THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

POSE ESTIMATION FROM 2D IMAGES BY USING 3D PRIOR

Master of Science Thesis

HUSEYIN INAN

İSTANBUL, 2015

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

POSE ESTİMATION FROM 2D IMAGES BY USING 3D PRIOR

Master of Science Thesis

HÜSEYİN İNAN

Supervisor: Asst. Prof. Dr. Övgü ÖZTÜRK ERGÜN

İSTANBUL, 2015

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Title of the Master's Thesis : Pose Estimation From 2D images by Using 3D Prior
Name/Last Name of the Student : Hüseyin İnan
Date of Thesis Defense : 02.06.2015

This thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assoc. Prof. Nafiz ARICA
Graduate School Director

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

Asst. Prof. Tarkan Aydın
Program Coordinator

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

Examining Committee Members	Signature
Thesis Supervisor: Asst. Prof. Dr. Övgü ÖZTÜRK ERGÜN	.....
Member: Asst. Prof. Dr. Tarkan AYDIN	.....
Member: Assoc. Prof. Dr. Bahadır Kürşat GÜNTÜRK	.....

ACKNOWLEDGEMENTS

First and foremost, I would like to thanks my advisor Asst. Prof. Dr. Övgü Öztürk Ergün who has continuously shared her academic and personal support and enthusiasm with her advises. I would like to appreciate to encourage me to research on this thesis.

I would also like to immensely thank my lab partners Bahtiyar Kaba, Şan Güneş Onur Özduru and Okan Şanlı for their moral support and their willingness to share time and knowledge.

Special thanks as well to my parents, my sister and my brother. They always support on me with my carrier and my researches. Words cannot tell their worth in my life. I also want to thank you my homemates Murat Çulha and Okan Doğan. Spending pleasure times and with them is really great.

I would like to say a big thank you for my "forever" friends Burak Eren, Selin Erdoğan, Sinem Dur, Gizem Güngör, Sinem Körlü, Yağmur Çınar, Deniz Can, Ceren Aydın and Gökhan Güler for their great friendship and priceless support.

İstanbul, 2015 Hüseyin İnan

ABSTRACT

POSE ESTIMATION FROM 2D IMAGES BY USING 3D PRIOR

Hüseyin İnan

Computer Engineering

Thesis Supervisor: Asst. Prof. Dr. ÖvgüÖztürkErgün

May 2015, 64 pages

Object recognition and 3D pose estimation is one of the popular topics of computer vision because of vast applicability in the fields such as robotics, scene understanding, augmented reality etc. While many methods are developed to deal with pose estimation problem, they have at least one of these problems; not real-time capable, or not robust to occlusion or fail in cluster environments or specific to a class like car, human etc. Additionally suggested reliable methods depend on initialized correspondences or they track 3D models based on their features. Therefore, initial pose estimation is still unsolved issue for computer vision. The complication originates from the requirement for quick and strong detection of known objects in the image.

In this dissertation, we research estimation pose of 3D rigid objects with a 3D mesh prior. 3D mesh of the object is constructed by using Kinect. Our work can be modularized as feature extraction from 2D image, 2D-2D corresponding point establishment, 2D-3D corresponding point establishment and fine pose estimation.

The result of the pose estimation algorithm is evaluated by using three objects; Buda head, boy and angel statue. Estimation is tested from variety angles. Error in degrees and estimation consistency are analyzed. We analyze robustness of our algorithm to far initial given poses. Additionally robustness to scale variances of the algorithm and the error variance is also examined by using the boy statue.

Keywords: 3D-2D Pose estimation

ÖZET

2-BOYUTLU RESİMDEN, 3-BOYUTLU ÜÇGENLENMİŞ BULUT YARDIMIYLA POZ TAHMİNİ YAPMA

Hüseyin İnan

Bilgisayar Mühendisliği

Tez Danışmanı: Yrd. Doç. Dr. Övgü Öztürk Ergün

Mayıs 2015, 64 sayfa

Nesne tanıma ve 3 boyutta pozunu tahmin etme bilgisayar görmenin popüler konularından biridir. Çünkü robotla ilgili uygulamalarda, sahne anlamlandırmada veya arttırılmış gerçeklik gibi bir çok geniş konu aralığında kullanımına ihtiyaç duyulmaktadır. Poz tahmin etme ile ilgili birçok yöntem geliştirilmiş olunsa da, bu yöntemlerin gerçek zamanlı çalışmamak, cismin bir kısmı başka bir cisim tarafından kapatıldığında pozu doğru tahmin edememe veya çok fazla cismin olduğu ortamlarda pozu tahmin etmekte zorluklar gibi bir veya birden fazla problemlere sahiptirler. Önerilen güvenilir yöntemler genelde başlangıçta benzer nokta eşleşmelerinin var olduğu saymaktadırlar veya özellik merkezli 3 boyutta obje takip etmektedirler. Fakat, bir objenin ilk pozunu tahmin etme bilgisayar görmenin hala çözülememiş problemlerinden biridir. İşin zorluğu belirli bir obje için hızlı ve dirençli poz tespitine ihtiyaçtan gelmektedir.

Bu tezde, 3 boyutlu sabit bir cismin pozunu tahmin etme ile ilgili araştırma yaptım. poz tahmin etme aşamasında yardımcı olarak cismin 3 boyutlu üçgenlenmiş nokta bulutuna sahip olduğumuzu varsaydık. Yapılan yöntem 2 boyutlu resimden özellik çıkarma, 2 boyut - 2 boyut eş nokta belirleme, 2 boyut - 3 boyut eş nokta belirleme ve ince poz tahmini yapma gibi parçalara bölünebilir.

Algoritmanın sonuçları Buda başı, çocuk ve melek heykeli olmak üzere üç tane obje kullanılarak değerlendirildi. Objelerin çeşitli açılardan pozları tahmin edilmeye çalışıldı. Derece olarak hatalar ve poz tahmin etme yönteminin tutarlılığı incelendi. Geliştirilen

yöntemin uzak poz tahminlerine rağmen pozu doru bulup bulamaması incelendi. Bunlara ek olarak boyut farklılıklarına rağmen pozun bulunabilmesi ve hata oranındaki değişimler incelendi.

AnahtarKelimeler: 3B-2B Poz tahmini

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

3D	: Three-dimensional
ACO	: Ant colonization algorithm
BB	: Branch and band
BOB	: Bag of Boundaries
CAD	: Computer-aided design
CCD	: Charge coupled devices
CRF	: Camera reference frame
CVFH	: The clustered viewpoint feature histogram
DPM	: Deformable part models
HOG	: Histogram of gradients
ICP	: Iterative closest points
QAP	: Quadratic Assignment Problem
RGB-D	: Red, green, blue, depth
SVD	: Singular value decomposition
SVM	: Support vector machine
VFH	: Viewpoint feature histogram
WCF	: World coordinate frame

1. INTRODUCTION

Estimating the pose of 3D rigid objects is one of the major topics of computer vision. It is a very popular research field in recent years. Various types of objects are used to estimate their poses. Addition to objects, human pose estimation is also one of the very interesting themes. Determining the pose of the object is attractive, because there are plenty of implementation areas of the method such as human-computer interaction applications, robot controlling and games. For instance, human pose estimation is used in game industry in recent years. Addition to the game sector, estimating the pose of humans can be also applied for medical applications.

Nonetheless, state of the art methods for estimating the pose of objects from the 2D image do not satisfy the demands of modern technologies. The demand is estimating the pose of the object in real time and in the real world.

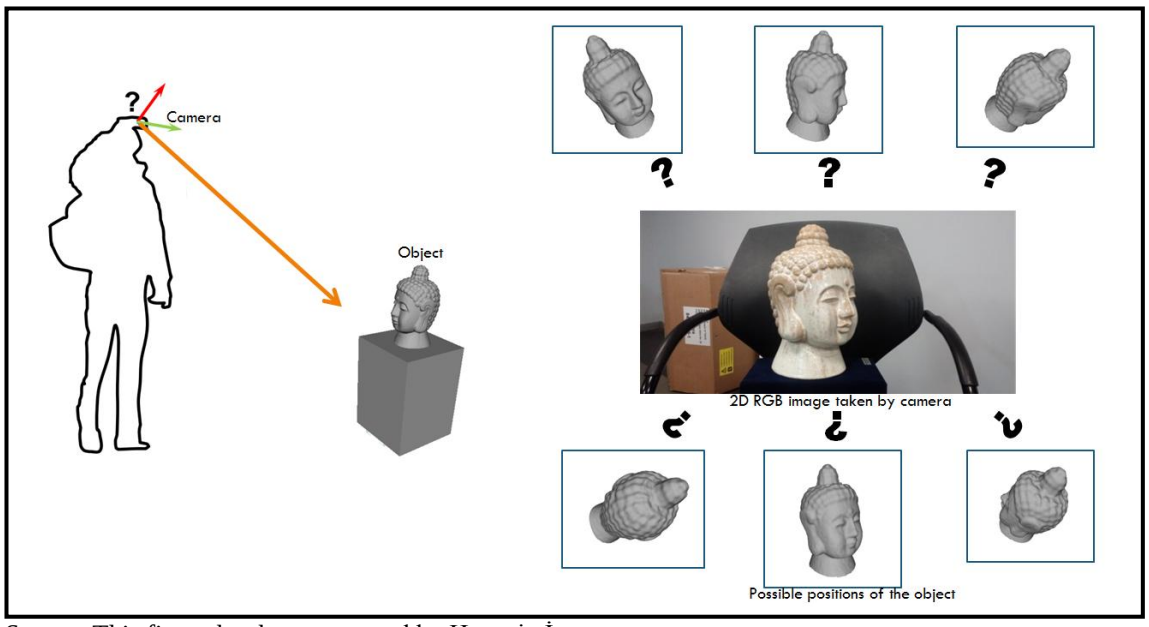
Estimating the pose of the object means that estimating the camera location and orientation relative to the scene or object's position and orientation relative to the camera. Even if there are a lot of pose estimation techniques suggested and some methods have very precise results, they do not work in every circumstance. Every method has some weakness for determining the pose of the object accurately for example algorithms with high precision may fail if another object occludes the object, which is a very typical case in the real world.

Occlusion problem occurs when the interested object is partly behind of other objects. Since object features cannot be retrieved completely, methods fail to recognize and estimate the pose of the object. Occlusion problem can be seen in Figure 1.2, from which state of the art methods have problems to estimate pose. In addition to occlusion, scale factor is also another consideration that influences the performance of the pose estimation. The interested object is not always in the center of the image with favorable scale. It may be too big or too small in the input picture. For that reason, pose estimation algorithms also should be robust for varying scales of the object in the query image.

The solution that suggested by this thesis aims to enhance current state of the art methods to finding poses of the 3D objects from 2D RGB images by using 3D model of the object. We aim to find the location and rotation angles of the object not just from good initial given pose, but also from far initializations. In addition to this, our other

purpose is developing a scale invariant method. That is to say, finding poses of the objects even on occasions when the object is too close to camera or so far from camera. Moreover, another important component of our method is estimating pose of the object by using only 3D model that is constructed by Kinect sensor, which has very poor details when compared with CAD, or laser scanned models.

Figure 1.1: Model-based pose estimation



Source: This figure has been prepared by Hüseyin İnan.

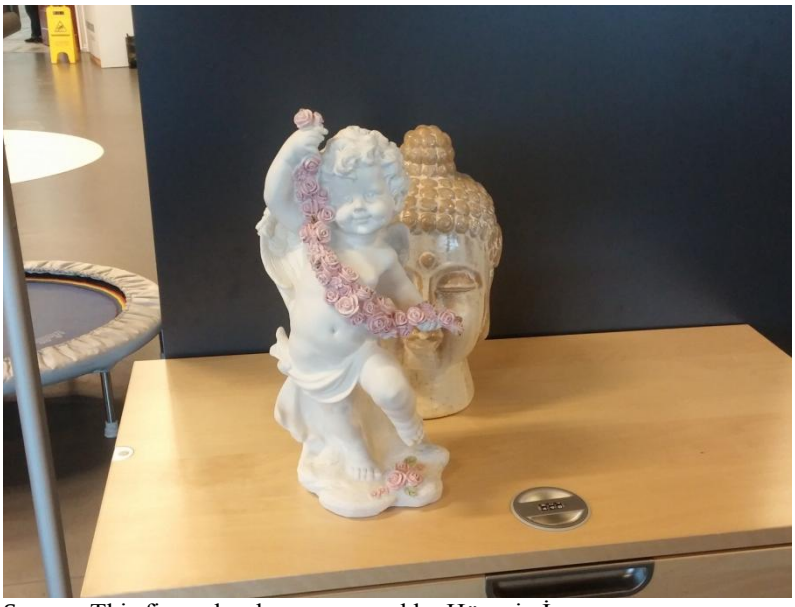
1.1 OVERVIEW OF 3D POSE ESTIMATION

In our method, three-dimensional pose estimation is based on data that is extracted from the input image of the scene or relevant features got from them. We try to handle to difficulties of scale variances of the object in the image and stretch out limits of good initialization. In addition to 2D image, some complementary methodologies using other scene understanding tools for pose estimation. For instance, pose estimation methods by stereo cameras intent to get useful domain segments that represent the object or object parts by using stereo image properties like motion attitude and three-dimensional data.

