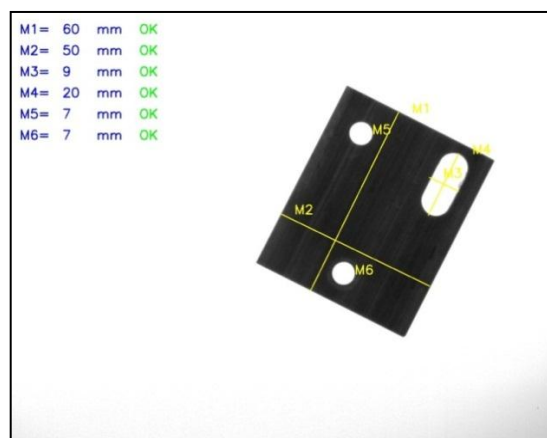


Figure 5.3. Inspection regions and measurements on test image1

5.1. Tests on objects with no fault

Before the inspection phase, target shape must be well aligned to reference shape for performing precise measurements. Hausdorff distances between reference shape and target shape pixels (two set of points) were calculated. Minimum distances were found for all rotation angle trials and then the average values of minimum distances calculated. The maximum value of the average values was chosen as best match. For the best match occasion, Minimum Hausdorff distances is shown in Figure 5.4.

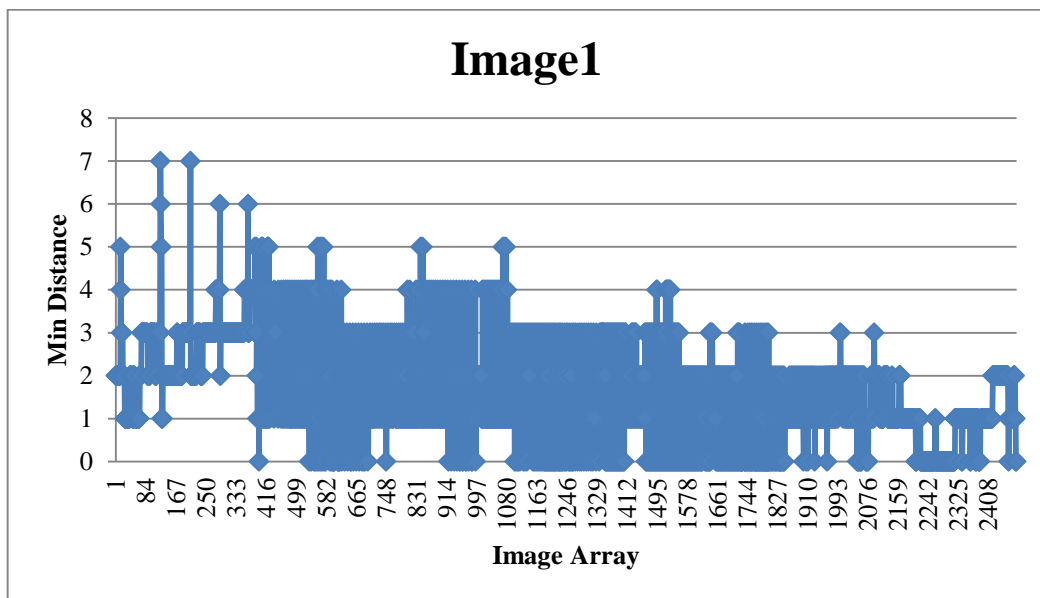


Figure 5.4. Hausdorff distance between two shape points

Ten different image of the object with different orientations and positions were processed under alignment algorithms. The results of translation parameters, rotation angle, maximum distance and mean distance are shown in the Table 5.1.

Table 5.1. Results of alignment experiments for object1

Images	Translation Row	Translation Column	Rotation Angle	Max Distance	Mean Distance
Object1-image1	123	-162	26	7	1,56
Object1-image2	128	-12	100	4	0,97
Object1-image3	41	43	199	3	0,81
Object1-image4	76	-68	69	3	0,99
Object1-image5	20	14	161	4	0,94
Object1-image6	45	-109	49	5	1,12
Object1-image7	-63	-172	24	5	1,42
Object1-image8	45	34	135	4	1,00
Object1-image9	154	38	114	4	1,16
Object1-image10	133	-56	343	6	1,36

One of the main disadvantages of classical Hausdorff distance algorithm is that it looks for the information for just one point (which is maximum of minimums) while mean distance gathers information from all minimum points. As the maximum distance can be easily affected from shape dissimilarities, mean distance has more stable attitude. The comparison of maximum distances and mean distances are shown with a graphical representation that is given in Figure5.5.

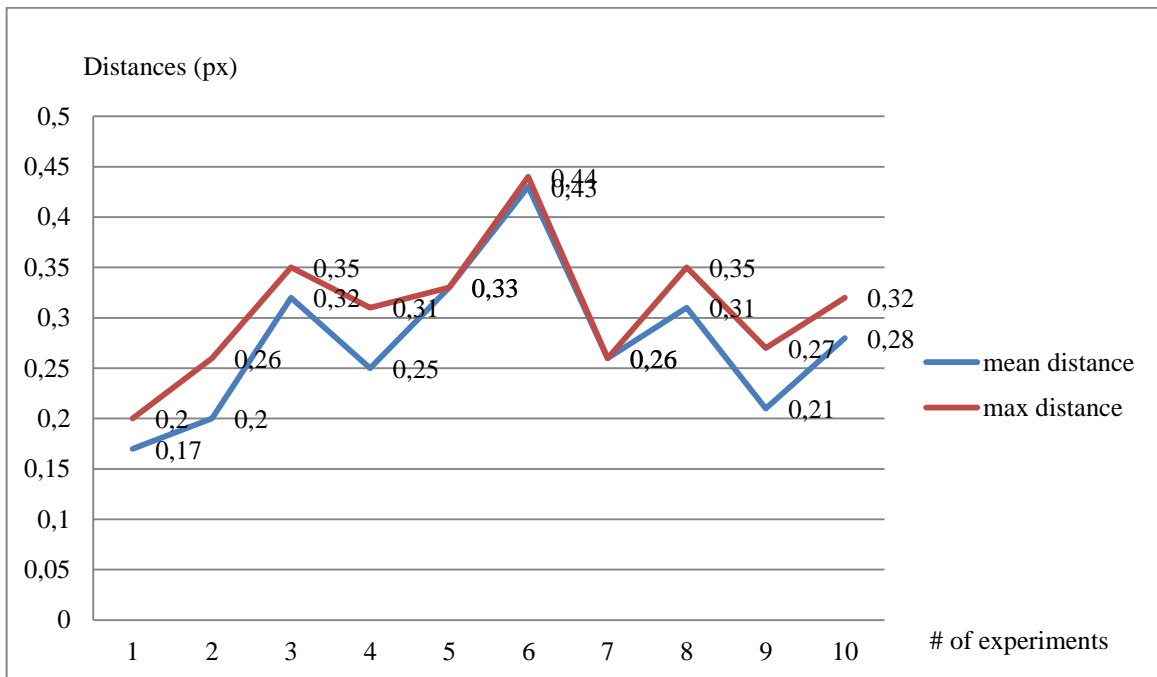


Figure 5.5. Maximum distances and mean distances of good objects (object1)

5.2. Tests on faulty objects

The other experiments were done with using the faulty object images. The results showed that if the target object was a faulty object, superimposing the centroids might not give the best match. The reason of that incident was the faulty parts of the target object might affect the centroid position, thus, a fine tuning algorithm was applied which calculated the error (mean Hausdorff distance) in each relative pose iteratively. The proposed algorithm shifted the image in +x, -x, +y, -y coordinates system iteratively and calculated error in every shifting. The iteration which gave the lowest error value was determined as the best match. Shifting 1 pixel in +x axis corresponds to adding 1 to all pixel values in column array, shifting 1 pixel in -x axis corresponds to subtracting 1 to all pixel values in column array, shifting 1 pixel in +y axis corresponds to subtracting 1 to all pixel values in column array and shifting 1 pixel in -y axis corresponds to adding 1 to all pixel values in column array. First, the algorithm searched for the mean distance error lower than the original, if it was found, the shape was shifted unless the error was minimized. The algorithms were applied five different faulty object images and the mean distance errors with iterations for five

faulty objects are shown in table 5.2, 5.3, 5.4, 5.5, 5.6. The iterations for rows described as r and iterations for columns described as c.

Table 5.2. Iterations and results for image11

# of Iterations for Image11	Max Distance Error	Mean Distance Error	Iterations
1	12	1,91	original
2	12	1,93	r +1
3	12	2,21	r -1
4	13	1,54	c +1
5	11	2,31	c -1
6	14	1,48	c +2
7	15	1,78	c +3

Table 5.3. Iterations and results for image12

# of Iterations for Image12	Max Distance Error	Mean Distance Error	Iterations
1	12	2,23	original
2	12	2,27	r +1
3	12	2,39	r -1
4	13	1,84	c +1
5	11	2,64	c -1
6	13	1,5	c +2
7	13	1,65	c +3

Table 5.4. Iterations and results for image13

# of Iterations for Image13	Max Distance Error	Mean Distance Error	Iterations
1	12	3,36	original
2	12	3,05	r +1
3	12	3,72	r -1
4	12	3,92	c +1
5	12	2,8	c -1
6	12	2,26	c -2
7	12	1,75	c -3
8	12	1,31	c -4
9	12	1,07	c -5
10	12	1,25	c -6
11	12	0,7	c -5 r +1
12	12	0,61	c -5 r +2
13	12	0,87	c -5 r +3

Table 5.5. Iterations and results for image14

# of Iterations for Image14	Max Distance Error	Mean Distance Error	Iterations
1	11	2,54	original
2	11	2,23	r +1
3	11	2,93	r -1
4	11	3,06	c +1
5	11	2,06	c -1
6	11	1,84	c -2
7	11	1,87	c -3
8	11	1,5	c -2 r +1
9	11	1,42	c -2 r +2
10	11	1,48	c -2 r +1

Table 5.6. Iterations and results for image15

# of Iterations for Image15	Max Distance Error	Mean Distance Error	Iterations
1	14	2,72	original
2	14	2,35	r +1
3	14	3,13	r -1
4	15	3,16	c +1
5	13	2,34	c -1
6	12	2,24	c -2
7	13	2,51	c -3
8	12	1,58	c -2 r +2
9	12	1,6	c -2 r +3

Maximum distance errors and mean distance errors before and after application of fine tuning algorithm for five faulty objects are shown in table 5.7.

Table 5.7. Distance errors before and after fine tuning algorithm (object1)

Images	Translation Row	Translation Column	Rotation Angle	Max Distance Error (1)	Mean Distance Error (1)	Max Distance Error (2)	Mean Distance Error (2)
Image11	-83	5	132	12	1,91	14	1,48
Image12	91	123	97	12	2,23	13	1,5
Image13	-38	16	352	12	3,36	12	0,61
Image14	-155	-137	202	11	2,54	11	1,42
Image15	-110	-184	164	14	2,72	12	1,58

The graphical representation of maximum distance errors and mean distance errors before and after application is shown in figure 5.6.