In general, 3D rigid object's pose estimation process is accomplished by projecting a group of 3D points of the model into the input image. Rotation and translation values are estimated in such manners that minimizing the chosen error measurement after the correspondences are established. Correspondingly, 3D rigid object pose estimation is equivalent to the issue of deciding the extrinsic parameters of the camera. Thus, setting

robust corresponding points between the triangulated model of the object and the 2D input image is an all-important issue for pose estimation.

Figure 1.2: Occlusion



Source: This figure has been prepared by Hüseyin İnan.

1.2 MODEL-BASED POSE ESTIMATION

Model-based pose estimation of rigid objects means to determine the orientation and position of an object respect to a definite viewer by using model of the object. To determine pose, the six degrees of freedom parameters of the object is estimated which are three translation and three rotation parameters. For non-rigid objects (articulated objects), pose estimation may refers much complicated optimization problem.

Generally, 3D data of the interested object is utilized to make pose estimation process simpler. In model-based pose estimation methods, 3D information is generally obtained from CAD models of the object. However, they can be also obtained from laser scanners or cheap depth sensors such as KINECT as we use. Addition to these, 3D mesh also can be formed by taking several pictures of the object.

Pose estimation methods which using 3D data are named as model-based pose estimation methods. In Figure 1.1, the overview of the model-based pose estimation is illustrated. A camera takes a photograph of an interested object (Buda). In addition to RGB camera, stereo cameras or depth sensors like Kinect can be utilized. There are

bunches of possible pose values of the object. The true pose is tried to be estimated, according to taken a 2D RGB image and a 3D model of the object.

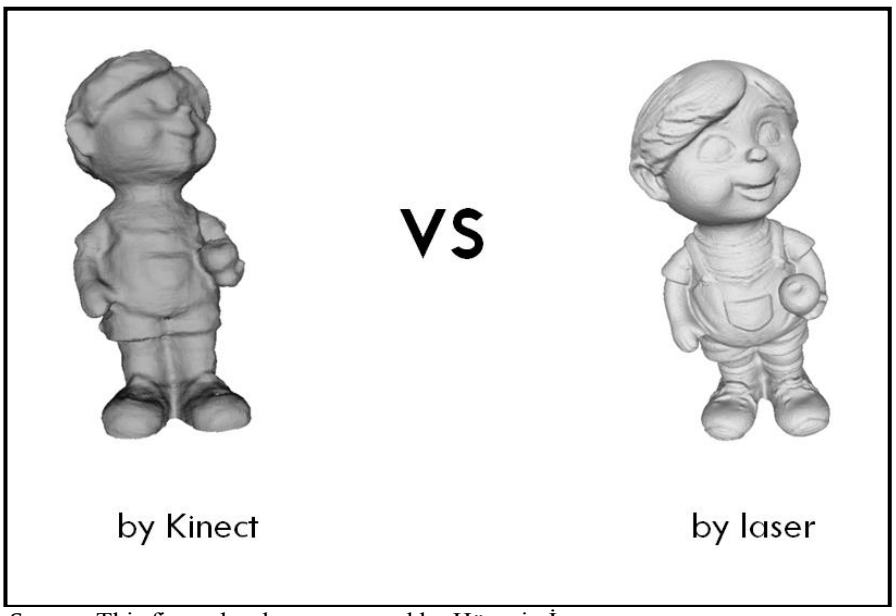
1.3 PROBLEMS FOR POSE ESTIMATION

As remarked before, there are some difficulties for estimating the pose of the object even for state of the art methods. These problems can be listed as; initialization, occlusions, cluttered scenes and scale invariance.

Firstly, most of the pose estimation algorithms depend on a coarsely estimated initial pose. For the precise pose estimation, good initialization is very significant. Otherwise, algorithms could return us wrong estimated position and orientation of the object. In our method, we try to estimate pose of the object even with far initializations.

Secondly, some pose estimation methods assume that there is only interested object in the scene, and the background color of the object will be different from the object with a uniform color. However, in the real world these premises very likely to fail. Even more, there will be similar objects in the scene that prevent to identify the interested objects from each other. Additionally, objects may not appear as whole. Some objects may occlude them partially.

Figure 1.3: Object modelled by using Kinect (left) and by laser scanner (right)



Source: This figure has been prepared by Hüseyin İnan.

In addition to the problems that are listed above, there are also some specific challenges for our pose estimation system. Firstly, we use a 2D RGB image by retrieving the scene. This means that we are deprived of depth information of the scene unlike other using devices like depth sensors or stereo cameras. This makes us using only 2D data of the object like edges, colors etc.

Secondly, 3D modeling procedure is another problem that we have to deal. Models that constructed by using Kinect is used as prior information for pose estimating. Currently, Kinect manages to produce the 3D model of an object, but they are not precise as laser scanned objects or modeled objects by using a program like CAD models. As shown in Figure 1.3, there are a great deal of difference between models that scanned by using Kinect and laser in terms of details. For instance, if the faces of the two models are compared, it is obvious that a lot of specification of the object get lost in Kinect model. We can barely distinguish eyes, mouth, ears etc. of the model. Furthermore, there are some distortions in the object like nose and hair of the boy, when Kinect constructs the model of the object.

Finally, limited prior data of the object is also one challenge for our method. As mentioned before, we have only 3D Kinect model of the object that is a coarse model of the object when compared with laser and CAD models. In addition, there is no texture of the model, so we cannot use the color information of the RGB image. Adding to these, some suggested pose estimation methods use additional data like templates of the object, real images that pose of the object is known, or doing preprocessing for teaching their algorithms.

1.4 PROPOSED METHOD

There are two premises are assumed hold by iterative pose estimation method:

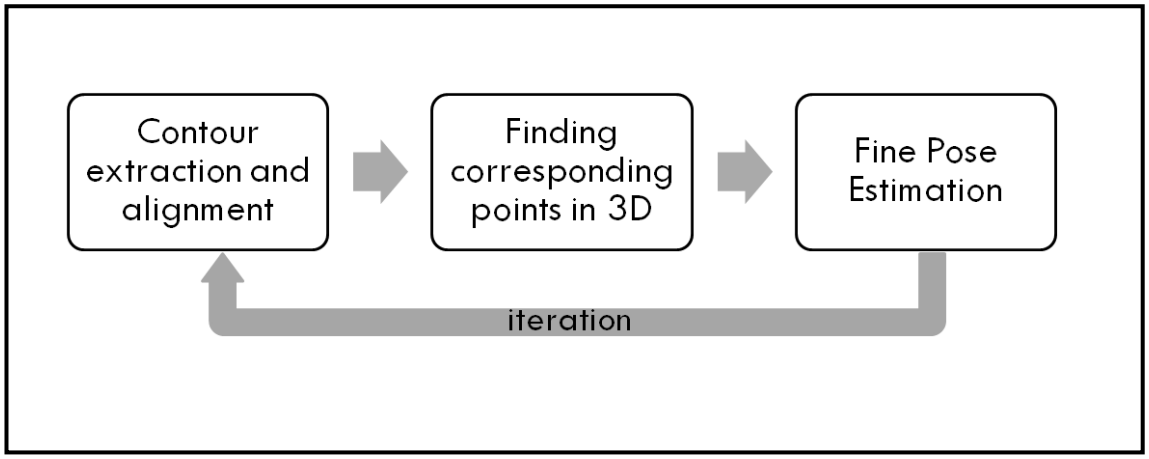
1. The object's 3D pose and its outer contour is in one-to-one correspondence. This means, for a particular contour, only one determinate corresponding pose of the object exists.

2. When the object's pose varies continuously, its corresponding contour on the image plane also changes continuously.

Premise-1 ensures the object's outer contour is a feasible feature for pose estimation, and premise-2 legitimates that iterative changes of the pose are reasonable. Premise-1 does not hold for artificial objects that are self-symmetric in certain views. For instance,

it is possible to contours of the top view and bottom view is same of an object. Therefore, it is hard to identify each other by using only a contour. However, it is mathematically rare situation, so pose estimation algorithm based on boundary can be still applied to these objects.

Figure 1.4: 3D pose estimation pipeline overview



Source: This figure has been prepared by Hüseyin İnan.

The remainder of this thesis is organized as follows. In section 2, we do literature survey and give details of state of the art methods about model-based pose estimation from RGB images. In section 3, we describe camera geometry that we used to map 3D models into 2D images. In section 4, we describe steps of our proposed method those are contour extraction and alignment, finding corresponding points in 3D and fine pose estimation. These steps are also illustrated in Figure 1.4. In section 5, we do the analysis to test our proposed method performance by using three objects. In section 6, we conclude the thesis by give remarks of our algorithm and mention about further researches.

2. LITERATURE SURVEY

In this section, we mention pose estimation methods that are suggested until today. Moreover, in the last part we explain four state of the art methods that are most closet algorithms to ours. Many methods have been suggested to solve 2D-3D pose estimation problem based on 3D model. These methods can be grouped according to the device they used for acquiring scene.

2.1 POSE ESTIMATION BY USING DEPTH SENSORS

Firstly, some of the suggested methods for pose estimation use RGB-D images. Generally, they use depth sensors to capture the scene. Since they have depth data of the scene addition to RGB, their feature space much more broad when it compared with others. For instance, W. Woo et al. (2011) combine both RGB and depth images information to utilize from both the geometry and the texture of a query object. A group of pre-processed templates by using both depth and RGB data are utilized to identify the query object. The depth image's 3D points are aligned with the reference templates after the object is detected. By using depth data addition to texture, they make the method more robust to lighting changes. A. Aldoma et al. (2011) also suggested an approach to extracting an accurate and fast 3D feature for estimating the pose of objects. They evolved the Clustered Viewpoint Feature Histogram (CVFH) from Viewpoint Feature Histogram, which can be efficiently computed. They combine the cameras roll histogram with CVFH, to handle partly occlusions and noise. Additionally, H.I. Christensen et al. (2012) also suggested a method by using depth sensors. They use pixel-wise information as a feature. To express object surfaces, oriented point pair feature is used. They adopted a voting scheme to aim for getting groups of possible 3D transformations between query scenes and object model. Oriented point pair features contribute this voting scheme. They learn interested object by scanning it several times from different views. By using color information, helping to enhance the performance of pose estimation from the point of time and accuracy. The color point pair feature is used to utilizing from color information. This leads to more efficient pose estimation, by taking advantage of it in voting scheme. Furthermore, E. Marchand et al. (2012) suggested two methods based on depth-assisted rectification, which are called Depth-Assisted Rectification of Contours for obtaining pose of the object from RGB-D

images. Extracted features of RGB image are transformed into canonical view. Depth data are employed in order to get invariant to scale, rotation, perspective distortions representation. While one method focus on textured planar or non-planar objects and another one is used for texture-less planar objects. Moreover, C. Choi et al. (2012) proposed an edge-based approach for pose estimation. An efficient matching algorithm is used to detect textureless objects. Hence, it provides some estimated coarse potential poses by matching between input image and object's edge templates. Based on matching costs, by using the coarse pose hypotheses randomly particles are initialized. The annealing process is employed for making certain that the initialized particles are not trapped at a local minimum. The refinement process is applied to enhance matching corresponding points between the edges of query image and edges of projected model. J. Xiao et al. (2012) also suggested an approach for pose estimation based on edge information. They extract features and estimate pose in real-time from RGB-D data. By using RGB and depth image, they find a set of edge cues. Then by using iterative closest point algorithm in 3D space, they align the down-sampled point clouds with feature points. Additionally, N. Navab et al. (2012) also proposed an approach for finding the poses of the objects based on LINEMOD approach (S. Hinterstoisser, 2011) for depth sensors. Multi-modal templates in the Kinect output can be detected efficiently by using LINEMOD. Possible appearance of interested objects is sampled by using LINEMOD templates for detection. Templates are made by using densely gradients of sampled image and normals of the depth map. When the object is detected in the query image, it also find the pose of the object. Furthermore, S. Behnke et al. (2013) manage to detect classes of object and estimate the pose of the detected objects by using depth sensors. They adopted Hough forests to group pixels and vote for 3D object orientation and position. By using dense depth, scale-invariant detection and pose estimation of the target object is fulfilled efficiently. They train their data by using turn-table setup which renders arbitrary scenes. Finally, D. Kraft et al. (2013) suggested a method that extract edges by using both RGB and depth data. They adopted 3D line segments method from N. Pugeault, 2010 which using stereo cameras. 2D line pieces are back-projected to build up 3D line pieces. Then intersect these line segments with 3D point cloud to estimate the pose of the object.

2.2 POSE ESTIMATION BY USING STEREO CAMERAS

Secondly, stereo cameras are other devices for estimating the pose of the objects. These cameras have two cameras since they acquire two images, and when the distances between cameras are known. The depth of the scene can be extracted. Hence, these methods also use depth information to enhance their results. For example, J. Gall et al. (2006) proposed a method to estimating the pose of textured models. In addition to comparison 3D model with segmented input image contour, they also use reliable features in textures to establishing correspondences. They use stereo cameras for retrieving scene. Additionally, W. Garage et al. (2010) also suggested method to finding pose of the objects by using stereo cameras. They named their descriptor Viewpoint Feature Histogram (VFH) that evolved from Fast Point Feature Histogram (M. Beetz, 2009) descriptors to classify 3D objects. By the help of descriptor, they encode important statistics between the surface normals on the mesh and the viewpoint. To classifying the object and the view, fast approximate K-Nearest Neighbors is used. Lastly, J. Xiao et al. (2013) suggested pose estimation method from a stereo camera. They integrate both dense motion and stereo cues by using corresponding sparse key points and also in the feature extraction step, model information is fed back from the model. This lead to method more accurate and it is not fragile to occlusions and noise. They use iterative closest point (ICP) algorithm (P. Besl, 1992) to minimize the pose error by using all the dense features together.

2.3 POSE ESTIMATION BY USING 2D RGB IMAGE

All of the methods that listed above use depth data for pose estimation. We will discuss the methods use only RGB information of the scene. There are earlier researches like RAPID (Harris 1992) is one of the edge-based algorithms that suggested on 1992. The algorithm mainly uses the sampled edges of a 3D object and high contrasted regions. There are also some optimized implementations of the algorithm (Weidemann, Ulrich 2008). There are also some methods that use edges explicitly for estimating the pose. In (Roller, Negal 1993), straight-line edge parts are detected and model's edges mapped to the image according to pose estimation. The algorithm tries to optimize the pose by using Mahalanobis distance of the edge parts.. Daniel DeMenthon developed pose estimation method PosIt (DeMenthon, Davis). This fast method uses orthographic

projection rather than perspective projection. Then, Coplanar PosIt is published. It is version PosIt that also works with planar point clouds. Moreover, both 3D model and images with known poses in method (Liebelt, Schmid). Both of them generalize to object class and capable to estimate pose, but they do not work real time and fragile to occlusions and clutter environment.

For instance, G. Sommer et al. (2004) suggested an approach for estimating the pose of 3D rigid objects. They represent the contour of images as a twist that is identical to a Fourier representation, which make possible to applying low-pass approximation of the object model to refine the estimation pose. Additionally, J. Weickert et al. (2005) proposed an edge-based method for pose estimation. They represent an energy function, which consists of both parameters of pose and the boundary of the image as unknown. They try to minimize the energy function by adjusting both segmentation of the image and pose parameters iteratively. Furthermore, N. Navab et al. (2010) also proposed a method by using edge properties of the object to estimation pose. They employ dominant gradient orientations that makes possible to check only the partial group of all possible locations of pixels when searching image. Additionally, 3D Model is represented by using a few template sets. They use branch-and-bound method to parse efficiently the query image. Moreover, R. Basri et al. (2011) proposed a method that utilizing a nonparametric voting procedure and discriminative re-scoring together to estimate the pose of 3D objects. The class model is constructed by uniting object's 3D shapes. For finding detection pose candidates, a non-parametric voting procedure contributes as an attention mechanism. To detect the class of the object pose, each image cue can vote. After, these potential poses are fed to SVM classifiers to credit a score to improve estimation pose. Finally, J. Piater et al. (2011) suggested a probabilistic method, based on edges. They do not establish correspondences between the input image and 3D model, instead the object model and 2D edges of the image are defined as probabilistic distributions of visual features.

2.4 THE STATE OF THE ART METHODS

Presently, there are four state of the art methods that are similar with our method, which means they estimate the pose of the object from RGB image. Firstly, R. Vidal et al. (2013) suggested a pose estimation method from single 2D image. They model 3D object, which form from edge segments by using 2D object blueprints. The model,

which they constructed, includes 3D edge direction, two normal vectors. Edge vectors will be mapped into the 2D image and then corresponding points will be established with regard to intensity discontinuities for verifying object pose. While the self-occlusion and primitive visibility will be determined by surface normals with regard to the pose of the object according to its pose. The model is projected into the image and trying to minimize matching cost between projected contour and the input image, according to their extracted HOG features. They modify HOG features to emphasize foreground objects and to suppress cluttered and flat regions. After the corresponding problem is solved, they introduce branch-and-band (BB) optimization scheme to find the true pose that maximize energy function. By using BB algorithm, firstly they disintegrate the search space. Then bounds on the objective are computed over each grouped element. These subgroups are sorted in the priority queue with regard to their evaluated bounds. Until the finest resolution is reached at convergence, the search space is divided into parts.

Secondly, A. Torralba et al. (2014) suggested a pose estimation method from a single image. For prior data, they use CAD models and real training images which model's pose is known. By using these data, they learn model's geometric and appearance information. They introduce FPM, fine pose part-based model that integrate geometric data and appearance information. Additionally, they also introduce 3D part sharing concept. They formed geometric data by using shared 3D parts. They train their model by assigning importance score to sharing vertices to measure the visibility and distinctiveness of the vertices. 3D shared parts are trained by rendering and real images. Then they improve DPM by using 3D shared parts. By the help of DPM with shared parts, the pose of the object is estimated accurately, additionally the score of object likelihood lead to enhance an unlabeled group of images to detect object locations in the image. They using sliding window approach for fine pose estimation to get all high scoring probable poses. Non-maximal suppression is applied to pose space to get estimation results. Moreover, for computing sliding window conveniently, sparse coding is applied on the weights of the root and part models as identified by R. Girshick et al. (2012) that makes possible to look for fine pose space in a feasible of time.

Thirdly, S. Todorovic et al. (2011) also proposed for finding poses of objects by using contour features of the image. They train the algorithm by using real images of the

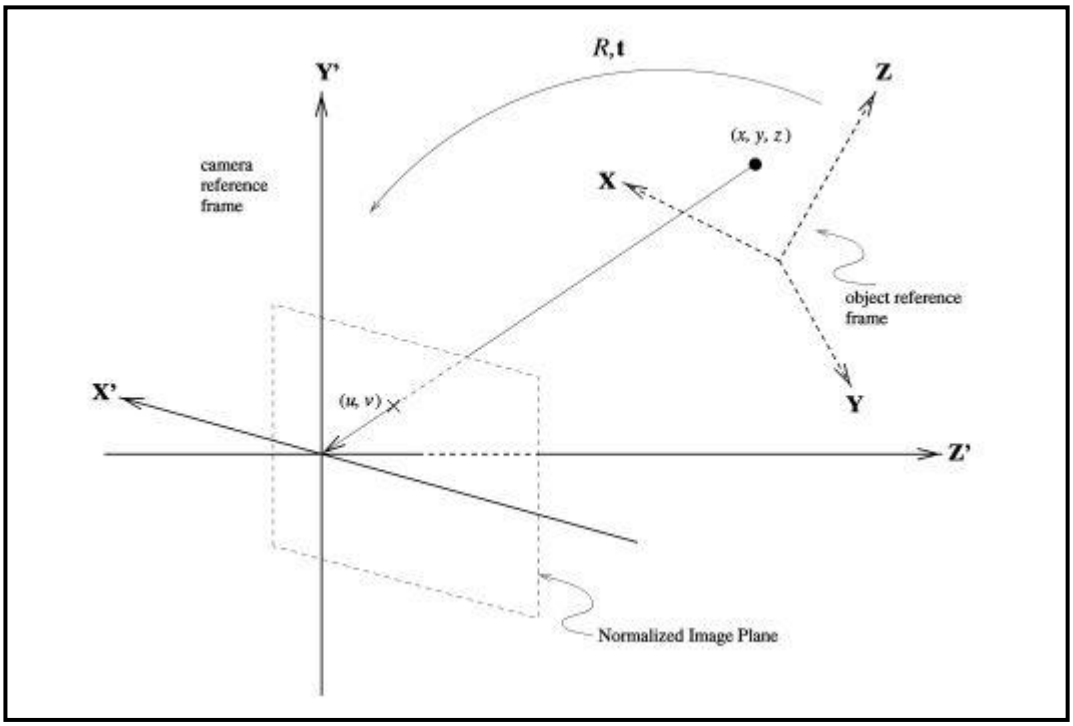
object taken from different views. By using SVM method, object poses in each training images are estimated. Moreover, they use view-dependent shape templates to learn the model. Then, objects boundaries are copied to a template and taken the average of them. They built a probabilistic shape map by counting each pixel in the template as average frequency. After templates are initialized, they match shapes of the query image and learned shape templates to detect a subset of the object in the image. The templates of bag of boundaries features match only with object contours in the image. The introduced mid-level features BOBs those are summaries of boundaries. They are used to identify object contours amongst all other edges of the scene. By using successive matching iterations, object boundaries are determined according to BOBs. They do boundary detection and pose estimation simultaneously. Finally, best matching shape template is depicted as the pose of the object.

Lastly, K. Daniilidis et al. (2014) proposed a method to estimate the pose of rigid objects from a single color image. It is the closest research with us, because they also use Kinect scanned 3D models as prior and use only contour data of the query image. They use gPb method to extract boundaries of the object in query image, which is introduced by P. Arbelaez in 2011. gPb is state of the art algorithm that suppresses texture and clutter edges and finds probable contours of the object. Then extracted contours are used as input to trained part-based object detector. They train deformable part models (DPM) with the set of silhouettes of close poses as positive exemplars and then the possible object locations are hypothesized at test time. This method detects the object and also at the same time estimate pose of it. Several hypotheses are produced as the output of the DPM method. They over-segment hypothesis region into super-pixels. They used chordiogram, a shape-based descriptor that is introduced by A. Toshev (2012) to match super pixels with query boundary. The help of chordiogram descriptor represents geometric relationship's distribution represented between matched boundary edges, termed chord. Closest chordiogram distance with query silhouette is chosen as a coarse pose. Then fine pose estimation process is performed by applying motion field equation to align model projection with query image silhouette.

3. CAMERA GEOMETRY

In this Section, the geometry of a camera is going to be depicted and mathematical problem description is going to be introduced. Internal camera calibration is a process of obtaining the parameters like image center's position in the image or focal length etc. These parameters are specific to cameras and affect imaging process. Additionally, the external camera calibration is the process of retrieving parameters that stand for orientation and location of the camera reference frame relative to a known world or object reference frame. These camera parameters hold a very significant position for a pose estimation task. Because by getting external camera parameters, 3D pose of the camera is also obtained. Thus, actually we look for external parameters of the camera. Furthermore, internal camera parameters should be known beforehand to get external camera parameters. Therefore, we assume that internal camera parameters were obtained already before the pose estimation process. Because cameras use perspective projection geometry to mapping 3D objects to 2D view to acquire the input image, we use perspective camera model to project 3D model into the 2D image.