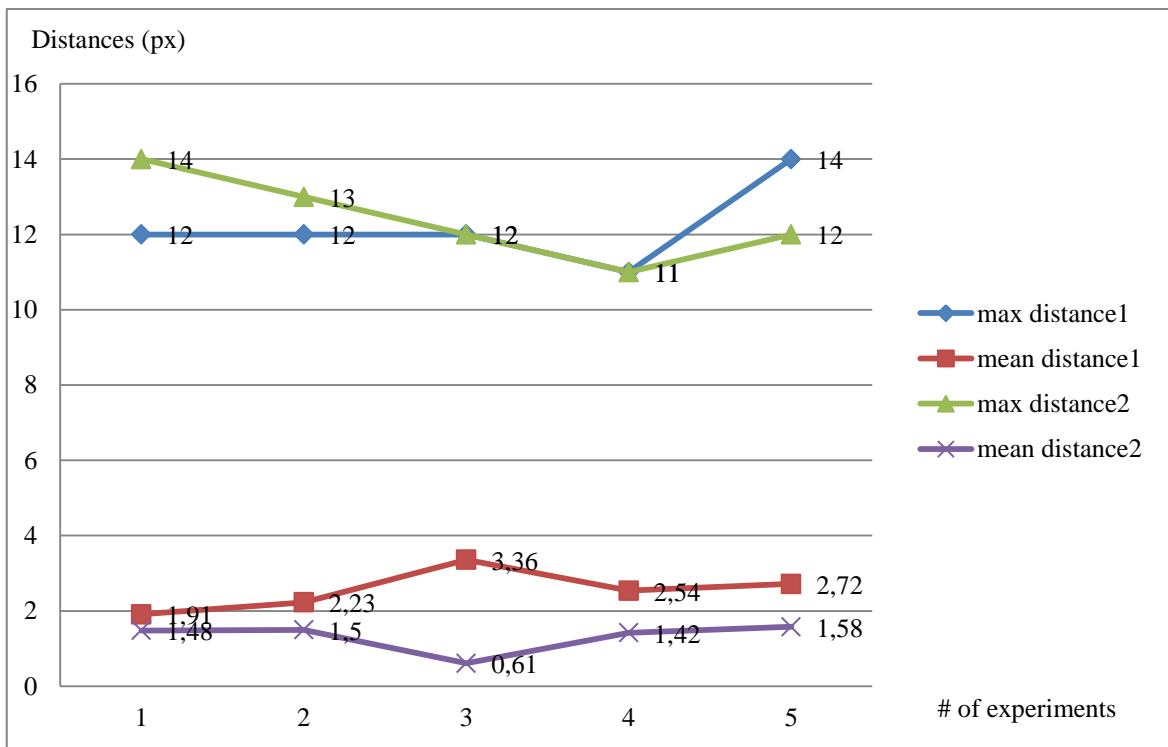


Figure 5.6. Maximum distances and mean distances of faulty objects (object1)

As it is seen in figure 5.6, after application of fine tuning algorithm the mean distance errors were decreased significantly. The figure also shows us that using maximum

distances for alignment is an unstable method as they are affected from faulty parts of objects.

Matching of reference object and target object before applying the fine tuning algorithm is shown in Figure 5.7. It can be seen that best alignment match could not be achieved and incorrect matching could cause unstable results for tolerance inspection of shape boundaries. The image after applying the fine tuning algorithm, reference object and target object is shown in Figure 5.8.

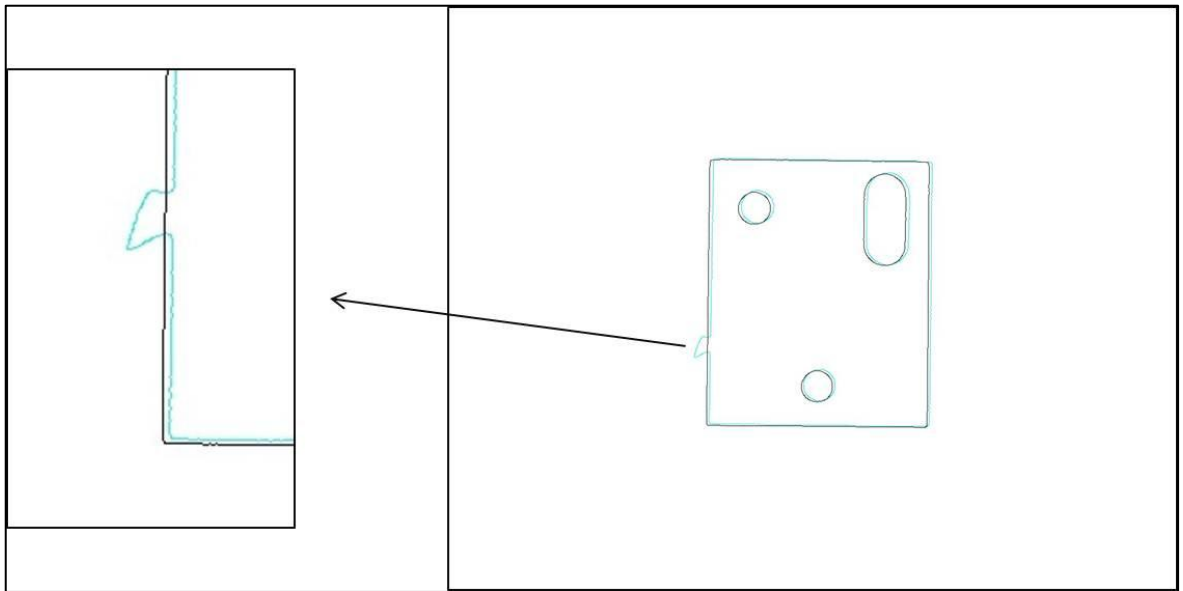


Figure 5.7. Before fine tuning alignment

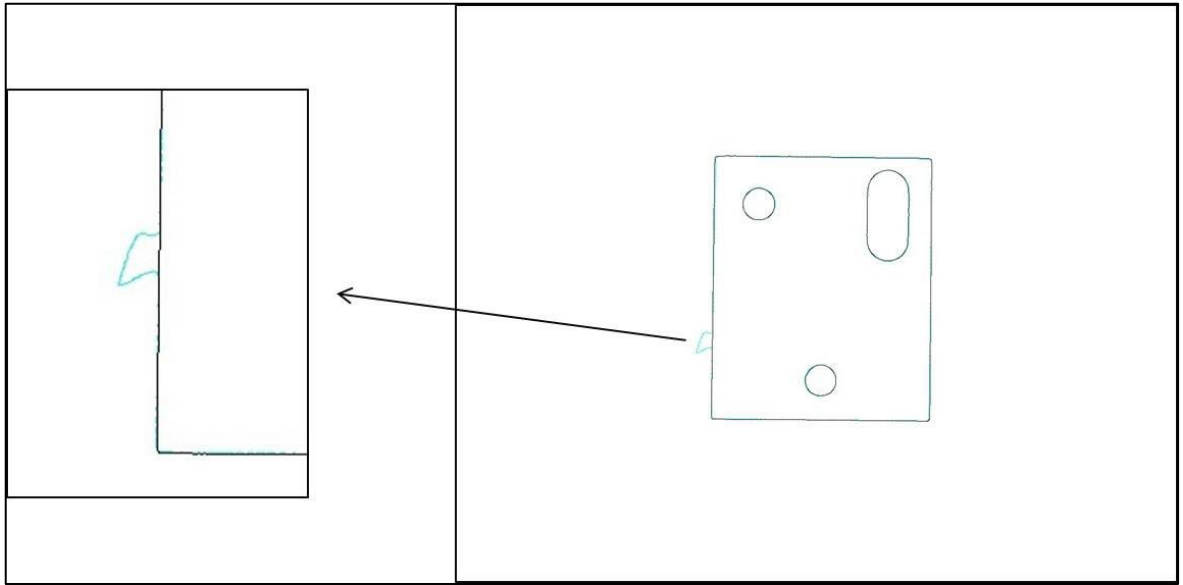


Figure 5.8. After fine tuning alignment

After successful alignment of the reference shape and the target shape, inspection algorithms were processed and results were shown. If the measurement of the inspected parameter was in the tolerance interval, the result was displayed on screen as OK. If it was not in the tolerance interval, then the result of the inspection process was displayed on screen as NOK (Not OK). An inspection result of faulty object is shown in Figure 5.9, width length measurement was more than given tolerance values (50 ± 2) thus it was displayed as NOK.

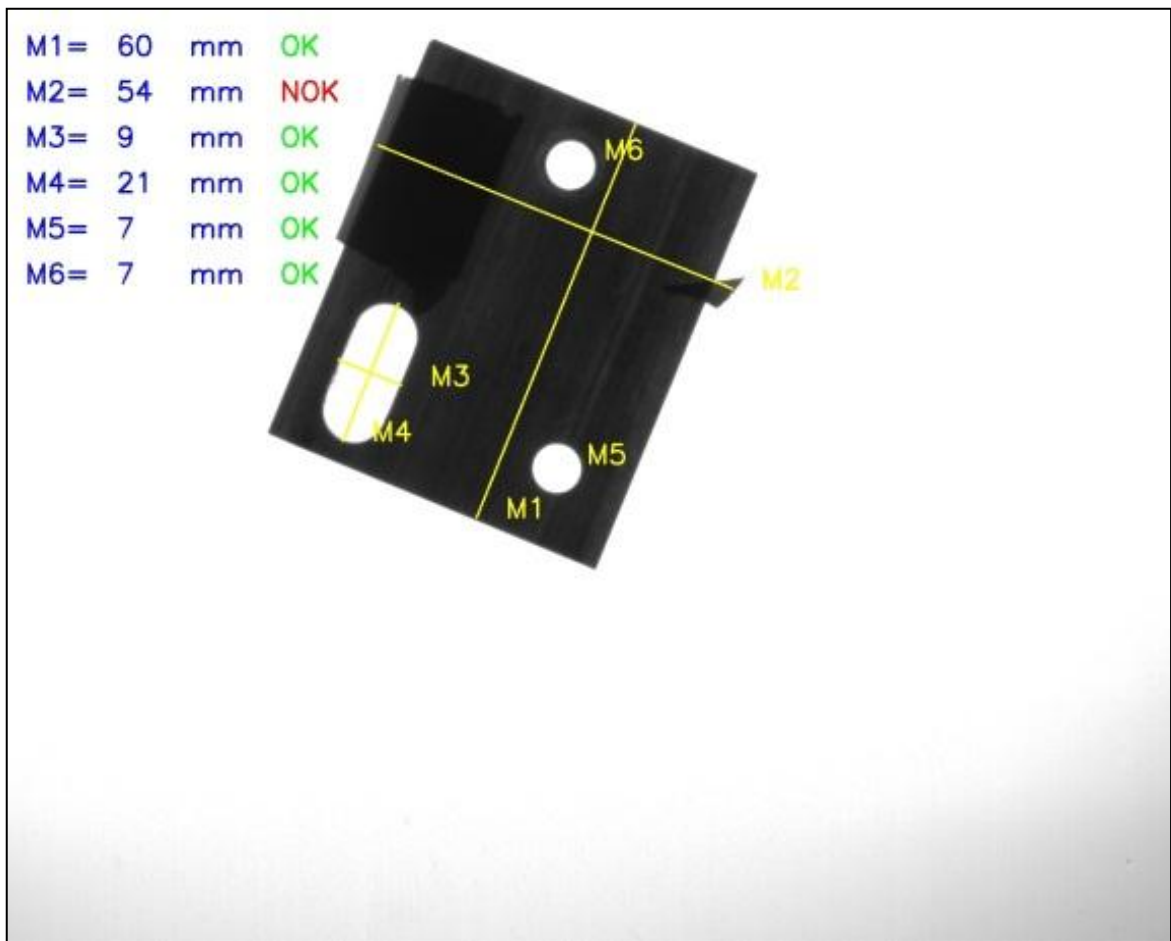


Figure 5.9. Inspection and measurements

The algorithms were also implemented on different test objects. Reference images and test results that were applied on other four different objects are shown below:

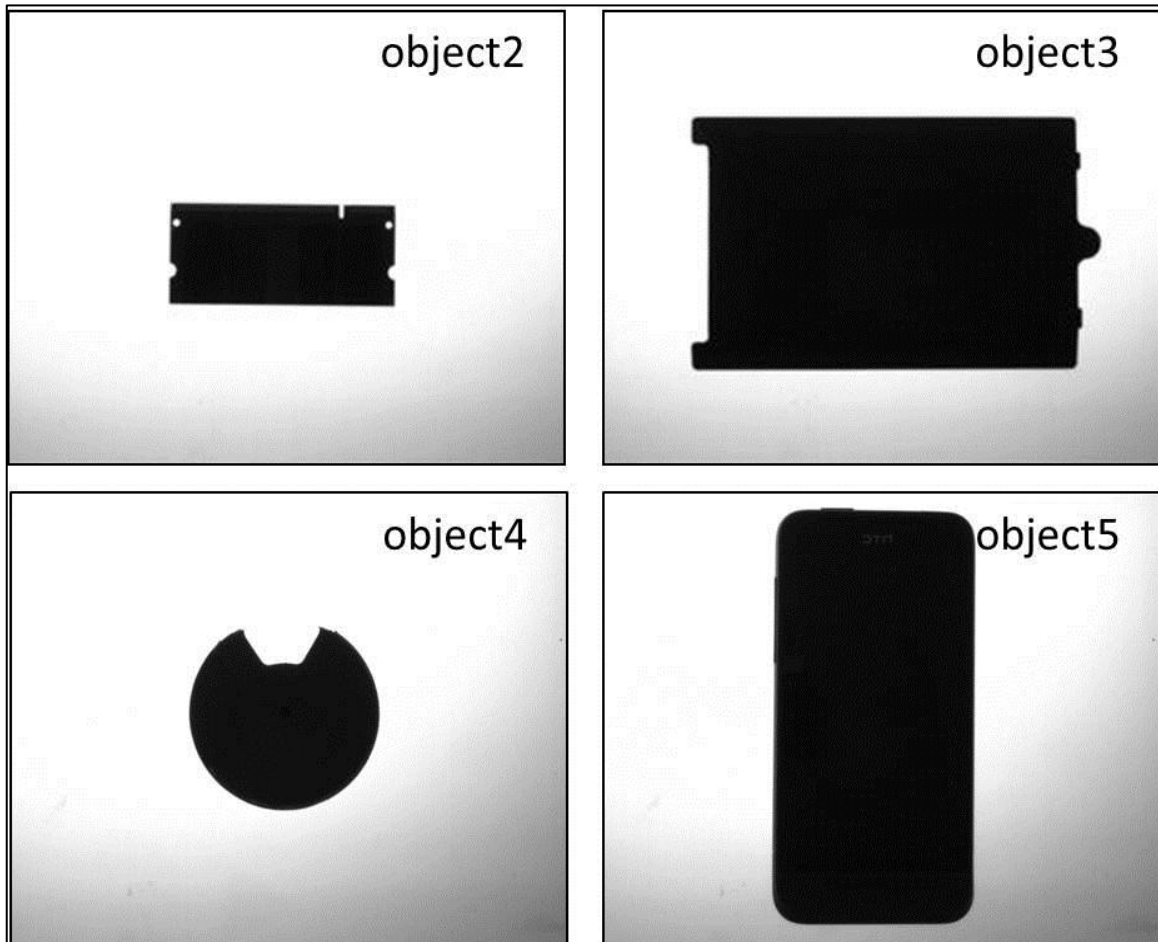


Figure 5.10. Reference images of other test objects

Table 5.8 Results of alignment experiments for object2

Images	Mean Distance Error (1)
Object2-image1	1,78
Object2-image2	0,96
Object2-image3	1,54
Object2-image4	1,56
Object2-image5	1,51
Object2-image6	0,60
Object2-image7	0,94
Object2-image8	0,88
Object2-image9	1,21
Object2-image10	1,51

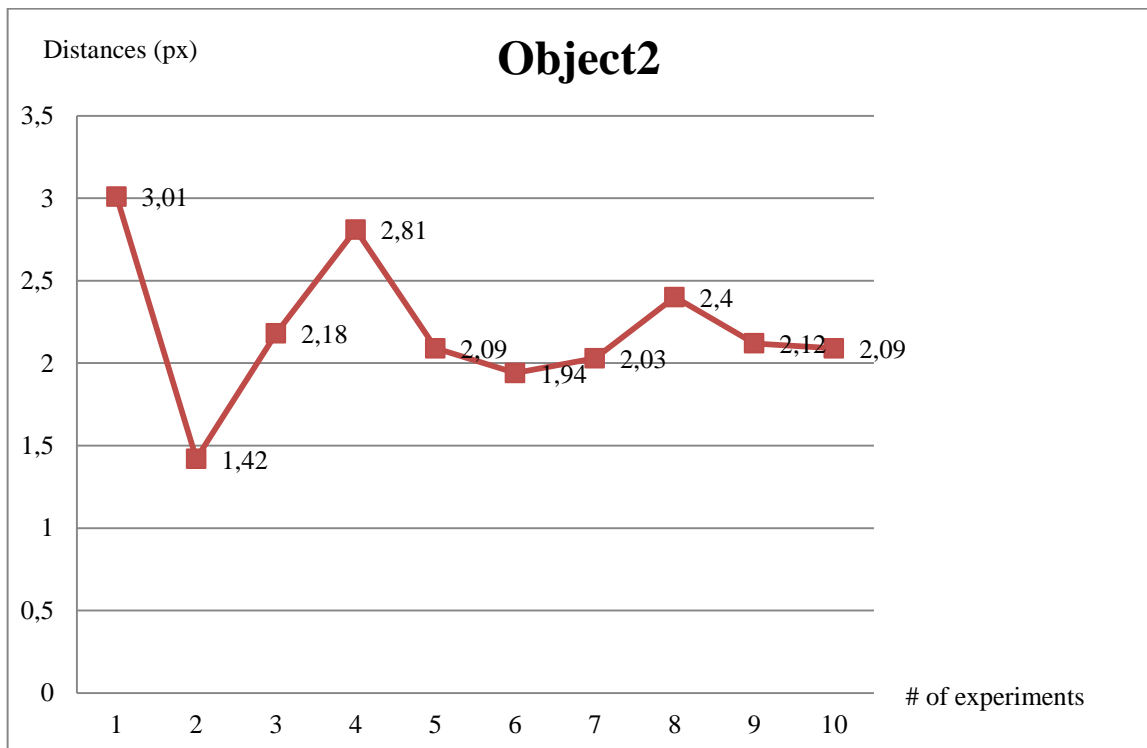


Figure 5.11. Mean distances of good objects (object2)

Table 5.9. Distance errors before and after fine tuning algorithm (object2)

Images	Mean Distance Error (1)	Mean Distance Error (2)
Object2-Image11	1,29	1,11
Object2-Image12	1,55	1,21
Object2-Image13	6,14	2,99
Object2-Image14	5,85	3,55
Object2-Image15	5,73	3,75

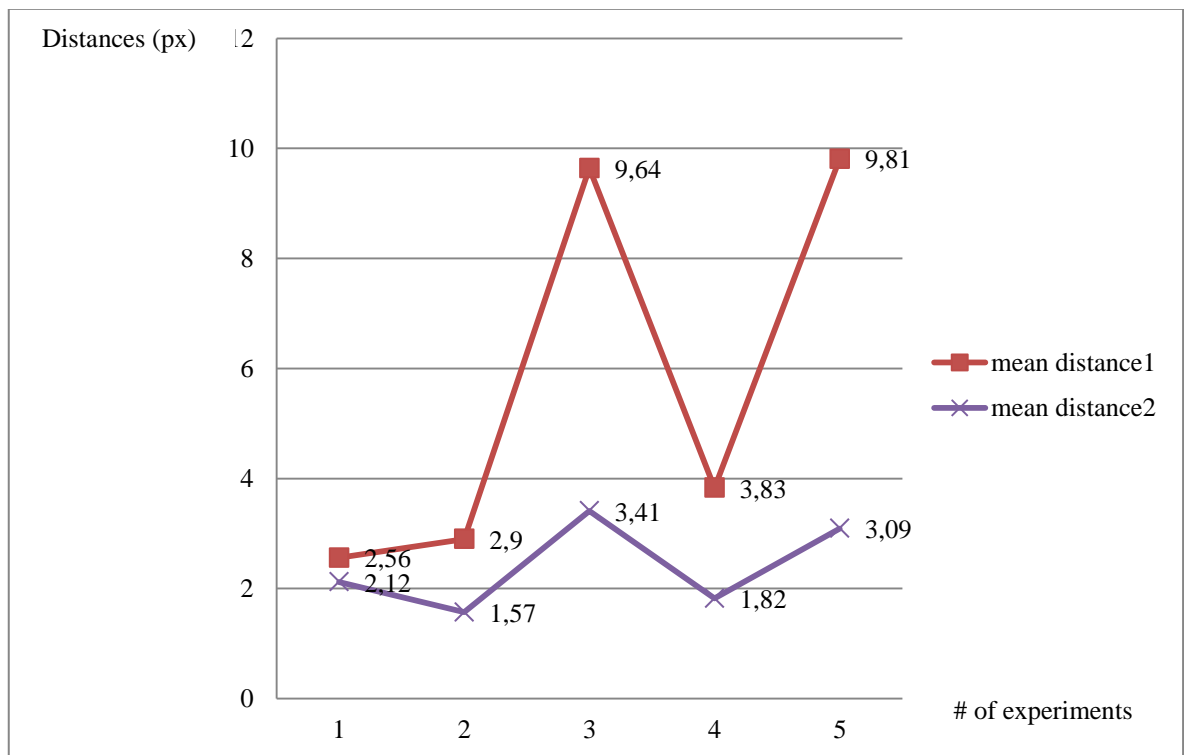


Figure 5.12. Maximum distances and mean distances of faulty objects (object2)

Table 5.10 Results of alignment experiments for object3

Images	Mean Distance Error (1)
Object3-image1	3,01
Object3-image2	1,42
Object3-image3	2,18
Object3-image4	2,81
Object3-image5	2,09
Object3-image6	1,94
Object3-image7	2,03
Object3-image8	2,40
Object3-image9	2,12
Object3-image10	2,09

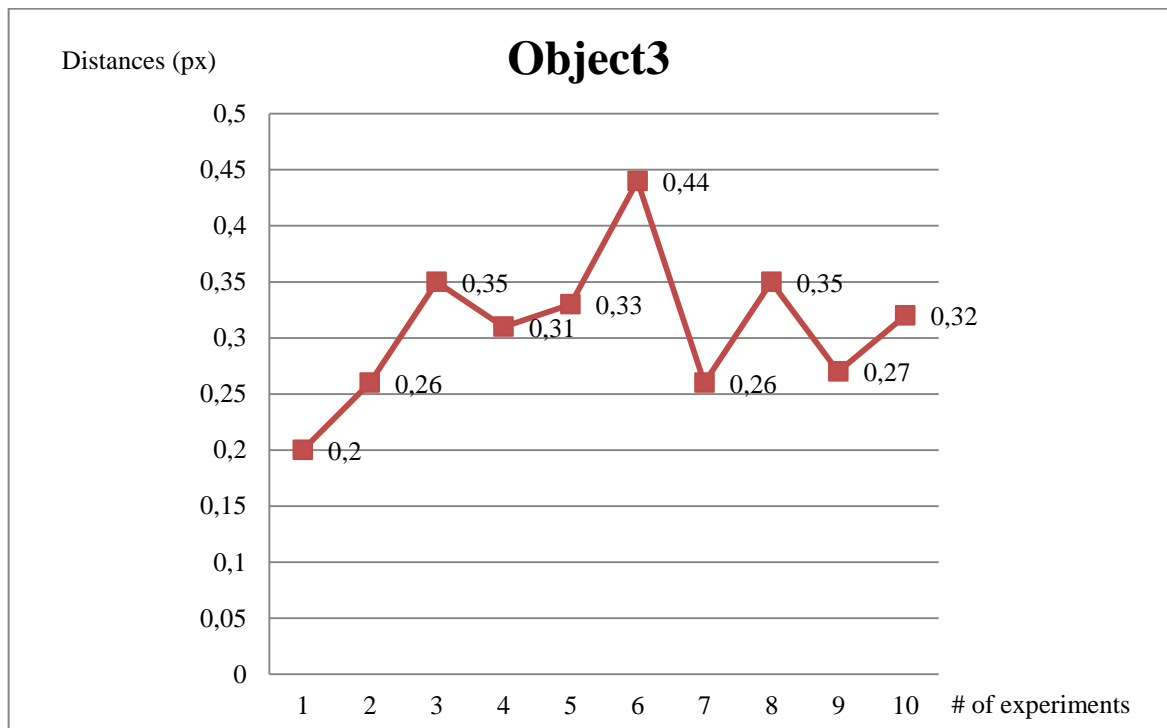


Figure 5.13. Mean distances of good objects (object3)