Figure 3.1: Object space to camera space transformation



Source: Lu, C.P.,Hager, G.D.,Mjolsness, E.: (2000) *Fast and Globally Convergent Pose Estimation From Video Images*

Perspective camera model is going to be described in the following section. A basic camera calibration algorithm is going to be analyzed in section 3.2. For more detailed information, textbooks such as (Hartley, Zisserman 2004), (Faugearas 1993),(Harralick 1992) can be used.

3.1 PERSPECTIVE CAMERA MODEL

By using the camera, 3D world is projected into the 2D image. Pinhole camera model is a standout amongst the most well-known geometric camera models that used in computer vision.

The geometry of pinhole camera model is depicted in Figure 3.1. The projection center of the camera is picked as the origin of the camera reference frame (CRF) that is a Euclidean coordinate system. The optical axis points in the positive z direction. By using the pinhole camera model, a 3D world point P with coordinates $(X, Y, Z)^T$ is projected onto 2D image point $(u, v)^T$ plane where a ray intersecting from pinhole to view plane. Correlation between image point and object point is (3.1)

$$u = f\frac{X}{Z}, \qquad v = f\frac{Y}{Z} \tag{3.1}$$

By using this equation, object reference frame is projected onto the image coordinates. The parameter f denotes the focal length of the camera. Normalized image plane with z=f=1 in the camera reference system is assumed if f is known.

CRS is also in a different space, which is called world coordinate system (WCS) which is also Euclidean coordinate frame. The relation between given object with the world coordinates $P_w = (X_w, Y_w, Z_w)$ and camera coordinates $P_c = (X_c, Y_c, Z_c)$ is rigid transformation.

$$P_c = R(P_w - T) \tag{3.2}$$

Where R is $[r_1, r_2, r_3]^t$ is the 3 x 3 rotation matrix describing the orientation of the camera in the WCS and t is (t_x, t_y, t_z) a translation vector.

Model point, P_c is mapped onto normalized image plane by the equation (3.3) and $p_i = [u_i, v_i, 1]$ is the perspective projection of P_c.

$$p_i = \frac{RP_w + t}{r_3 P_w + t_z} \tag{3.3}$$

Equation (3.3) is called as collinearity equation and means that projection center of the camera, P_w and P_c are collinear.

The orthogonal projection P_c on the projection ray of P_w is equal to P_c itself is also an approach to idea of collinearity.

$$RP_w + t = F_i(RP_w + t) \tag{3.4}$$

F_i stands for projection operator (Ping Lu, Hager 2000) and formulation is (3.5)

$$F_i = \frac{p_i p_i^t}{p_i^t p_i} = \frac{1}{\|p_i\|^2} \begin{pmatrix} u_i^2 & u_i v_i & u_i \\ u_i v_i & v_i^2 & v_i \\ u_i & v_i & 1 \end{pmatrix} \tag{3.5}$$

Equation (3.1) is called image space collinearity equation and (3.4) is called object space collinearity. Rigid transformation parameters of the camera, which denotes orientation, and the location of the camera in WCS are named external parameters of the camera.

Estimating the pose of the object with regard to 2D image by using a 3D model of the object is finding the closest rigid transformation, rotation matrix and translation vector. To checking best-fit transformation, the error between 3D and 2D objects are tried to be optimized by using image or object space collinearity methods.

Alternatively, a vector with six variables can be used to describe the pose, v= (t_x, t_y, t_z, α, β, γ). First three symbols are translation values and the others stand for rotation parameters around the axis. The principal axis is the line through camera center and perpendicular to the image plane. The junction of the principal axis with the principal plane is named the principal point. Earlier, it is assumed that the origin of the coordinates in principle plane and principal points are in same point. However, in practice it may not occur. In this situations, a 2D translation is necessary to translate into coordinates of the principal point C=(C_x, C_y).

Besides, it is accepted that in horizontal and vertical directions, the size of pixels are both equal. But, the pixels sizes in axis directions might not be same in images are obtained by using CCD cameras. The scale ratio of horizontal and vertical axis also

should be evaluated. Assume m_x and m_y are the quantity of pixels per unit distance in image coordinates. At that point, by multiplying equation 3.3 with the calibration matrix, equation 3.6, pixel coordinates are received from the world coordinates with translation.

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \textbf{(3.6)}$$

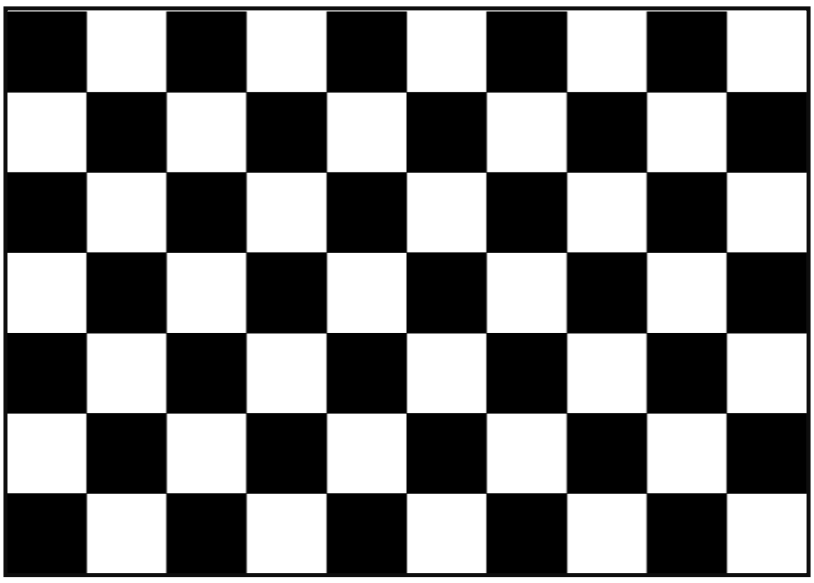
With f_x is equals to f multiplication with m_x and f_y stands for multiplication f with m_y. They are focal length of x, y axis. Symbol s is the skew parameter. Equation 3.3 can be rewritten in more brief form, world point P = (X, Y, Z, 1) in homogeneous representation; image point p= (u, v, 1) in homogeneous representation.

$$P = MP \quad \textbf{(3.7)}$$

Where M is the camera projection matrix

$$M = K[R|t] \quad \textbf{(3.8)}$$

Figure 3.2: Calibration grid to used in camera calibration



Source: graphics.stanford.edu

3.2 CAMERA CALIBRATION

As mentioned before, before starting pose estimation process, we should firstly obtain camera intrinsic parameters. Computing the view projection matrix is the issue of

camera calibration. Chessboard patterned calibration grid, which contains 7X10 squares is used to calibrate intrinsic parameters of the camera. Example of a calibration grid can be seen in Figure 3.2. The grid is mounted onto a flat plane. Then approximately eight photos are taken of the calibration grid from different views. With the image pre-processing of this images, corners of the squares are found and camera matrix (M) is determined that can be splitted into intrinsic and extrinsic matrices.

3.2.1 The Camera Matrix Estimation

Camera matrix M maps 3D points P_i in the world space onto 2D image points p_i. Camera matrix is used to transform 3D points to 2D by $p_i = MP_i$ equation for every i. Various cost functions can be minimized for finding camera matrix. Distance between projected points and corresponding 2D points in the image is one formula that can be used as a cost function. By using the formula, minimizing the sum of the distances between all corresponding points helps to find the camera matrix. The formula of the cost function is $\sum_i d(p_i, MP_i)^2$, where d stands for Euclidean length between 2D image point p_i and projected 3D point MP_i. Calculated errors are assumed Gaussian the result to

$$\min M \ \sum_i d(p_i, MP_i)^2 \qquad \textbf{(3.9)}$$

has the highest probability to be camera matrix. Iterative methods like Levenberg-Marquardt can be used to minimize of the non-linear error function. However, for iterative methods end up with true camera matrix, good initialization is also required.

Another solution for minimizing distances between corresponding points is a linear method that is the reformed formula of 3.7 as cross product vectors that is as $p_i \times MP_i = 0$. By using this linear solution, finding the minimal model view is easier. The formula is

$$p_i \times MP_i = \begin{pmatrix} v_i m^{3T} P_i - m^{2T} P_i \\ m^{1T} P_i - u_i m^{3T} P_i \\ u_i m^{2T} P_i - v_i m^{1T} P_i \end{pmatrix} \qquad \textbf{(3.10)}$$

Where the j^{th} row of the matrix M is represented as m^{jT}. 11 equations are needed to find camera matrix, M, because M has 12 parameters and 11 degrees of freedom is ignored. Three linearly dependent equations are occurred by using each corresponding points. Choosing the first two yields

$$\begin{pmatrix} O^T & -P_i^T & v_i P_i^T \\ P_i^T & O^T & -u_i P_i^T \end{pmatrix} \begin{pmatrix} m^1 \\ m^2 \\ m^3 \end{pmatrix} = 0 \qquad \textbf{(3.11)}$$

For each point correspondences, the equation 3.11 is lined row by row to get 2n×12 matrix A is for a group of n corresponding points. Linear equation structure Am=0 is taken by this process, the m parameter has 12 parameters of the camera projection matrix. Reducing the residual ||Am|| to minimum can be used to get camera projection matrix M. ||Am|| is algebraic that normalization constraint is applied.

A solution is obtained from the unit singular vector of a corresponding to the smallest singular value. In order to avoid numerical instability, it is important to carry out some sort of data normalization. The points in the image can be normalized appropriately by a translation. Thus, their centroids are at the origin and scaled so that their RMS distance from the origin is $\sqrt{2}$ "(Harralick 1992).

Figure 3.1 illustrates the transformation from object space to camera space by using parameters R and t. Object's 3D coordinates (x, y, z) transform into image pixels (u, v).

3.2.2 The Camera Matrix Decomposition

For projecting 3D points into a 2D image, camera matrix M is adequate. However, the information it holds is compact. That is to say, location and orientation of the camera, internal parameters of the camera are unachievable directly. After camera projection matrix is determined, it can be spitted into extrinsic and intrinsic matrices that form like formula 3.8. Extrinsic camera parameters stand for location and orientation of the camera. By using camera matrix's SVD decomposition that proposed by Richard Hartley, the center of the camera t can be obtained.

3.3 THE FUNDAMENTAL MATRİX

The fundamental matrix F is a 3x3 matrix of rank 2, which leads to admitting only seven independent parameters as stated by O. Faugeras (2001). If a point in 3D is

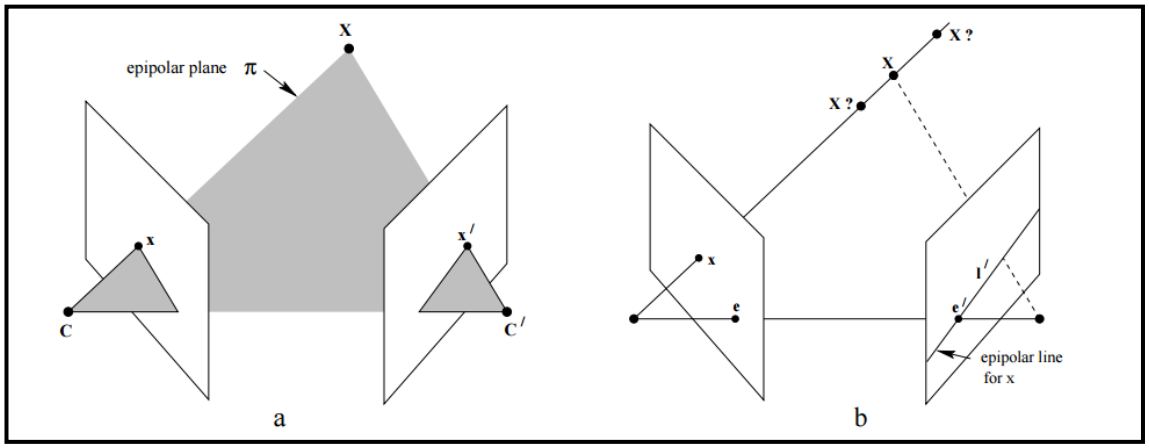
corresponds to a' in one image and a" in the second image. Then the for these corresponding points fundamental matrix satisfy the condition

$$a'Fa'' = 0 \quad \textbf{(3.12)}$$

F actually gives rotation and translation between one camera to another camera where,

$$F = [t]_x R \quad \textbf{(3.13)}$$

Figure 3.3: Epipolar geometry and the fundamental matrix



Source: Multiple View Geometry in Computet Vision, R. Hartley and A. Zisserman

As shown in Figure 3.3, there is an epipolar line l' in each image that correspond to a point x in other image. The point x' should be on the epipolar line to be corresponding point of x. The epipolar line is obtained by projecting ray from the point x through the first camera's origin C in the second view. Therefore, a point in one image corresponds to an epipolar line in other image. This projection performed by fundamental matrix.

4. POSE ESTIMATION METHOD

4.1 ABSTRACT OF THE ALGORITHM

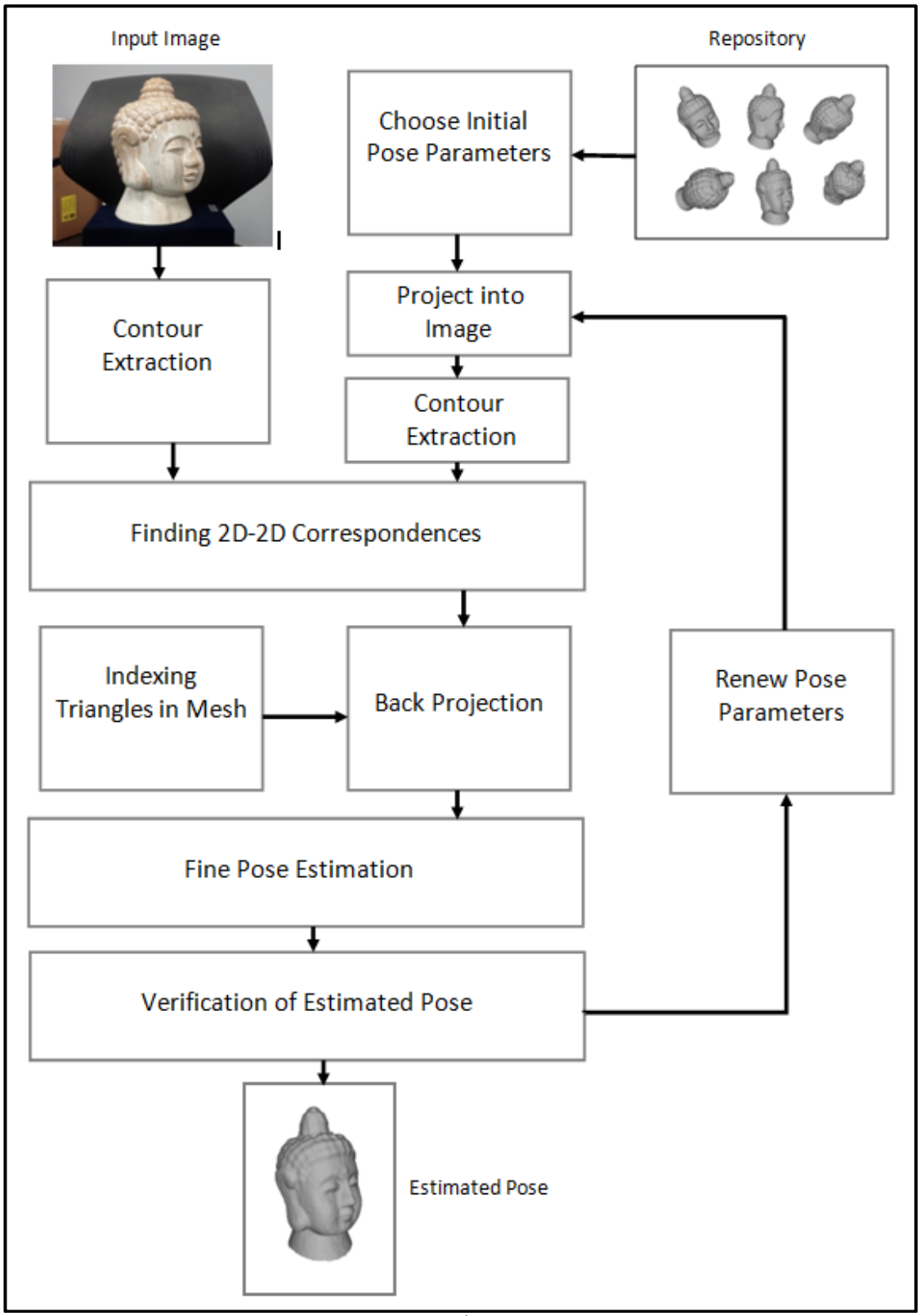
Summary of our pose estimation algorithm can be seen as a flowchart diagram in Figure 4.1. We firstly find contours of the input image by using active contours method that is mentioned in 4.2.1. According to the chosen rotation matrix R and translation vector T, we project 3D object over the input image. Projected mesh contour is also found by using a basic edge detection algorithm. After two interested boundaries are extracted, 2D-2D correspondences are established by using Ant Colonization Optimization method that will be mentioned in section 4.3.2. The next step is transforming 2D-2D correspondences to 2D-3D. To do that, we initialize 3D mesh of object. According to the column number of the matrix, we index each triangle in the mesh and encode them with RGB, as mentioned in 4.4.1. Afterwards, by the guide of indexes, we set up 2D-3D corresponding points. For the starting point initial pose parameters should be given. Since we try to extend the initialization scope, we manually choose pose parameters coarsely to test our method in the first iteration. After 2D-3D point correspondence established, pose of the object is estimated by using PnP algorithm with RANSAC. The details of the fine pose estimation algorithm can be found in section 4.6.1. The steps mentioned above iterated by predefined number. The pose with smallest value that is calculated according to criteria that is mentioned in section 4.6 is chosen as a true pose of the algorithm.

The pose estimation method that we used in the thesis is got inspired from Pose Estimation of 3D Rigid Object Based on Tentative Point Correspondences that is proposed by D. Leng and W.Sun. Their proposed method only works with good initial pose parameters. I try to enhance their method to more robust to bad initializations. Figure 4.2 is example for input image, and Figure 4.6 is illustration for the 3D mesh.

4.2 INITIALIZE QUERY IMAGE AND PROJECTED CONTOURS

Since suggested pose estimation method is contour based, the contour of the query image and projected silhouettes contours should be extracted precisely. That means continuous, clean and single-pixel wide contour is necessary. For that reason, active contours method is used to find the contour of the object in query image.

Figure 4.1: Flow diagram of the pose estimation method



Source: This Figure has been prepared by Hüseyin İnan

Addition to query image contour, model's contour of projected silhouette also should be found. However, contrary to the input image's background, the projected image is not clustered. Therefore, simple image processing operations such as canny edge detection algorithm can be used to extract contours of projected image.

There are several reasons for using only contour as feature extraction process from the input image and 3D model. First one is our 3D model is constructed by using Kinect and it does not covered with texture. Hence, it leads us to eliminate algorithms, which need RGB information or grayscale, such as SIFT, SURF etc. Another cause for using only boundary information is lack of details on models that are obtained by Kinect. As mentioned before laser-scanned models have much more detail, if they were used edge information of projected models also can be used. However, edge data of Kinect-scanned models probably lead us to false matching between projected model and query image.

Figure 4.2: Input image with Buda object

Source: This figure has been prepared by Hüseyin İnan

4.2.1 Active Contours Method

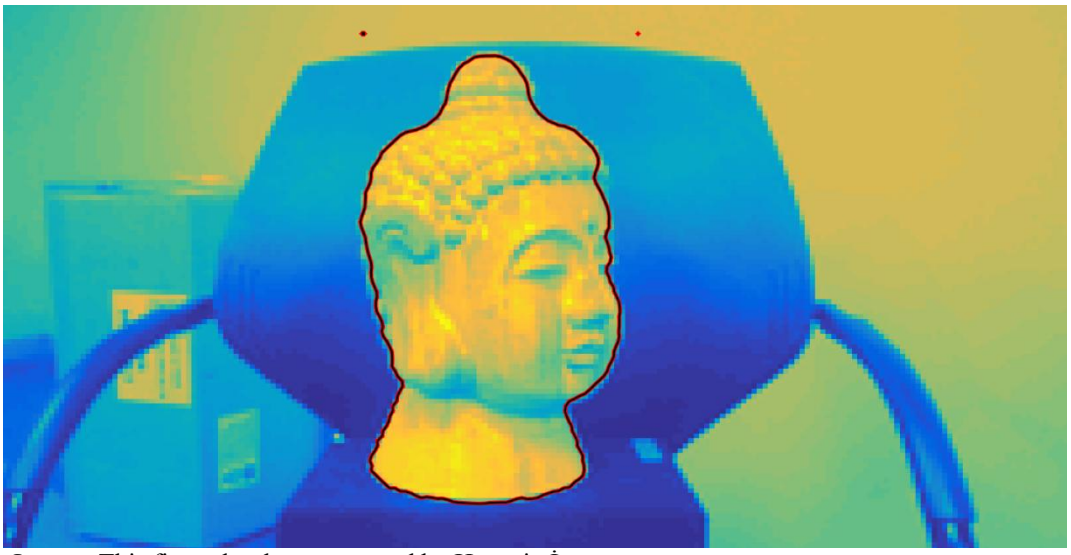
Extract object contour from the cluttered scene is one of the fundamental problems of computer vision. The scene that illustrated in Figure 4.2, there is a chair and a box in the background of the image. The basic methods like canny edge detection extracts all edges that belongs also other objects in the image, so it fails to finding the interested object boundary. Other more complex methods should be employed for detecting

contour, we use active contours model to detect interest of boundary. D. Terzopoulos introduce it in 1988.

As mentioned before, basic edge detection methods fail to get the contour of the object. There are also other methods for extracting the boundary of the images as gPb. However, we prefer to apply active contours method for our algorithm because of relatively simple implementation.

For analyzing scenes automatically, image segmentation holds great role for the accuracy of the algorithm to retrieve the object from the surroundings. One of the most based segmentation method is edge detection that identifies sharp transitions in the image brightness. By using edge detection, the contour of the object is detected. Another method for retrieving objects from the scene is region growing. Region grows by seeding pixel and adds neighbor pixels according to some pre-defined features such as color, texture or intensity of the pixel. The help of this method segments different featured areas in the image segmented out from each other. Moreover, eventually, the algorithm creates edges between these regions.

Figure 4.3: Edge detection with active contour model



Source: This figure has been prepared by Hüseyin İnan

Addition to edge based and region growing methods, active contours or snake method is also one of the most used methods for object segmentation. Active contours method detects the boundary by minimizing energy spline that is proposed by M. Kass et al. Active contours are exposed under two kinds of forces; one of them is internal constraint forces and another one is external image forces such as lines and edges.

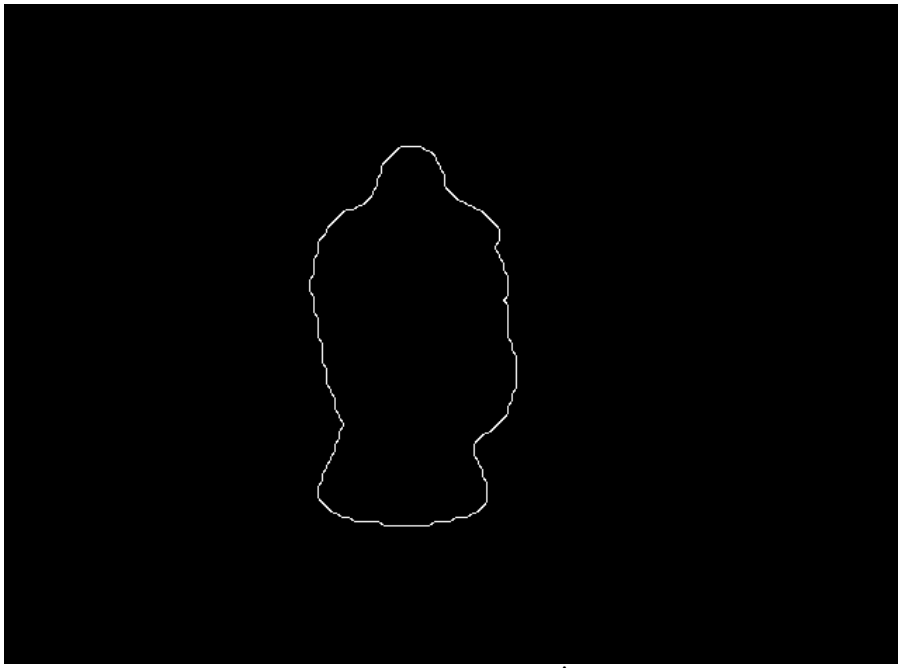
Contour pixels are represented as c = {p_1, p_2 ...pN}, and p_i is column and row number of contours i^{th} pixel. Energy function that active contours method tries to minimize is:

$$E(v) = \sum_{i=0}^{NM-1} (E_{int}(p_i) + E_{ext}(p_i)) \quad \textbf{(4.1)}$$

This minimization function composed of two energy elements. One of them is internal energy and other one is external energy. The first one charges with properties of boundary such as discontinuity and curvature. Additionally external energy is characterized by image's gradients at an active contour's point.