Table 5.11. Distance errors before and after fine tuning algorithm (object3)

Images	Mean Distance Error (1)	Mean Distance Error (2)
Object3-Image11	2,56	2,12
Object3-Image12	2,90	1,57
Object3-Image13	9,64	3,41
Object3-Image14	3,83	1,82
Object3-Image15	9,81	3,09

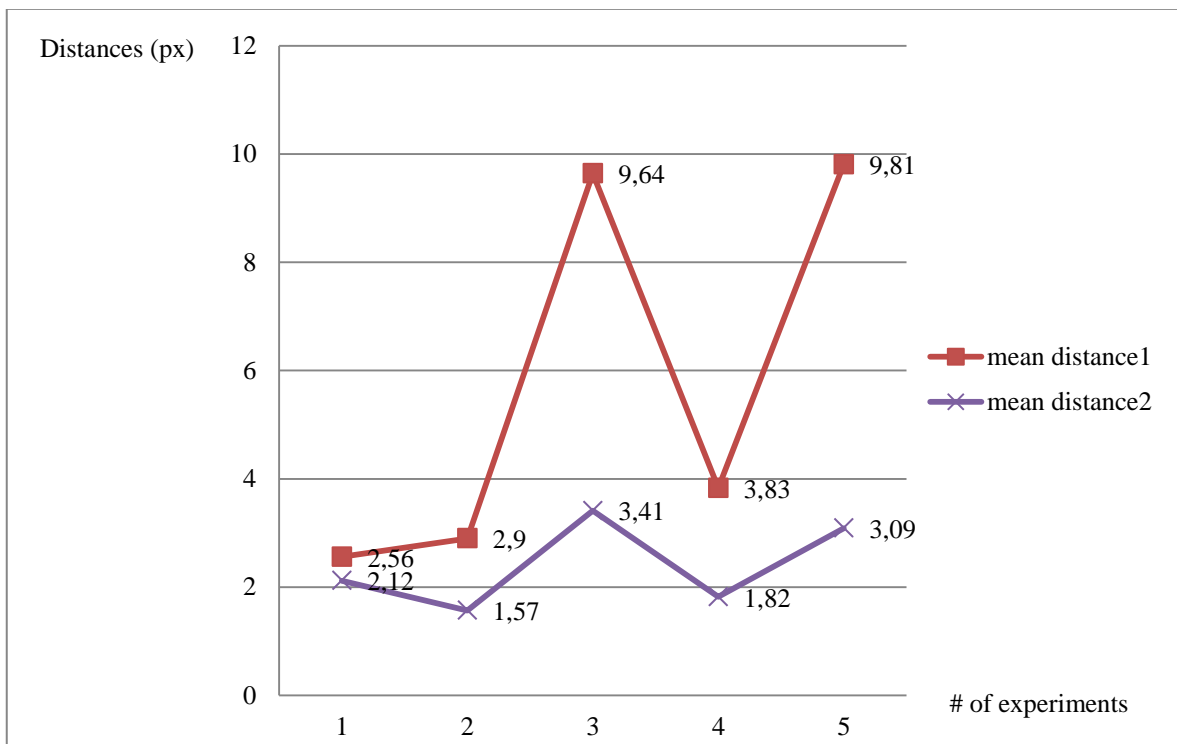


Figure 5.14. Maximum distances and mean distances of faulty objects (object3)

Table 5.12 Results of alignment experiments for object4

Images	Mean Distance Error (1)	Distance Max
Object4-image1	0,10	2,83
Object4-image2	0,25	3,16
Object4-image3	0,90	1,00
Object4-image4	0,36	5,66
Object4-image5	0,32	6,32
Object4-image6	0,16	3,00
Object4-image7	0,30	4,47
Object4-image8	0,22	5,00
Object4-image9	0,22	5,10
Object4-image10	0,28	3,00

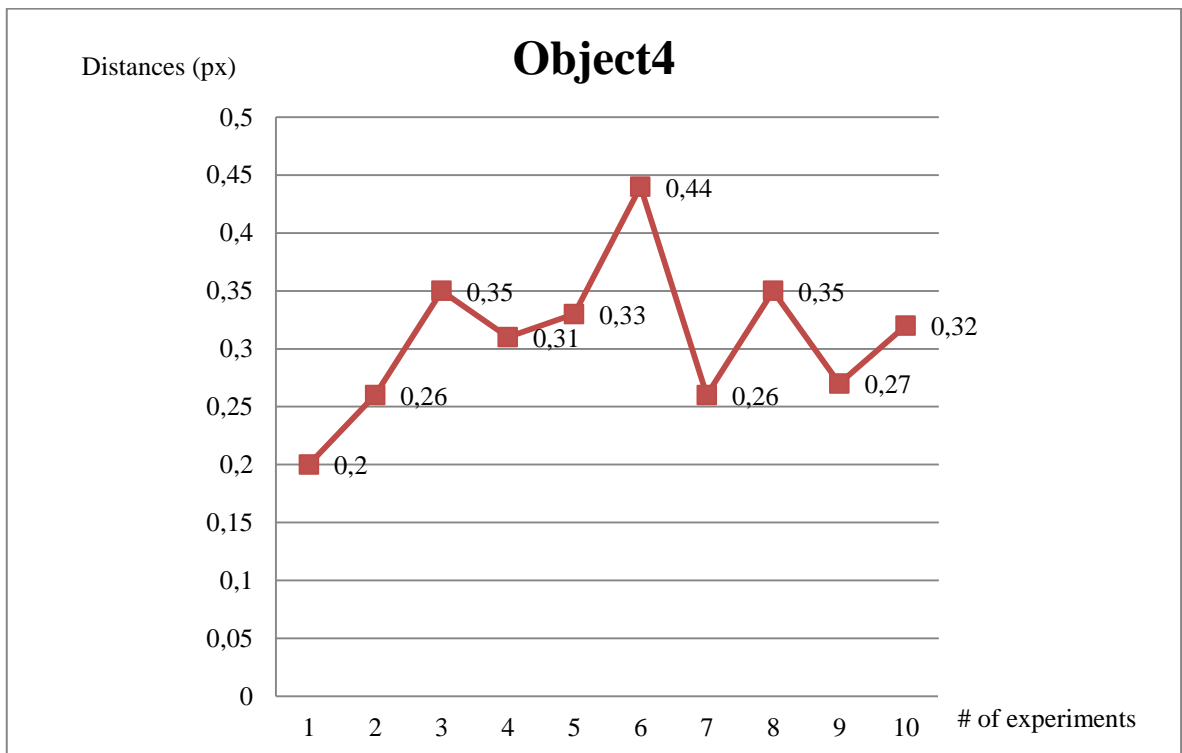


Figure 5.15. Mean distances of good objects (object4)

Table 5.13. Distance errors before and after fine tuning algorithm (object4)

Images	Mean Distance Error (1)	Mean Distance Error (2)	Distance Max
Object4-Image11	0,56	0,45	25,96
Object4-Image12	1,79	0,78	30,41
Object4-Image13	0,83	0,33	27,20
Object4-Image14	1,33	0,62	28,16
Object4-Image15	0,49	0,31	19,03

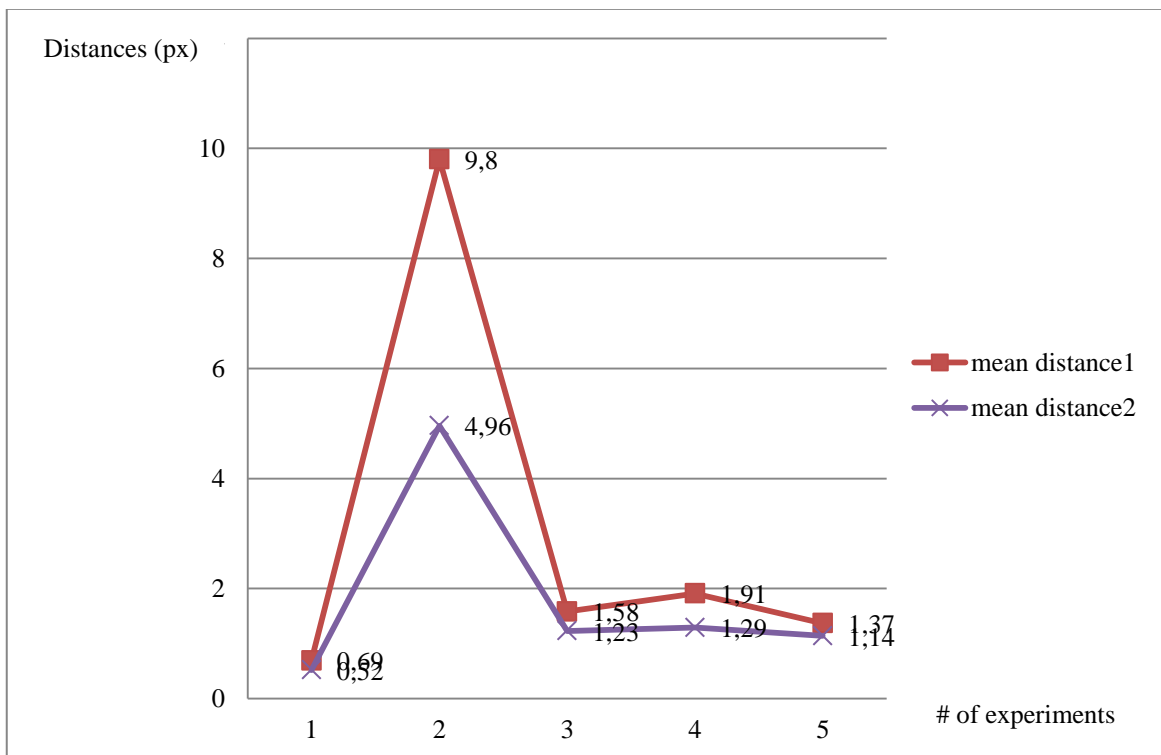


Figure 5.16. Maximum distances and mean distances of faulty objects (object4)

Table 5.14 Results of alignment experiments for object5

Images	Mean Distance Error (1)	Distance Max
Object5-image1	0,20	3
Object5-image2	0,26	5
Object5-image3	0,35	5
Object5-image4	0,31	3,61
Object5-image5	0,33	4,47
Object5-image6	0,44	6,40
Object5-image7	0,26	5,10
Object5-image8	0,35	5,00
Object5-image9	0,27	5,83
Object5-image10	0,32	4,00