In this thesis, greedy algorithm is used to minimize this energy function. Evaluation of each p_i's local neighbor leads us to choose the best neighbor to move p_i. After energy function is run for each adjacent pixel, the point with minimum energy is chosen for transferring p_i. The process continues until predefined iteration count or contours get balanced.

Figure 4.4: Boundary of the input image



Source: This figure has been prepared by Hüseyin İnan

Each of the energies relevant to the snake is used with various differences. The total energy of the algorithm can be prescribed as the distance between points through the active contour. Hence, active contours' points are maintained as close as possible

together. The energy of curvature is the second derivative at each pixel of the active contour. Moreover, divide each energy into largest number among the adjacent pixels to normalize them.

External energy is opposite gradient magnitude. After all, for computing the gradients of the image performing preliminary processing on the image is presupposed. This preliminary processing is applying Gaussian blurring on the image. The normalizing process helps to wipe off weak edges or noise edges.

Initialization is the one of the most important steps of the active contours method to segment object out to scene and get contours of the object. Failures may occur, if the initialized active contour is improper, the output of the algorithm will be also wrong segmented. Therefore, first active contour should cover the object within the area of it.

In our algorithm, we take an input image just like Figure 4.2. After than by applying active contour method image segmented as foreground and background of the object as illustrated in Figure 4.3. After segmentation process, the contour of the object is extracted like in Figure 4.4.

4.3 2D-2D POINT CORRESPONDENCES

4.3.1 Feature Extraction

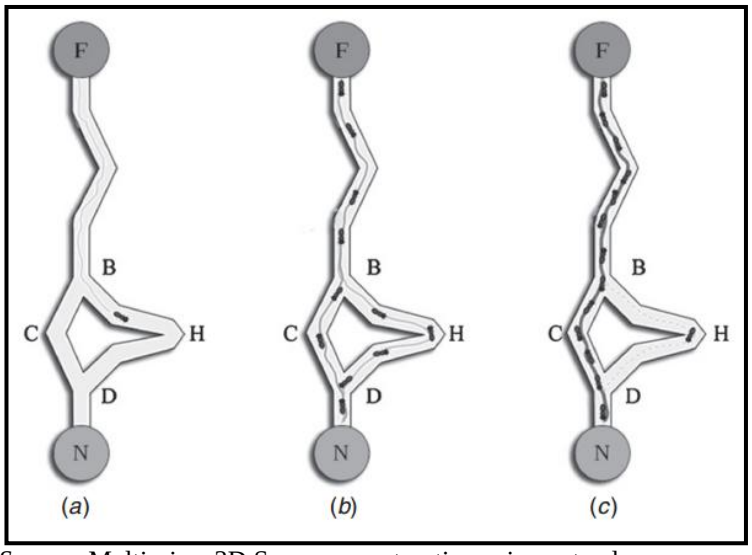
For finding corresponding points between the query image and prior 3D mesh, first step is extracting some features of contours. Then extracted features are matched to estimate the pose of the object in the image. In our method, we use shape of context as a feature descriptor. Shape of context measures similarity of shapes on returns matching points. We need to find 2D correspondences between projected model and the query image because we will transform these matching into 2D-3D correspondences. A simple 2D-2D shape matching is not adequate for our method, because they only answer one question that if the shapes are similar or not. However, we also need corresponding points. We adopt Ant Colony Optimization algorithm to find correspondences. Since it uses the shape of context as feature metric, and shape of context is scale invariant. It makes also our algorithm scale invariant.

4.3.2 Contour Correspondence by Using Ant Colony Optimization

Finding corresponding points between two contours is a challenging problem. It should be invariant to orientation, transition and scale to be used for pose estimation. In addition to invariance, the method should find correct correspondences despite some

noise or deformations on the contour, because there is a high probability to extract deformed contour. Hence, for the accurate pose estimation results, our 2D-2D matching algorithm should not be fragile to such noises and has specialty of invariance to scale change.

4.5: Ant colony optimization



Source: Multi-view 3D Scene reconstruction using ant colony optimization techniques.

We use Contour Correspondence via Ant Colony Optimization that proposed by P. Wighton, as a source of finding 2D-2D point correspondences between projected object boundary and query object contour. Their method generally works well and manages to find true corresponding points. Point corresponding problem formulated by using Quadratic Assignment Problem (QAP) that proposed by Stutzle. QAP is an optimization problem that cannot be figured out. Despite this, by using ant colonization optimization method as a heuristic, we can get close to the solution.

For Solving NP-hard optimization problems, Ant colony optimization is one of the algorithms that is used. As can be understood by the name of the algorithm, it inspires from Ants behaviors. If an ant examined by itself, It has basic abilities. On the other, when we look at an ant colony, we see that the group of ants can solve complicated tasks. Their qualification seems especially while they looking around for food and getting contact with each other. For finding foods, ants start to go around from their nets aimlessly. Their random food search ends, when they find interest piece to carry to the nest. While they return to the nest, they spread out a chemical material that excites other ants to inform that there is a food source to carry. By following this chemical scent,

other ants also reach to source ant start to trace between the food source and nest. The communication between ants in the colony is illustrated in Figure 4.5.

For obtaining the optimum solution for point correspondence, the same methodology also works. The ACO algorithm adopts same instinct to finding optimized solution. Firstly, the problem is transformed into graph model, and then artificial ants start to try to get the optimal solution. The solution of the problem is found by ants who traversing between vertices of the graph. Applicable solutions are each ant's trace. In this method, it is computed by an objective function. For finding the best solution, the algorithm controls the density of chemicals that traversing ants spreading. This chemical again is used for communication.

Connected bipartite graphs are used to form 2D-2D matching problem. Bipartite graph's components are points of each contour. While an ant goes from when vertex to another, paths emerge.

Heuristic data and chemical emission of the edge from ants determine which edge the ant will choose. Proximity and the shape descriptor provide heuristic information for that decision. After the ACO algorithm iterate just once, it means that pre-determined amount of ants traverse the graph. By the end of the first iteration, some quantity of chemicals also accumulated in edges. By the help of chemical data, ants abstain from local minima.

By the time the ACO algorithm is analyzed, it is seen that artificial ants look into the whole search space. This leads to the algorithm to return various solutions. Even though, the number of chemical materials is increased in edges that are preferred by the objective algorithm. Accordingly, ants start to traverse only particular edges. Heuristic information is necessary for starting point. That is to say if predecessor information is sufficient, ants choose right trace to traverse the graph. Additionally, Chemical materials also necessary for restricting search space and direct the algorithm to the optimal solution.

As mentioned before, the ACO algorithm tries optimizing an objective function. J and I are two groups of points that represent the contour of the object. Finding corresponding points between J and I means to try to minimize the objective function. π^* is the value that is tried to be minimized.

$$\pi^* = argmin_{\pi}\big(OBJ(\pi, M, N)\big) \quad \textbf{(4.2)}$$

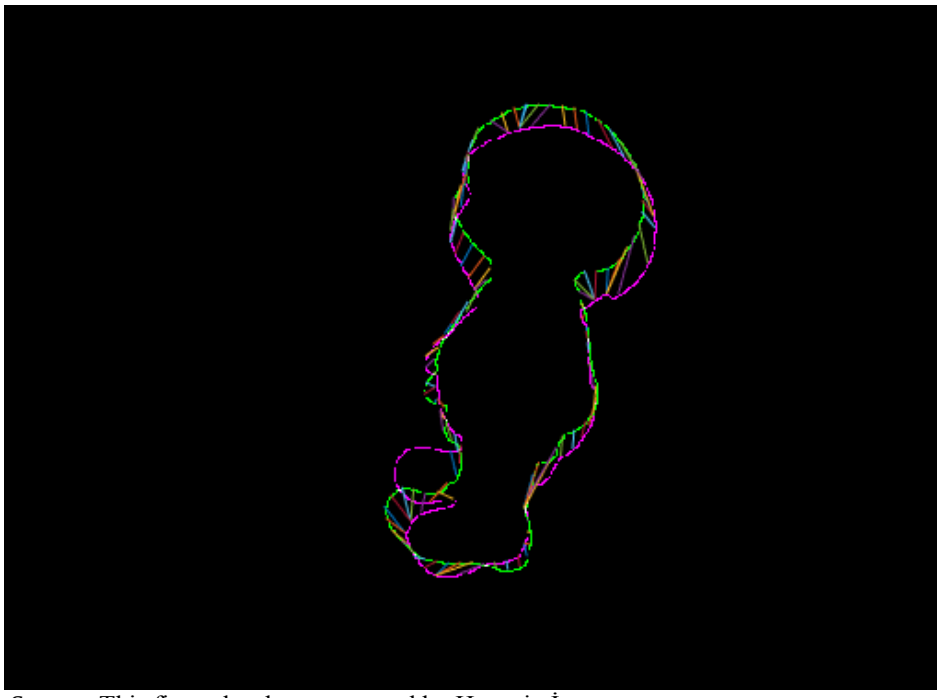
Where OBJ stands for objective function. It evaluates the matching π in correlation to the shapes that outlined by I and J, and π is a mapping such that $\forall m \in M, \exists n \in N: \pi(i) = j$.

A group of features should be extracted to matching shapes. For this method, shape of context is used as shape descriptor. On the other hand, shape descriptors are not sufficient alone for finding true correspondences. Additionally, by using order preserving, matching two irrelevant positioned points can be avoided. Moreover, together with shape descriptor and order preserving, proximity is another element for finding good correspondences. In detail, if two points are matched, then their closest neighbors also should be matched.

Despite, finding the global solution is not certain by using the ACO method. It is proved that using the ACO for QAP is one of the most successful methods. Puzzling out correspondence problem can be seen as solving QAP.

In Figure 4.5, point correspondences of projected contour and input image contour is shown, as seen in the example the algorithm is achieved sufficient success for pose estimation.

Figure 4.6: 2D-2D image correspondences that found by the ACO



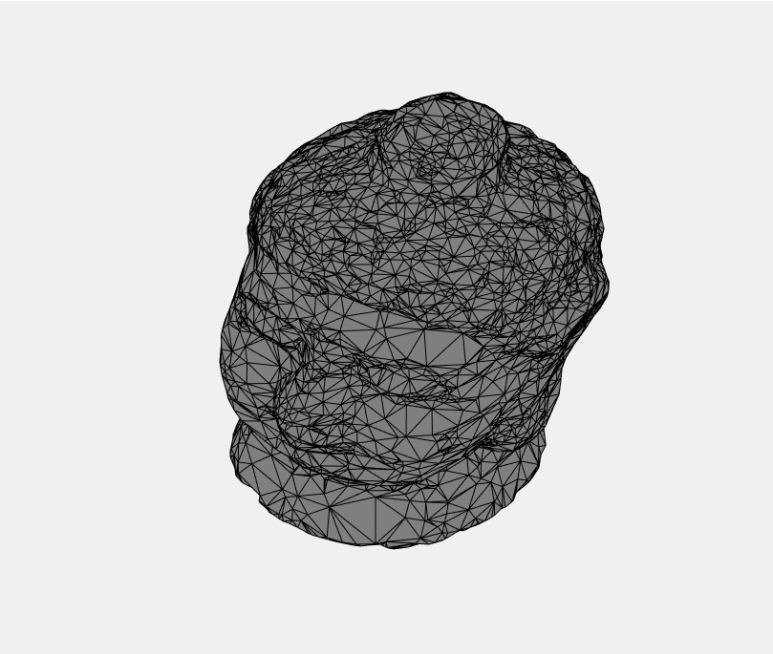
Source: This figure has been prepared by Hüseyin İnan

In a given input image (purple contour), feet of the boy are separated from each other, however for projected image (green contour), they are united. Thus, the ACO method is matched with only one foot, this is the one allowable mistake of the algorithm. Other than other parts, correspondences seem established fine to enhance pose of the object for next iteration.