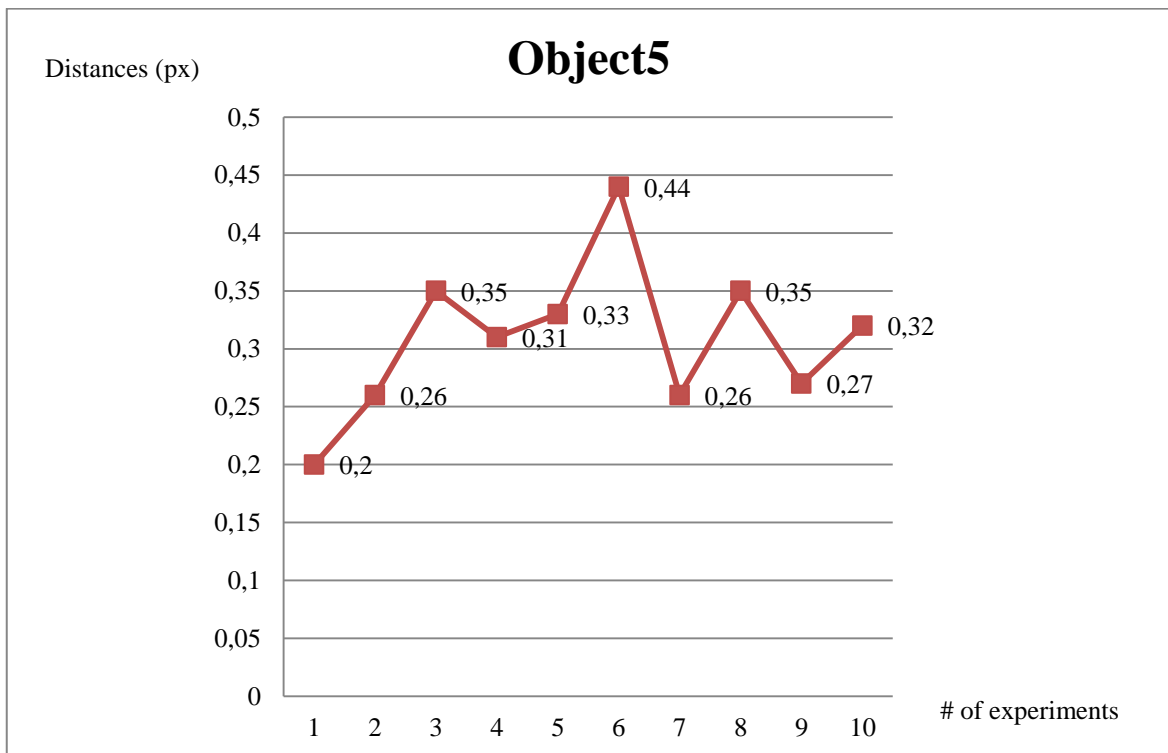


Figure 5.17. Mean distances of good objects (object5)

Table 5.15. Distance errors before and after fine tuning algorithm (object5)

Images	Mean Distance Error (1)	Mean Distance Error (2)	Distance Max
Object5-Image11	0,69	0,52	18,00
Object5-Image12	9,80	4,96	222,77
Object5-Image13	1,58	1,23	47,17
Object5-Image14	1,91	1,29	49,82
Object5-Image15	1,37	1,14	44,00

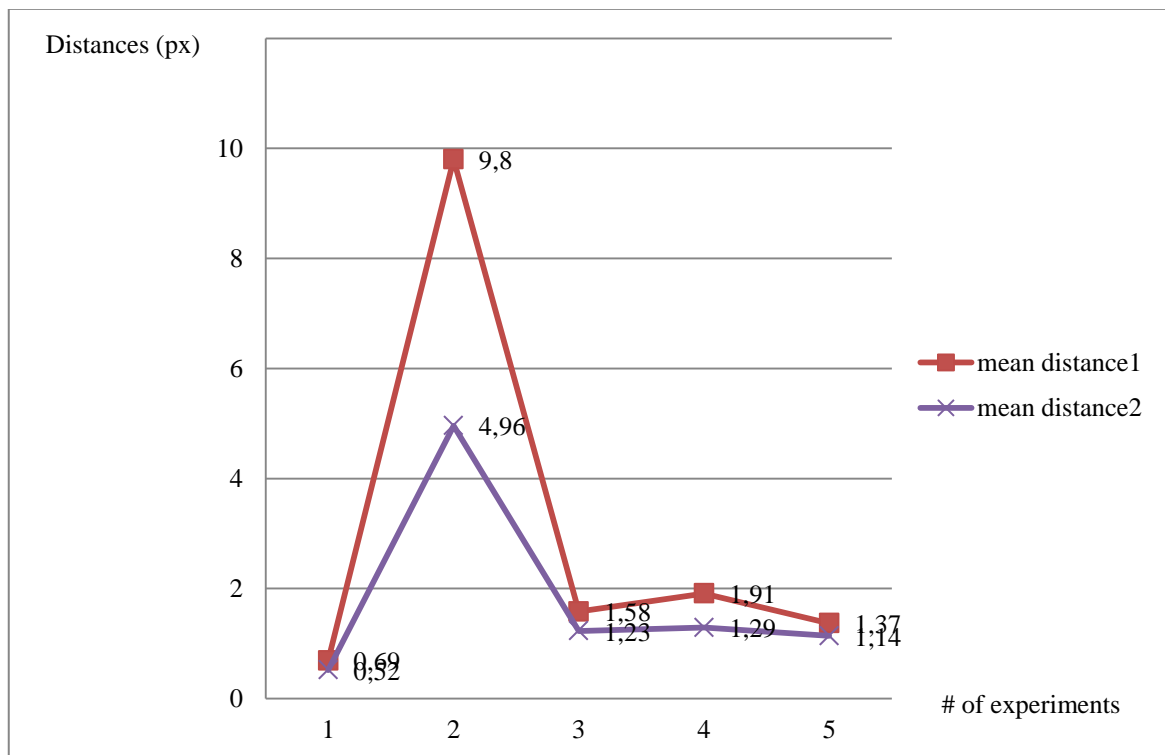


Figure 5.18. Maximum distances and mean distances of faulty objects (object5)

5.3. Software Speed Tests

Finally, the time consumptions for the algorithm that we proposed were calculated. Our proposed system that processed alignment with 56 angle trials was compared with classical 360 angle trials. Results of using fewer pixels for Hausdorff distance calculations were also given with total computation data.

Table 5.16. Time consumption of alignment process for 360 Angle Trials

360 Angle Trials		
Iterations 1 by 1	Time (sec)	# of Computations
1	27	894.600
2	14	447.300
3	10	298.200
4	7	223.650
5	6	178.920

Table 5.17. Time consumption of alignment process for 56 Angle Trials

56 Angle Trials			
Iterations 10 by 10	Iterations 1 by 1	Time (sec)	# of Computations
1	1	6	139.160
2	1	4	94.430
3	1	3	79.520
4	1	3	72.065
5	1	3	67.592
5	2	2	42.742
5	3	1	34.459
5	4	1	30.317
5	5	1	27.832

6. CONCLUSION

In this study, a real-time inspection system was proposed for the tolerance inspection of free form objects. Our system was able to inspect two dimensional free form shapes within given tolerances. Our inspection method is based on Hausdorff distance algorithm that we concentrate on improving the speed and precision of the matching process with using different auxiliary techniques. We developed a fast, robust and automated system that calculates geometrical measurements of manufactured objects.

For further research topic, as Hausdorff distance can be applied to three dimensional objects, our method can also be suitable for inspection of three dimensional forms.

As vision systems carry out statistical data of production, statistical modeling of productivity may become a further research topic. By the statistical information gathered from the inspection process, the system may become self-learning in that it recognizes and classifies recurrent defects.

The system can be implemented conveniently to the precision manufacturing process of free-form surfaces. The tests on sample parts have been carried out to verify the developed techniques. The process time is less than 2 seconds which makes it suitable for industry processes.

REFERENCES

1. Jinkerson R. A., Abrams S. L., Bardis L., Chrysostomidis C., Clement A., Patrikaskis N. M., Wolter F. E., "Inspection and feature extraction of marine propellers", *Journal of Ship Production*, Vol. 9, No. 2, pp. 88-106, 1993.
2. Menq C. H., Yau H. T., Lai G. Y., "Automated precision measurement of surface profile in CAD-directed inspection", *Journal of IEEE Transactions on Robotics and Automation*, Vol. 8, No. 2, pp. 268-278, 1992.
3. Newman T. S., Jain A. K., "A survey of automated visual inspection", *Journal of Computer Vision and Image Understanding*, Vol. 6, No. 2, pp. 231-262, 1995.
4. Cheng W. L., Menq C. H., "Integrated laser/CMM system for the dimensional inspection of objects made of soft material", *International Journal of Advanced Manufacturing Technology*, Vol. 10, pp. 36-45, 1995.
5. Nashman M., Hong T. H., Rippey W. G., Herman M., "An integrated vision touch-probe system for dimensional inspection tasks", *Proceedings of SME Applied Machine Vision '96 Conference*, Cincinnati, pp. 196-205, 1996.
6. Shen T., Huang J., Menq C. H., "Multiple-sensor integration for rapid and high-precision coordinate metrology", *IEEE/ASME Transaction on Mechatronics*, Vol.5, No. 2, pp. 110-121, 2000.
7. Chin R. T., "Automated Visual Inspection: 1981 to 1987", *Computer Vision Graphics and Image Processing*, Vol. 41, pp. 346-381, 1988.
8. Verestoy J. and Chetverikov D., "Shape Defect Detection in Ferrite Cores", *Machine Graphics and Vision*, Vol. 6, No. 2, pp. 225-236, 1997.

9. Unsalan C. and Ercil A., “Automated tolerance inspection by implicit polynomials”, *Proceedings of ICIAP’99*, Venice, Italy, September; pp. 1218-1225, 1999.
10. Gander W., Golub G.H., and Strebel R., “Least squares fitting of circles and ellipses”, *BIT Numerical Mathematics*, Vol. 34, pp. 558-578, 1994.
11. Stojmenovic M., and Nayak A., “Direct Ellipse Fitting and Measuring Based on Shape Boundaries”, *Proceedings of PSIVT 2007*, Vol. 4872, pp. 221-235, 2007.
12. Rosin P. L., “Further five-point fit ellipse fitting”, *Graphical Models and Image Processing*, Vol. 61, No. 5, pp. 245-259, 1999.
13. Yu J., Kulkarni S. R., and Poor H. V., “Robust Fitting of Ellipses and Spheroids”, *Proceedings of the 43rd Asilomar Conference on Signals, Systems and Computers*, pp. 94–98, 2009.
14. Traver V. J., and Pla F., “Dealing with 2D translation estimation in log polar imagery”, *Image Visual Computation*, Vol. 21, No. 2, pp. 145–160, February 2003.
15. Araujo H., and Dias J. M., “An introduction to the logpolar mapping”, *Proceedings of 2nd Workshop Cybernetic Vision*, pp. 139–144, December 1996.
16. Korutcheva E., and Koroutchev K., “Figures design for surface coding with orientation”, *Proceedings of 14th International Workshop on Combinatorial Image Analysis*, Robust Multi-Class Gaussian Process Classification, 2011.
17. Matungka R., Zheng Y. F., and Ewing R. L., “Image registration using adaptive polar transform”, *IEEE Transactions on Image Processing*, Vol. 18, No. 10, pp. 2340–2354, 2009.
18. Pan W., Qin K., and Chen Y., “An adaptable-multilayer fractional fourier transform approach for image registration”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on Image Processing*, Vol. 31, No. 3, pp. 400–414, March 2009.

19. Besl P.J. and McKay N., "A Method for Registration of 3-D Shapes", *Proceedings of IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, pp. 239-256, February 1992.
20. Bispo E. M., and Fisher R. B., "Free-Form Surface Matching for Surface Inspection", *Proceedings of the 6th IMA Conference on the Mathematics of Surfaces*, p. 119-136, September 01, 1994
21. Zinser T., Schmidt J., and Niemann H., "Point Set Registration with Integrated Scale Estimation", *Proceedings of the Eighth International Conference on Pattern Recognition and Image Processing*, R. Sadykhov, S. Ablameiko, A. Doudkin, and L. Podenok, eds., pp. 116–119, 2005.
22. Kaneko S., Kondo T., and Miyamoto A., "Robust Matching of 3D Contours Using Iterative Closest Point Algorithm Improved by M-Estimation", *Pattern Recognition*, Vol. 36, pp. 2041-2047, 2003.
23. Yang S., Wang C., and Chang C., "RANSAC Matching: Simultaneous Registration and Segmentation", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2010)*, Anchorage, Alaska, May 2010.
24. Gelfand N., Ikemoto L., Rusinkiewicz S., and Levoy M., "Geometrically Stable Sampling for the ICP Algorithm", *Fourth International Conference on 3D Digital Imaging and Modeling*, pp. 260-267, 2003.
25. Huttenlocher D. P., Klanderman G. A., and Rucklidge W., "Comparing images using the hausdorff distance", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 9, pp. 858–863, 2009.
26. Rotter P., Skulimowski A.M.J., Kotropoulos C. and Pitas I., "Fast shape matching using the Hausdorff distance", *Mirage 2005: Computer Vision / Computer Graphics Collaboration Techniques and Applications*, March, 1-2 2005, INRIA Rocquencourt, France, 2005.