4.4 DETERMINING 2D-3D POINT CORRESPONDENCE

Finding corresponding points between 2D image and 3D mesh is one of the crucial problem of pose estimation. In the previous section, we achieve to find correspondences between 2D-2D contours. Now these correspondences should be transformed into 2D-3D corresponding points for estimating the point based pose estimation. We have correspondences information between projected contour and image contour. If we can back-project projected contour pixels onto 3D mesh object point. 2D-3D correspondence problem will be solved. For this purpose, we use the method that indexes meshes by color RGB values and project object into the image by color with regard to RGB values of the mesh.

Figure 4.7: 3D mesh of Buda object

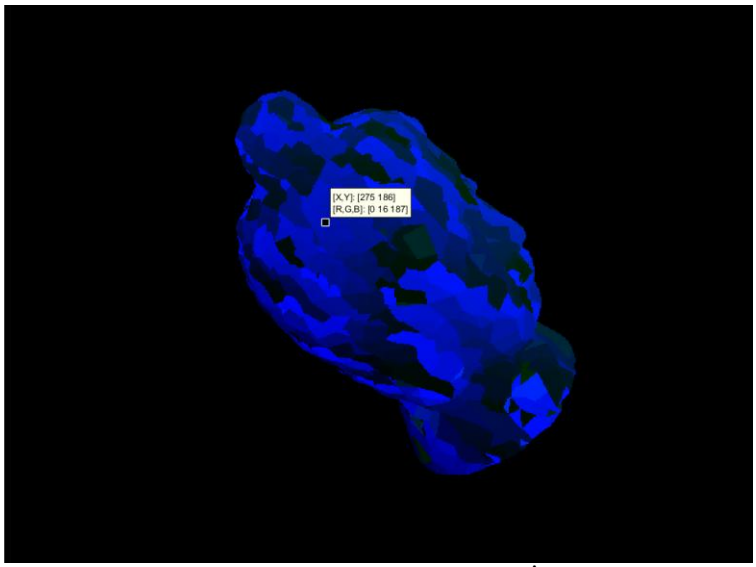


Source: This figure has been prepared by Hüseyin İnan

4.4.1 Assigning Colors to Mesh Indexes

For doing back-projection from 2D contour pixels to 3D mesh object points, we index 3D model's meshes with RGB color values. In RGB images, every pixel has three integer values between 0 and 255. The RGB values represent respectively red, green and blue color of the pixel. We can use this pixel property to give identification to each triangle in the mesh.

Figure 4.8: Indexed and RGB encoded projected 3D mesh



Source: This figure has been prepared by Hüseyin İnan

As illustrated in Figure 4.7, 3D model of the object is composed of triangles. Here we firstly index each triangle of the mesh according to alignment in the file or matrix that stored and then encode these indexes. If our mesh has N triangle, then we index all meshes with M=[1,2,3,...i,...,n]. We encode each mesh by using property of mathematic modules. We encode the triangle line number of the triangle to the variables RGB by taking modules 255 of the number. We firstly divide the number i to 255 the result is FD (first division). Then the blue variable is the number minus FD multiplication with 255. Then we divide FD to 255 again and the result is SD (second division). Then, the green component is FD minus SD multiplication with 255. In addition, the red component is second division. By using this encoding procedure, we can identify $255^3 + 255^2 + 254$ triangle, which equals to 16646654.

After coloring each triangle, we can easily achieve back-projection process by doing back calculation above procedure. For the given RGB values. 255^2 multiplication with R addition with 255 multiplications with G and addition with B is the result of the index of mesh. After back-projection process, we have the associated triangle with corresponding 2D point. Since triangle has three vertices. We should get one 3D point by using these three vertices. There is no rotation and translation transform between camera coordinate and object frame is assumed. Three vertices of the founded triangle is $X_v=\{x_{v1}, x_{v2}, x_{v3}\}$, this triangle define a plane which is:

$$P = \begin{pmatrix} (x_{v1} - x_{v3}) \times (x_{v2} - x_{v3}) \\ -x_{v3}.(x_{v1} \times x_{v2}) \end{pmatrix} \quad \textbf{(4.8)}$$

If x_g is 3D coordinates of the contour point, then coordinates of the corresponding point is:

$$x_{vg} = \left(-P(4)/x_g \cdot P(1:3)\right) x_g \quad \textbf{(4.9)}$$

As shown in the Figure 4.9, image pixel with the coordinates (x, y) →(275 186) is mapped from triangle $(16\times255 + 187)^{th}$ indexed triangle, which is 4267^{th} triangle of the 3D mesh.

4.5 FINE POSE ESTIMATION

After assigned candidate 2D-3D point correspondence relationship between the query image and object's 3D mesh by using back projection as mentioned above, we can start to estimate the pose of the object. To establish the point-based pose estimation sub-procedure, PnP algorithm with enhanced by RANSAC is used. Since we do not have precise matches between 2D image and 3D model, we use RANSAC to eliminate outlier correspondences. We use the OpenCV implementation of the algorithm.

4.5.1 Perspective-N-Points Pose Estimation with RANSAC

For the finding pose of the object by using 2D-3D point correspondences, we use PnP algorithm implemented by RANSAC. PnP is an algorithm that finds poses of objects with n correspondences. However, if there are many noises in the input correspondences, it ends up with badly estimated pose parameters. RANSAC is used for eliminating noises in the input. Henceforth, only sieved input correspondences used in PnP algorithm and the output estimations are much more precise than estimation with raw input data. For informing the algorithm PnP, Evaluating Pose Estimation Methods for Stereo Visual Odometry on Robots that proposed by M. B. Dias is used as base document. Additionally, Overview of the RANSAC Algorithm that is enrolled by K. G. Derpanis is base for information about RANSAC.

PnP problem is estimating rotation matrix and translation vector of the camera by given N 3D points X that are determined in coordinate system A and their 2D corresponding points x with respect to A. Camera that is used in pose estimation is assumed calibrated. Before getting into the PnP problem, analyzing P3P problem is much better for understanding. The solution for P3P problem is solving the depths on each ray. After finding the depth values, absolute orientation stage is used to get camera parameters. By the using method that proposed by M. Nölle, the unknown depth of point i from the camera center in the camera frame is $S_i = \|X_i - C\|$ and known inter-point distance obtained from stereo in the world frame is $d_{ij} = \left\|X_i - X_j\right\|$. $r_i = K^{-1}x_i$ is the viewing ray for each points. R_i is than can be used for calculating the angle between each pair of rays: $cos\theta_{ij} = r_i^T r_j / (\|r_i\|\left\|r_j\right\|)$. By using cosine rule, series of polynomials can be formed:

$$^{2}_{ij} = S_i^2 + S_j^2 - 2S_iS_j\left(cos\theta_{ij}\right), for\ i,j\ \in \{(1,2),(1,3),(2,3)\} \quad \textbf{(4.11)}$$

For the finding the unknown depths, this polynomial system of equations can be solved in different ways. By eliminating each of the other depths from Formula 4.11, one of the unknown depths such as S_j is solved in the original P3P algorithm. This leading to an 8^{th}-degree polynomial consisting of all powers of S_i, as depth must be strictly positive, leads to a quadratic polynomial that has up to four solutions.

$$'(u) = \alpha_4u^4 + \alpha_3u^3 + \alpha_2u^3 + \alpha_1u + \alpha_0 \quad \textbf{(4.12)}$$

An excellent review of a wealth of nonlinear algorithms that have been developed to addressing this problem is provided by Nölle. This approach called original P3P.

We used EPnP in our method; it is an alternative non-linear solution with a linear running time that was proposed by P. Fua. It is showed that more efficient suggesting its use in real-time systems.

PnP algorithm is used with RANSAC. The RANdom Sample Consensus algorithm that proposed by R.C. Bolles, is used in pose estimation methods for dealing with large number of outlier in correspondences. RANSAC is a resampling method that estimates the parameters of the base model by generating applicant solutions by using the minimum number observations required to estimate the underlying model parameters. RANSAC, unlike to other techniques it uses the smallest set possible and carries on to extend this set with consistent data points that proposed by R.C. Bolles.

Steps of RANSAC algorithm:

1. Choose a random minimum numbered subset of points for finding out the parameters of model.
2. Fit the model by using parameters found.
3. Count the how many inliers from the set of all points approximately fit the model.
4. Repeat the steps predefined time.
5. The model with the largest set of inliers is selected.

The iteration number, N, is selected high guarantee that at least one of the sets of random samples does not contain outlier with probability p. P is usually 0,99. If u is the probability that any chosen point is an inlier, than 1-u is the probability of occurrence an outlier.

$$1 - p = (1 - u^m)^N \tag{4.13}$$

Where N is the iteration number, m is the minimum number of points that required. and with some reformulation,

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)} \tag{4.13}$$

4.6 ITERATIVE METHOD

Although, with the fine pose estimation step, the closer pose of the object is estimated, generally only one iteration is not adequate to obtain the real pose of the object. Therefore, we run our algorithm iteratively to get satisfied results. Each output of fine pose estimation step is used as input of our method. After the algorithm is iterated with given number N, five best estimations are chosen. Then the estimations are averaged according to the assigned weights.

There are two criteria to pick closest poses. First, one is checking areas of silhouettes. Bitwise AND operation is applied on the binary image extracted from the query image and the binary projection image projected from object's 3D mesh.

$$A_{ratio} = \frac{area(BI_{ob} \cdot BI_{pr})}{area(BI_{pr})} \tag{4.10}$$

Where BI_{ob} stands for the extracted binary image and BI_{pr} stands for projected binary image. To finalize pose estimation iteration stage, A_{ratio} should be close to one. We use a threshold value to if our ratio number is close to one, end up the iteration. If the similarity coefficient that is calculated by formula 4.10 is above the set threshold, we label pose parameters as a potential pose of the object. Unfortunately, this area comparison is not very efficient to choose true estimation. Hence, we combine it with second criteria, which is sum of the distances of 2D correspondences. We choose five closest smallest sums of distances as potential poses of the object. These poses are weighted with A_{ratio} and then averaged to get the ultimate pose.

As mentioned above pose estimations depend on good initialization. Therefore, our method also needs an initial rough location and orientation parameters. To achieve this initialization task, methods based on templates can be used. These methods compare the

projection silhouette between pre-computed template silhouettes that pose are known. Closest silhouette according to using metric is chosen as the input of iterative algorithm. For instance, shape of context is one of the metric that can be used for choosing initial pose parameters. Currently, we initialize pose of the object manually. Virtual cameras are put around the object by degree of freedom 10 degrees. Totally, we have 613 images each of them corresponds to the different pose of the object.

5. EXPERIMENTS AND RESULTS

Our method is applied to rigid objects to find their poses. We tested our method by using three objects; Buda head, a boy and an angel statue from different angles. 2D color images of the objects are shown in Figure 5.1. Objects' meshes are constructed by Kinect as illustrated in Figure 5.2. These meshes are obtained by the algorithm that introduced by Bahtiyar Kaba in 2014. When the output model is compared with laser-scanned models, their accuracy is 5 millimeters for 90 per cent. Our analysis can be grouped into five groups. First three test cases assume query object and projected objects are in same scale. In addition, the other two tests that obtained by using boy statue, test our pose estimation algorithm for different scales. The first one finds the pose of the object that input object is bigger scale than projected object, and other one is vice-versa, mapped mesh is bigger than query object.

Figure 5.1: Point clouds of buda, boy and angel that are obtained by Kinect

Source: This figure has been prepared by Hüseyin İnan

For analyzing accuracy of our method, the ground truth of pose of the object is needed. Thus, instead of finding poses of the real world objects, we test our algorithm on synthesize objects. For the checking our algorithm to robustness the initially given pose parameters closeness to real pose parameters, objects poses are tried to be estimated from different views. Additionally, initial pose estimation's rotation difference is chosen in a range of 10 degrees to 80 degrees. Moreover, discrepancy is different in multiple axes.