27. Rucklidge W.J., “Efficiently Locating Objects Using the Hausdorff Distance”, *International Journal of Computer Vision*, Vol. 24, No. 3, pp. 251–271, 1997.
28. Chetverikov D. and Khenokh Y., “Matching for shape defect detection”, *Computer Analysis of Images and Patterns*, pp. 367-374, Springer, Verlag, 1999.
29. Alt H., Braß P., Godau M., Knauer C., and Wenk C., ”Computing the Hausdorff distance of geometric patterns and shapes”, *Discrete and Computational Geometry*, pp. 65–76. Springer, Verlag, 2003.
30. Nutanong S., Jacox E. H., and Samet H., “ An incremental Hausdorff distance calculation algorithm”. *Journal of PVLDB*, Vol. 4, No. 8, pp. 506–517, 2011.
31. Agarwal P. K., Har-Peled S., Sharir M., and Wang Y., “Hausdorff distance under translation for points, disks, and balls”, *Proceedings of the. 19th Annual ACM Symposium on Computational Geometry*, pp. 282–291, 2003.
32. Tang M., Lee M., and Kim Y.J., “Interactive Hausdorff distance computation for general polygonal models”, *ACM Transactions on Graphics (SIGGRAPH)*, 2009.
33. Aspert N., Santa-Cruz D. and Ebrahimi T., “MESH: Measuring Error between Surfaces using the Hausdorff distance”, *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 1, pp. 705-708, 2002.
34. Alt H., and Scharf L., “Computing the Hausdorff distance between sets of curves”, *Proceedings of the 20th European Workshop on Computational Geometry (EWCG)*, pp. 233–236, 2004.

APPENDIX A: ALGORITHMS

Algorithm 5.1. The selection sort algorithm implemented in C++ programming language

```
#include "stdafx.h"
#include <stdio.h>
#include "highgui.h"
#include "cv.h"
#include "cxcore.h"
#include <math.h>
#include <cstdlib>
#include <iostream>
#include <windows.h>
#define square(x) x*x
#define PI 3.14159265
using namespace std;
#include <time.h>

#include <fstream>

// ***** Reference image variables *****
    int size_w1;
    int size_h1;
    int mat1r[20000]={};
int mat1c[20000]={};
uchar* data1;
    int refcenr;
    int refcenc;

// ***** Target image variables *****

uchar* data2;

IplImage* img2;
```

```
IplImage* edge1;

int c1r;
int c1c;

int size_w;
int size_h;

int mat2r[20000]={};
int mat2c[20000]={};
int mattr[20000]={};
int mattc[20000]={};

int k1;
int k2;

// ***** Variables for angle finding *****

int s;
int degree;
int transr;
int transc;
double maxDistBA;
int i;
int j;
double distanceMax;
double distanceH;
double distcont;

double distance1[360] = { };
double degree1[360] = { };
double distance2[360] = { };
double degree2[360] = { };
double mindist;
int degr;
int angle;
double minA;
```

```
double meanMin(int k3)
{
    distanceMax=0;
    int minim;

    int toplam_minim=0;

    for(i=0; i<k1; i+=5) // performing with less pixels with the same intervals
    {
        minim=1000;
        for (j=0; j<k3; j++)
        {
            distanceH=sqrt(square(double(mat1r[i]-
mat1r[j]))+square(double(mat1c[i]-mat1c[j])));

            if (distanceH < minim)
            {
                minim = distanceH;
            }
        }

        toplam_minim = minim + toplam_minim;

        if (distanceMax < minim)
        {
            distanceMax = minim;
        }
    }

    double mean_minim = double (toplam_minim)/ double (k1) ;

    return mean_minim;
}
```

```

}

// ten by ten algorithm
int maxmidist(int degr)
{
    int s;

    for (s=0; s<20; s++)
    {
        int toplam_minim1=0;

        degree=degr+s;
        transr = refcencr - c1r*cos(degree*PI/180) + c1c*sin(degree*PI/180);
        transc = refcenc - c1r*sin(degree*PI/180) - c1c*cos(degree*PI/180);

        for (i=0; i<k2;i++)
        {
            mattr[i]=(mat2r[i]*cos(degree*PI/180))-
(mat2c[i]*sin(degree*PI/180))+transr;
            mattc[i]=(mat2r[i]*sin(degree*PI/180))+
(mat2c[i]*cos(degree*PI/180))+transc;
        }

        // ***** Hausdorff algorithm *****
        maxDistBA=0;
        distanceMax=0;

        int counter=0;

        for(i=0; i<k1; i+=5) // performing with less pixels with the same intervals

```

```

        {
            minA=1000;
            for (j=0; j<k2; j++)
            {
                distanceH=sqrt(square(double(mat1r[i]-
mat1r[j]))+square(double(mat1c[i]-mat1c[j])));

                if (distanceH < minA)
                {
                    minA = distanceH;
                }
            }

            toplam_minim1 = minA + toplam_minim1;
            counter++;

            if (distanceMax < minA)
            {
                distanceMax = minA;
            }
        }

        double mean_minim1 = double (toplam_minim1)/ double (counter) ;

distcont=mean_minim1;

        distance2[s]=distcont;
        degree2[s]=degree;

        if (distcont < mindist)
        {
            mindist = distcont;
        }
    }

```

```

        for (s=0; s<angle;s++)
        {
            if (distance2[s] == mindist)
            {
                break;
            }
        }

        degr=degree2[s];

        return degr;
    }

//*****

// function lefttoright (c1: left point, c2: right point, rt: rowtarget)
int lefttoright (int c1, int c2, int rt, int size_w)
{
    int j;
    for (j=c1;j<c2;j++)
    {
        if( data2[rt*size_w+j] == 255 )
        {
            break;
        }
    }
    return j;
}

//function righttoleft (c1: left point, c2: right point, rt: rowtarget)
int righttoleft (int c1, int c2, int rt, int size_w)
{
    int j;
    for (j=c2;j>=c1;j--)

```



```
    {
        if( data2[rt*size_w+j] == 255 )
        {
            break;
        }
    }
    return j;
}

// function uptodown (r1: up point, r2: down point, ct: columntarget)
int uptodown (int r1, int r2, int ct, int size_w)
{
    int i;
    for (i=r1;i<r2;i++)
    {
        if( data2[i*size_w+ct] == 255 )
        {
            break;
        }
    }
    return i;
}

//function downtoup (r1: up point r2: down point, ct: columntarget)
int downtoup (int r1, int r2, int ct, int size_w)
{
    int i;
    for (i=r2;i>=r1;i--)
    {
        if( data2[i*size_w+ct] == 255 )
        {
            break;
        }
    }
}
```

```
        return i;
    }

    /*******

    // ***** Finding centroid row value *****

    int centroidr (int roi_row1, int roi_col1, int roi_row2, int roi_col2, int size_w)
    {
        int r;
        int c;
        int k=0;
        int rowsum=0;
        int rowavg;

        for (r=roi_row1;r<=roi_row2;r++)
        {
            for (c=roi_col1;c<=roi_col2;c++)
            {
                if( data2[r*size_w+c] == 255 )
                {
                    k++;
                    rowsum=r+rowsum;
                }
            }

            if (k==0)
            {
                k=1;
            }

            rowavg=rowsum/k;

            return rowavg;
        }
    }
```

```
// ***** Finding centroid col value *****
int centroidc (int roi_row1, int roi_col1, int roi_row2,int roi_col2, int size_w)
{
int r;
int c;
int k=0;
int colsum=0;
int colavg;

    for (r=roi_row1;r<=roi_row2;r++)
    {
        for (c=roi_col1;c<=roi_col2;c++)
        {
            if( data2[r*size_w+c] == 255 )
            {
                k++;
                colsum=c+colsum;
            }
        }

        if (k==0)
        {
            k=1;
        }

        colavg=colsum/k;

    return colavg;
}

//*****
```

```
//// ***** Finding diameter *****  
float diameter(int roi_row1, int roi_col1, int roi_row2, int roi_col2)  
{  
  
int right;  
int left;  
int up;  
int down;  
  
int c1r = centroidr (roi_row1, roi_col1, roi_row2, roi_col2, size_w);  
int c1c = centroidc (roi_row1, roi_col1, roi_row2, roi_col2, size_w);  
  
right = lefttoright (c1c, size_w , c1r, size_w);  
int radius1 = right-c1c;  
  
left = righttoleft (0, c1c , c1r, size_w);  
int radius2 = c1c-left;  
  
up = downtoup (0, c1r , c1c , size_w);  
int radius3 = c1r-up;  
  
down = uptodown (c1r, size_h, c1c, size_w);  
int radius4 = down - c1r;  
  
float diameter;  
  
diameter= float (radius1+radius2+radius3+radius4);  
  
diameter=diameter/2;  
  
return diameter;  
}
```

```

//*****

// ***** Finding coordinates by mouse click *****
void on_mouse( int event, int x, int y, int flags, void* param )
{
    if ( event == CV_EVENT_LBUTTONDOWN )
        printf("row:%d\n column:%d \n ", y, x);
}

//*****

// ***** bwareaopen *****
void bwareaopen(IplImage* image, int size)
{
    CvMemStorage *storage;
    CvSeq *contour;
    IplImage *input;
    double area;

    if (image == NULL || size == 0)
        return;

    input = cvCloneImage(image);
    storage = cvCreateMemStorage(0);

    cvFindContours(input, storage, &contour, sizeof (CvContour),
        CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));

    while(contour)
    {
        area = cvContourArea(contour, CV_WHOLE_SEQ, 1);

        if (-size <= area && area <= 0)
        {
            // removes white dots
            cvDrawContours(image, contour, CV_RGB(0,0,0), CV_RGB(0,0,0), -1, CV_FILLED, 8,
                cvPoint(0, 0));
        }
    }
}

```

```

    }
    else if (0 < area && area <= size)
    {
        // fills in black holes
        cvDrawContours(image, contour, CV_RGB(0xff,0xff,0xff),CV_RGB(0xff,0xff,0xff), -1,
CV_FILLED, 8, cvPoint(0,0));
    }

    contour = contour->h_next;
}

cvReleaseMemStorage(&storage);
cvReleaseImage(&input);
}

// ***** Rotate image *****
IplImage *rotateImage(const IplImage *src, float angleDegrees)
{

    float m[6]; // Create a map_matrix, where the left 2x2 matrix
    CvMat M = cvMat(2, 3, CV_32F, m); // is the transform and the right 2x1 is the
dimensions.
    int w = src->width;
    int h = src->height;

    float angleRadians = angleDegrees * ((float)CV_PI / 180.0f);
    m[0] = (float)( cos(angleRadians) );
    m[1] = (float)( sin(angleRadians) );
    m[3] = -m[1];
    m[4] = m[0];
    m[2] = w*0.5f;
    m[5] = h*0.5f;

    CvSize sizeRotated; // Make a spare image for the result
    sizeRotated.width = cvRound(w);
    sizeRotated.height = cvRound(h);

    IplImage *imageRotated = cvCreateImage( sizeRotated, src->depth, src-
>nChannels ); // Rotate

```

```

        cvGetQuadrangleSubPix( src, imageRotated, &M); // Transform the image

        return imageRotated;
    }

//
*****
*****

// ***** main program
*****

int main( int argc, char** argv )
{

// ***** reference image datas *****

IplImage* img1 = cvLoadImage( "C:\\appgui\\mec\\reference.jpg" ); // load reference image

IplImage* img1g = cvCreateImage( cvSize( img1->width,img1->height ), img1->depth, 1);
// create an image for grayscale
cvCvtColor(img1, img1g ,CV_BGR2GRAY); // convert rgb image to grayscale image
IplImage* BW1 = cvCreateImage( cvSize(img1g->width,img1g->height), img1g->depth,
img1g->nChannels ); // Create an image for BW
edge1 = cvCreateImage( cvSize(img1g->width,img1g->height), img1g->depth, img1g-
>nChannels ); // Create an image for edge

cvThreshold(img1g, BW1, 80, 255, CV_THRESH_BINARY);
bwareaopen(BW1, 10);

```

```

cvSaveImage("C:\\appgui\\mec\\results\\thresh.jpg",BW1);

CvMat* mask=0;
mask = cvCreateMat(3,3,CV_32FC1);
cvSet2D( mask, 0, 0, cvRealScalar(0) );
cvSet2D( mask, 0, 1, cvRealScalar(-1) );
cvSet2D( mask, 0, 2, cvRealScalar(0) );
cvSet2D( mask, 1, 0, cvRealScalar(-1) );
cvSet2D( mask, 1, 1, cvRealScalar(4) );
cvSet2D( mask, 1, 2, cvRealScalar(-1) );
cvSet2D( mask, 2, 0, cvRealScalar(0) );
cvSet2D( mask, 2, 1, cvRealScalar(-1) );
cvSet2D( mask, 2, 2, cvRealScalar(0) );

cvFilter2D( BW1, edge1, mask, cvPoint(-1,-1) );

data1 = (uchar *)edge1->imageData; // imageData dan gelecek veriyi data1 arrayine at

size_w1 = img1->width;
size_h1 = img1->height;

// ***** object coordinates to two arrays *****
k1=0;
for (i=0; i<(size_w1*size_h1);i++)
{
    if ( data1[i]== 255)
    {
        mat1r[k1]=i/size_w1;
        mat1c[k1]=i%size_w1;
        k1++;
    }
}

// ***** Reference image Centroid Coordinates
*****

// for image1

```



```
int trow1=0;
for (i=0; i<k1;i++)
{
trow1 += mat1r[i];
}

int tcoll=0;
for (i=0; i<k1;i++)
{
tcoll += mat1c[i];
}

refcenr=trow1/k1;
refcenc=tcoll/k1;

ROIs for the target object1

// ROIs for Height (uzunluk 1)

int height_r1 = 176;
int height_c1 = 598;

int height_r2 = 820;
int height_c2 = height_c1;

// ROIs for Width (uzunluk 2)

int width_r1 = 605;
int width_c1 = 336;

int width_r2 = 605;
int width_c2 = 1020;

// ROIs for uzunluk 3-4 (elips)
int ellipse_r1 = 284;
int ellipse_c1 = 706;
```

```
int ellipse_r2 = 474;
int ellipse_c2 = 822;

// ROIs for Circle1 (cap 1)
int circle1_r1 = 310;
int circle1_c1 = 492;

int circle1_r2 = 400;
int circle1_c2 = 582;

// ROIs for Circle2 (cap 2)
int circle2_r1 = 626;
int circle2_c1 = 592;

int circle2_r2 = 716;
int circle2_c2 = 698;

restart:

int iii=0;
char folder[256];

CvCapture* capture = cvCreateCameraCapture( 2 ); // 2 is for external cam
assert( capture );

IplImage *image = cvQueryFrame( capture );
cvNamedWindow( "video_stream",0 );
cvResizeWindow( "video_stream", 640, 512 );

while(image)
```

```

{
    IplImage *image = cvQueryFrame( capture );
        iii++;

        char c = cvWaitKey(1);
    if(c == 99)
    {
        sprintf(folder,"%s\\img-%d.jpg", "C:\\appgui\\results",iii);
        cvSaveImage(folder,image);
        break;
    }

        cvShowImage( "video_stream", image);
        char e = cvWaitKey(1);
    if( e == 27 )
        {
            goto exit;
        }
    }

img2 = cvLoadImage( folder );

//***** Inspect Command *****

IplImage* img2g = cvCreateImage( cvSize( img2->width,img2->height ), img2->depth, 1);
// create an image for grayscale
IplImage* BW2 = cvCreateImage( cvSize(img2g->width,img2g->height), img2g->depth,
img2g->nChannels ); // Create an image for BW
IplImage* edge2 = cvCreateImage( cvSize(img2g->width,img2g->height), img2g->depth,
img2g->nChannels ); // Create an image for edge

cvCvtColor(img2, img2g ,CV_BGR2GRAY); // convert rgb image to grayscale image
cvThreshold(img2g, BW2, 80, 255, CV_THRESH_BINARY);
bwareaopen(BW2, 10);

```

```

cvFilter2D( BW2, edge2, mask, cvPoint(-1,-1) );

        size_w = img2->width;
        size_h = img2->height;

data2 = (uchar *)edge2->imageData;

// ***** object coordinates to two arrays *****

k2=0;

for (i=0; i<(size_w*size_h);i++)
{
    if ( data2[i]== 255)
    {
        mat2r[k2]=i/size_w;
        mat2c[k2]=i%size_w;
        k2++;
    }
}

// ***** Target image Centroid Coordinates *****

c1r = centroidr (1,1,size_h,size_w,size_w);
c1c = centroidc (1,1,size_h,size_w,size_w);

mindist = 10000;
angle = 360;

for (s=0; s<angle; s+=10)
{
    degree = s;
    transr = refcencr - c1r*cos(degree*PI/180) + c1c*sin(degree*PI/180);
    transc = refcenc - c1r*sin(degree*PI/180) - c1c*cos(degree*PI/180);
}

```

```

for (i=0; i<k2; i++)
{
    mattr[i]=(mat2r[i] * cos(degree*PI/180)) - (mat2c[i] * sin(degree*PI/180))
+ transr;
    mattc[i]=(mat2r[i] * sin(degree*PI/180)) + (mat2c[i] *
cos(degree*PI/180)) + transc;
}

// ***** Hausdorff algorithm *****

maxDistBA=0;
distanceMax=0;

for(i=0; i<k1; i+=5) // performing with less pixels with the same intervals

{
    minA=1000;
    for (j=0; j<k2; j++)
    {
        distanceH=sqrt(square(double(mat1r[i]-
mattr[j]))+square(double(mat1c[i]-mattc[j])));

        if (distanceH < minA)
        {
            minA = distanceH;
        }
    }

    if (distanceMax < minA)
    {
        distanceMax = minA;
    }
}

distcont=distanceMax;
distance1[s]=distcont;
degree1[s]=degree;

```

```
        if (distcont < mindist)
        {
            mindist = distcont;
        }
    }

    for (s=0; s<angle;s++)
    {
        if (distance1[s] == mindist)
        {
            break;
        }
    }

    degr=degree1[s];

    if (degr==0)
    {
        degr = maxmiddist(350);
    }
    else
    {
        degr = maxmiddist(degr-10);
    }

    // ***** Create translated image *****

    BW2=rotateImage(BW2, -degr);

    IplImage* edge3 = cvCreateImage( cvSize(img2g->width,img2g->height), img2g->depth,
    img2g->nChannels ); // Create an image for edge
```

```

cvFilter2D( BW2, edge3, mask, cvPoint(-1,-1) );

int mat3r [10000]= {};
int mat3c [10000]= {};

uchar* data3;
data3 = (uchar *)edge3->imageData;

// ***** object coordinates to two arrays *****
int k3=0;
for (i=0; i<(size_w1*size_h1);i++)
{
    if ( data3[i]== 255)
    {
        mat3r[k3]=i/size_w;
        mat3c[k3]=i%size_w;
        k3++;
    }
}

// ***** Rotated Target image Centroid Coordinates
*****

int rthrow1=0;
for (i=0; i<k3;i++)
{
    rthrow1 += mat3r[i];
}

int rtcoll=0;
for (i=0; i<k3;i++)
{
    rtcoll += mat3c[i];
}

int c2r=rthrow1/k3;
int c2c=rtcoll/k3;

```

```
int transr2=refcendr - c2r;
int transc2=refcenc - c2c;

printf("transr2 : %d \n",transr2 );
printf("transc2 : %d \n",transc2 );

for (i=0; i<k3;i++)
{
mattr[i]= mat3r[i] + transr2;
mattc[i]= mat3c[i] + transc2;
}

double mean_minim;

bool bCont = true;
while(bCont)
{
    double mean_minall = 9999;
    int iShift = 0;
    mean_minim = meanMin(k3);

    if (mean_minim < mean_minall)
    {
        mean_minall = mean_minim;
        iShift = 0;
    }

    for (i=0; i<k3;i++)
    {
        mattr[i]= mattr[i] - 1;
    }
    double mean_minim_r_1 = meanMin(k3);

    if (mean_minim_r_1 < mean_minall)
    {
```



```
        mean_minall = mean_minim_r_1;
        iShift = -1;
    }

    for (i=0; i<k3;i++)
    {
        mattr[i]= mattr[i] + 2;
    }
    double mean_minim_r1 = meanMin(k3);

    if (mean_minim_r1 < mean_minall)
    {
        mean_minall = mean_minim_r1;
        iShift = 1;
    }

    for (i=0; i<k3;i++)
    {
        mattr[i]= mattr[i] - 1;
        mattc[i]= mattc[i] - 1;
    }
    double mean_minim_c_1 = meanMin(k3);

    if (mean_minim_c_1 < mean_minall)
    {
        mean_minall = mean_minim_c_1;
        iShift = -2;
    }

    for (i=0; i<k3;i++)
    {
        mattc[i]= mattc[i] + 2;
    }
    double mean_minim_c1 = meanMin(k3);

    if (mean_minim_c1 < mean_minall)
    {
        mean_minall = mean_minim_c1;
        iShift = 2;
```

```
    }

    for (i=0; i<k3;i++)
    {
        mattc[i]= mattc[i] - 1;
    }

    if (iShift == -1)
    {
        for (i=0; i<k3;i++)
        {
            mattr[i]= mattr[i] - 1;
        }
    }
    else
        if (iShift == 1)
        {
            for (i=0; i<k3;i++)
            {
                mattr[i]= mattr[i] + 1;
            }
        }
    else
        if (iShift == -2)
        {
            for (i=0; i<k3;i++)
            {
                mattc[i]= mattc[i] - 1;
            }
        }
    else
        if (iShift == 2)
        {
            for (i=0; i<k3;i++)
            {
                mattc[i]= mattc[i] + 1;
            }
        }
}
```

```

    }

    if (iShift == 0)
        break;
}

// ***** comparing the new image with reference *****
IplImage* img9 = cvCreateImage( cvSize( img1->width,img1->height ), img1->depth, 3); //
create an image for grayscale

// ***** tolerance inspection phase *****

maxDistBA=0;
distanceMax=0;

for(i=0; i<k1; i+=1) // performing with less pixels with the same intervals
{
    minA=1000;
    for (j=0; j<k3; j++)
    {
        distanceH=sqrt(square(double(mat1r[i]-
mat1c[j]))+square(double(mat1c[i]-mat1c[j])));

        if (distanceH < minA)
        {
            minA = distanceH;
        }
    }

    if (minA > 10) // tolerance inspection
    {
        CvPoint p1m=cvPoint(mat1c[i], mat1r[i]);
        cvCircle(img9, p1m, 10, cvScalar(255, 0, 255, 0), 1);
    }
}

```

```
        if (distanceMax < minA)
        {
            distanceMax = minA;
        }
    }

    for (i=0;i<k3;i++)
    {
        CvPoint p1t=cvPoint(matte[i], mattr[i]);
        CvPoint p2t=cvPoint(matte[i], mattr[i]);
        cvLine(img9,p1t, p2t, cvScalar(0, 0, 255, 0), 1);
    }

    for (i=0;i<k1;i++)
    {
        CvPoint p1r=cvPoint(mat1c[i], mat1r[i]);
        CvPoint p2r=cvPoint(mat1c[i], mat1r[i]);
        cvLine(img9,p1r, p2r, cvScalar(255, 255, 255, 0), 1);
    }

    IplImage* imgt = cvCreateImage( cvSize(size_w,size_h), img2->depth, 1);

    for (i=0;i<k3;i++)
    {
        CvPoint p1=cvPoint(matte[i], mattr[i]);
        CvPoint p2=cvPoint(matte[i], mattr[i]);
        cvLine(imgt,p1, p2, cvScalar(255, 255, 255, 255), 1);
    }

    data2 = (uchar *)imgt->imageData;
```

```

// ***** measurements *****

double cal=0.13;

// measurement of height
int meayup = uptodown(1,size_h,height_c1,size_w);
int meaydown = downtoup(1,size_h,height_c1,size_w);
int meay=meaydown-meayup;
double meaymm= double (meay) * cal;
meay=meaymm;

// measurement of Width
int meaxleft = lefttoright(1,size_w-1,width_r1,size_w);
int meaxright = righttoleft(1,size_w-1,width_r1,size_w);
int meax=meaxright-meaxleft;
double meaxmm= double (meax) * cal;
meax=meaxmm;

// measurement of uzunluk 3-4 (ellipse)
int c1r_e = centroidr (ellipse_r1, ellipse_c1, ellipse_r2, ellipse_c2, size_w);
int c1c_e = centroidc (ellipse_r1, ellipse_c1, ellipse_r2, ellipse_c2, size_w);

int el1uz3 = righttoleft (ellipse_c1, c1c_e , c1r_e, size_w);
int el2uz3 = lefttoright (c1c_e, ellipse_c2 , c1r_e, size_w);
int eluz3 = el2uz3 - el1uz3;
double eluz3mm= double (eluz3) * cal;
eluz3=eluz3mm;

int el1uz4 = downtoup (ellipse_r1, c1r_e, c1c_e, size_w);
int el2uz4 = uptodown (c1r_e, ellipse_r2, c1c_e, size_w);
int eluz4 = el2uz4 - el1uz4;
double eluz4mm= double (eluz4) * cal;
eluz4=eluz4mm;

// measurement of diameter1(circle1)
float dia1= diameter (circle1_r1, circle1_c1, circle1_r2, circle1_c2);
double dia1mm= double (dia1) * cal;
dia1=dia1mm;

```

```

// measurement of diameter2(circle2)
float dia2= diameter (circle2_r1, circle2_c1, circle2_r2, circle2_c2);
double dia2mm= double (dia2) * cal;
dia2=dia2mm;

//----- screen print -----
char sbuf[10];
CvFont font;
cvInitFont (&font, CV_FONT_HERSHEY_SIMPLEX, 1.0, 1.0, 0, 2, CV_AA);
//----- end of screen print -----

// ***** ROI positioning-1 height *****
degr=-degr;

      transr = c1r - refcenc*cos(degr*PI/180) + refcenc*sin(degr*PI/180);
      transc = c1c - refcenc*sin(degr*PI/180) - refcenc*cos(degr*PI/180);

int mroi1rt=(meayup*cos(degr*PI/180)) - (height_c1*sin(degr*PI/180)) + transr;
int mroi1ct=(meayup*sin(degr*PI/180)) + (height_c1*cos(degr*PI/180)) + transc;
int mroi2rt=(meaydown*cos(degr*PI/180)) - (height_c2*sin(degr*PI/180)) + transr;
int mroi2ct=(meaydown*sin(degr*PI/180)) + (height_c2*cos(degr*PI/180)) + transc;

CvPoint pt1t = { mroi1ct,mroi1rt};
CvPoint pt2t = { mroi2ct,mroi2rt};

cvLine (img2, pt1t, pt2t,cvScalar(0, 255, 255, 0),2, 8, 0 );
cvPutText (img2, " M1", pt1t, &font, cvScalar (0, 255, 255, 0));

itoa(meaymm, sbuf, 10);
int writeline1=50;

cvPutText (img2, "M1= ", cvPoint(20,writeline1), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline1), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline1), &font, cvScalar (255,0,0));

```

```

int tol1 = 2;
int ref1 = 60;

if ( (ref1+tol1 >= meaymm) && (meaymm >= ref1-tol1) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline1), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(300,writeline1), &font, cvScalar (0, 0, 255, 0));
}

// ***** ROI positioning-2 width *****
mroi1rt=(width_r1*cos(degr*PI/180)) - (meaxleft*sin(degr*PI/180)) + transr;
mroi1ct=(width_r1*sin(degr*PI/180)) + (meaxleft*cos(degr*PI/180)) + transc;
mroi2rt=(width_r2*cos(degr*PI/180)) - (meaxright*sin(degr*PI/180)) + transr;
mroi2ct=(width_r2*sin(degr*PI/180)) + (meaxright*cos(degr*PI/180)) + transc;

CvPoint pt1t2 = {mroi1ct,mroi1rt};
CvPoint pt2t2 = {mroi2ct,mroi2rt};

cvLine (img2, pt1t2, pt2t2,cvScalar(0, 255, 255, 0),2, 8, 0 );
cvPutText (img2, " M2", pt1t2, &font, cvScalar (0, 255, 255, 0));

itoa(meaxmm, sbuf, 10);
int writeline2=100;

cvPutText (img2, "M2= ", cvPoint(20,writeline2), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline2), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline2), &font, cvScalar (255,0,0));

int tol2 = 2;
int ref2 = 51;

```

```

if ( (ref2+tol2 >= meaxmm) && (meaxmm >= ref2-tol2) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline2), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(300,writeline2), &font, cvScalar (0, 0, 255, 0));
}

// ***** ROI positioning-3 Ellipse 3 *****
mroi1rt=(c1r_e*cos(degr*PI/180)) - (el1uz3*sin(degr*PI/180)) + transr;
mroi1ct=(c1r_e*sin(degr*PI/180)) + (el1uz3*cos(degr*PI/180)) + transc;
mroi2rt=(c1r_e*cos(degr*PI/180)) - (el2uz3*sin(degr*PI/180)) + transr;
mroi2ct=(c1r_e*sin(degr*PI/180)) + (el2uz3*cos(degr*PI/180)) + transc;

CvPoint pt1t3 = {mroi1ct,mroi1rt};
CvPoint pt2t3 = {mroi2ct,mroi2rt};

cvLine (img2, pt1t3, pt2t3,cvScalar(0, 255, 255, 0),2, 8, 0 );
cvPutText (img2, " M3", pt1t3, &font, cvScalar (0, 255, 255, 0));

itoa(eluz3, sbuf, 10);
int writeline3=150;

cvPutText (img2, "M3= ", cvPoint(20,writeline3), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline3), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline3), &font, cvScalar (255,0,0));

int tol3 = 2;
int ref3 = 8;

if ( (ref3+tol3 >= eluz3) && (eluz3 >= ref3-tol3) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline3), &font, cvScalar (0, 255, 0, 0));
}
else

```



```

{
    cvPutText (img2, "NOK", cvPoint(300,writeline3), &font, cvScalar (0, 0, 255, 0));
}

// ***** ROI positioning-4 Ellipse 4 *****
mroi1rt=(el1uz4*cos(degr*PI/180)) - (c1c_e*sin(degr*PI/180)) + transr;
mroi1ct=(el1uz4*sin(degr*PI/180)) + (c1c_e*cos(degr*PI/180)) + transc;
mroi2rt=(el2uz4*cos(degr*PI/180)) - (c1c_e*sin(degr*PI/180)) + transr;
mroi2ct=(el2uz4*sin(degr*PI/180)) + (c1c_e*cos(degr*PI/180)) + transc;

CvPoint pt1t4 = {mroi1ct,mroi1rt};
CvPoint pt2t4 = {mroi2ct,mroi2rt};

cvLine (img2, pt1t4, pt2t4,cvScalar(0, 255, 255, 0),2, 8, 0 );
cvPutText (img2, " M4", pt1t4, &font, cvScalar (0, 255, 255, 0));

itoa(eluz4, sbuf, 10);
int writeline4=200;

cvPutText (img2, "M4= ", cvPoint(20,writeline4), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline4), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline4), &font, cvScalar (255,0,0));

int tol4 = 2;
int ref4 = 19;

if ( (ref4+tol4 >= eluz4) && (eluz4 >= ref4-tol4) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline4), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(300,writeline4), &font, cvScalar (0, 0, 255, 0));
}

// ***** ROI positioning-5 Dial *****
mroi1rt=(circle1_r1*cos(degr*PI/180)) - (circle1_c1*sin(degr*PI/180)) + transr;
mroi1ct=(circle1_r1*sin(degr*PI/180)) + (circle1_c1*cos(degr*PI/180)) + transc;
mroi2rt=(circle1_r2*cos(degr*PI/180)) - (circle1_c2*sin(degr*PI/180)) + transr;

```

```

mroi2ct=(circle1_r2*sin(degr*PI/180)) + (circle1_c2*cos(degr*PI/180)) + transc;

CvPoint pt1t5 = {mroi1ct,mroi1rt};
CvPoint pt2t5 = {mroi2ct,mroi2rt};
CvPoint pt0t5= {(mroi1ct+mroi2ct)/2,(mroi1rt+mroi2rt)/2};

cvPutText (img2, " M5", pt0t5, &font, cvScalar (0, 255, 255, 0));

itoa(dia1, sbuf, 10);
int writeline5=250;

cvPutText (img2, "M5= ", cvPoint(20,writeline5), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline5), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline5), &font, cvScalar (255,0,0));

int tol5 = 2;
int ref5 = 6;

if ( (ref5+tol5 >= dia1) && (dia1 >= ref5-tol5) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline5), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(300,writeline5), &font, cvScalar (0, 0, 255, 0));
}

// ***** ROI positioning-6 Dia2 *****
mroi1rt=(circle2_r1*cos(degr*PI/180)) - (circle2_c1*sin(degr*PI/180)) + transr;
mroi1ct=(circle2_r1*sin(degr*PI/180)) + (circle2_c1*cos(degr*PI/180)) + transc;
mroi2rt=(circle2_r2*cos(degr*PI/180)) - (circle2_c2*sin(degr*PI/180)) + transr;
mroi2ct=(circle2_r2*sin(degr*PI/180)) + (circle2_c2*cos(degr*PI/180)) + transc;

CvPoint pt1t6 = {mroi1ct,mroi1rt};
CvPoint pt2t6 = {mroi2ct,mroi2rt};
CvPoint pt0t6= {(mroi1ct+mroi2ct)/2,(mroi1rt+mroi2rt)/2};

```

```

cvPutText (img2, " M6", pt0t6, &font, cvScalar (0, 255, 255, 0));

itoa(dia2, sbuf, 10);
int writeline6=300;

cvPutText (img2, "M6= ", cvPoint(20,writeline6), &font, cvScalar (255, 0, 0, 0));
cvPutText (img2,sbuf,cvPoint(120,writeline6), &font, cvScalar(255,0,0));
cvPutText (img2, "mm", cvPoint(200,writeline6), &font, cvScalar (255,0,0));

int tol6 = 2;
int ref6 = 6;

if ( (ref6+tol6 >= dia1) && (dia2 >= ref6-tol6) )
{
    cvPutText (img2, "OK", cvPoint(300,writeline6), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(300,writeline6), &font, cvScalar (0, 0, 255, 0));
}

// Tolerance inspection
cvPutText (img2, " TOLERANCE INSPECTION:", cvPoint(400,40), &font, cvScalar (255,
0, 0, 0));
if ( distanceMax<10 )
{
    cvPutText (img2, "OK", cvPoint(800,40), &font, cvScalar (0, 255, 0, 0));
}
else
{
    cvPutText (img2, "NOK", cvPoint(800,40), &font, cvScalar (0, 0, 255, 0));
}

// ***** Showing the result image *****

```

```
cvShowImage( "video_stream", img2);

char a = cvWaitKey(0);
    if( a == 97 )
    {
        goto restart;
    }

exit:

// ***** Releasing the memory *****
cvReleaseImage( &img1g);
cvReleaseImage( &img2g);
cvReleaseImage( &BW1);
cvReleaseImage( &BW2);
cvReleaseImage( &edge1);
cvReleaseImage( &edge2);
cvReleaseImage( &imgt);

}
```