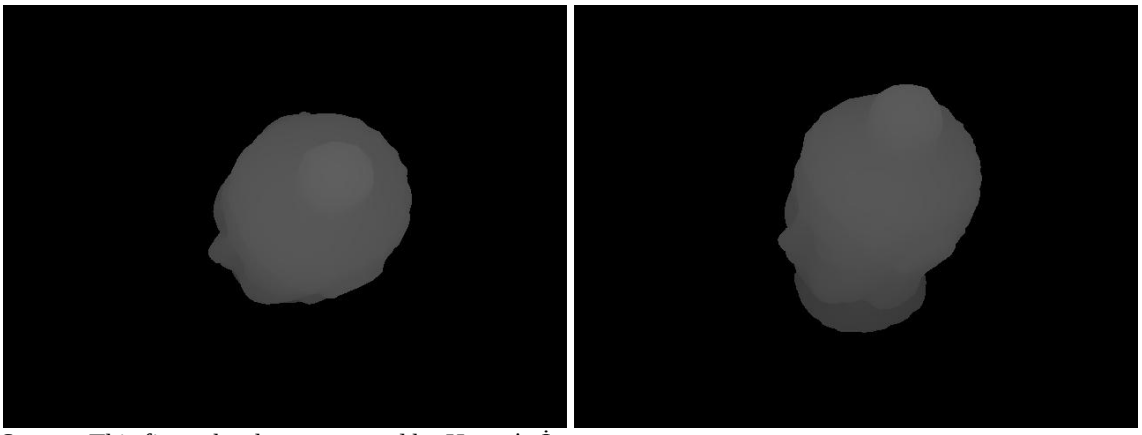
While testing the algorithm, it is assumed that there is no occlusion or cluttered background. Therefore, the contour of the object is extracted precisely.

For Buda head statue, 15 poses from different views are tried to be estimated. Eleven of them are successful to estimate the real pose of the object. However, to finding limits of our method, in addition to close starting points, we also try to estimate the real pose of the object by starting far from initial estimation. Our algorithm is iterated between 6 and 17 to find the true pose of the object. In just one case, pose is estimated in 31^{th} iteration.

Figure 5.2: 2D RGB Images of buda, boy and angel

Source: This figure has been prepared by Hüseyin İnan

Figure 5.3: Initial estimate pose (left) and real pose (right) of the analyze 4

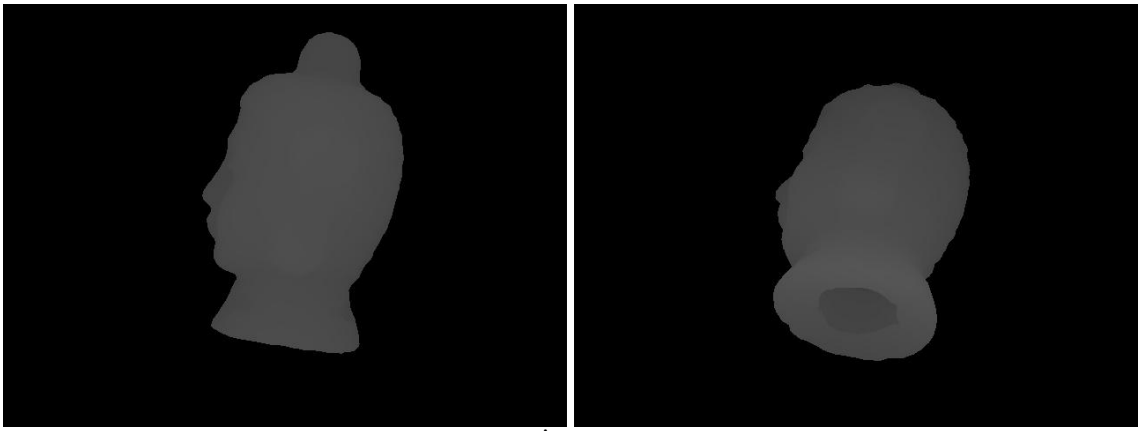


Source: This figure has been prepared by Hüseyin İnan

Firstly, Figure 5.3 illustrates close guessed pose estimation. These poses are different 20 degrees in x-axis. The error of the estimated pose can be found in Table 5.1. The error in x axis is 0,1 degree, 0,2 degree in y axis and 0,7 in z axis. It is pretty close to real pose of the object. The vice versa of the poses also estimated in analyze 5.

For poses that illustrated in Figure 5.4 are used for two analyze. Difference between poses are 40 degrees in axis x. In first one, pose of the object's in left image is tried to be estimated by using pose of right one. The error is 1.9, 0.3 and 0.2 respectively for axis x, y and z. However, while estimating right ones pose initializing by right one, our method fail to estimate pose.

Figure 5.4: Initial and Real Poses of analyses 6 and 7

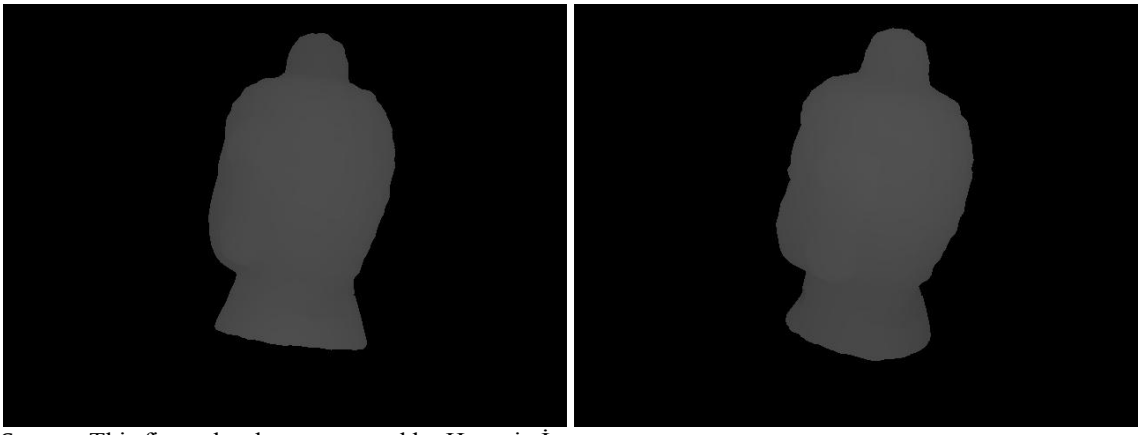


Source: This figure has been prepared by Hüseyin İnan

In Figure 5.5, two varied poses of the Buda is illustrated. Result of the pose estimation test can be found in Table5.1. Differences between poses are 20, 0 and 20 respectively for axis x, y and z. Despite the degree differences are not so far, our method works only

one way it manage to find pose from right image in Figure 5.5 of left image but the other way around the algorithm fails to find true pose of the object. It end up in local minimum and in Figure 5.7 right image is contours of input image (purple) and contour of projected image (green) is the situation that estimate is end it up. Although, it seems that pose of the object is estimated truthfully, it is far away from the true solution. This is the one of the drawbacks of the pose estimation from only contour.

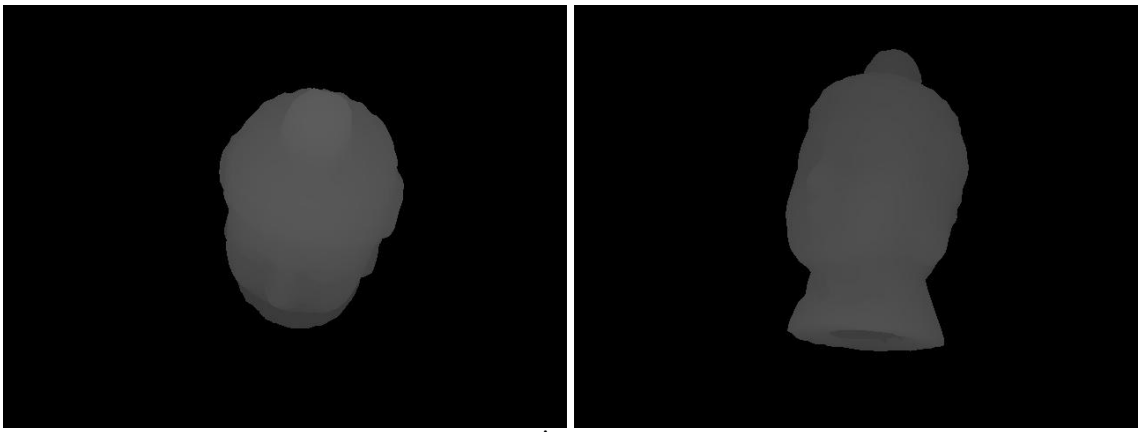
Figure 5.5: Initial and Real Poses of analyzes 10 and 11



Source: This figure has been prepared by Hüseyin İnan

Furthermore, Figure 5.6 illustrates poses that our method fails to estimate. However, for this case it is hard to find true pose from just contours because difference is huge between these poses of the Buda. It is 80, 0, 20 respectively for axis x, y and z.

Figure 5.6: Given pose (left) and real pose of the object for analyze 13 and 14

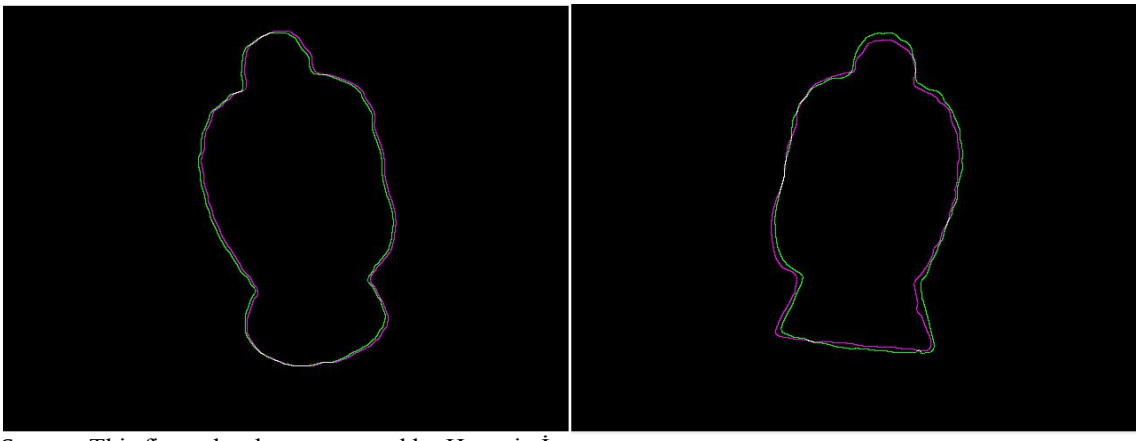


Source: This figure has been prepared by Hüseyin İnan

In Figure 5.7, left image true pose estimation contours of the two poses that analyzed in analyze 15. As seen in the figure, contours are close. In addition, pose estimation errors are 2.2, 0, 0.2 respectively for axis x, y and z.

In addition to table, results of the pose estimation by using Buda head also can be seen Figure 5.8. In the graph, real poses and estimated poses are compared. Column of the graph represents the degrees range forum 0 to 360, and row of the graph stands for different analysis. In that graph, failed tests are not shown.

Figure 5.7: Left image the result of analyze 15, Right image for analyze 7



Source: This figure has been prepared by Hüseyin İnan

Table 5.1: Given poses, initial poses and estimated poses for the object buda

Analyze Number	Iteration Number	Initial Estimate (angle)			Real Pose (angle)			Estimated Pose (angle)		
		x	y	z	x	y	z	x	y	z
1	10	220	0	10	250	0	10	248,6	1,2	8,8
2	17	220	0	10	250	0	20	250,9	-1,7	20,7
3	6	220	0	20	250	0	20	248,3	0	18,5
4	10	190	0	70	210	0	70	210,1	0,2	69,3
5	8	210	0	70	190	0	70	190,3	0	68,6
6	11	310	0	90	270	0	90	271,9	-0,3	90,2
7	fail	270	0	90	310	0	90	279,5	-7	90,1
8	9	220	0	100	220	0	90	219,6	0	91
9	9	220	0	100	230	0	80	235,2	-2	84,5
10	31	280	0	140	260	0	120	259,7	-0,3	123,4
11	fail	260	0	120	280	0	140	268,5	3	108,8
12	15	280	0	160	250	0	150	251,8	-0,3	150,4
13	fail	200	0	20	280	0	40	280,5	-3,5	17
14	fail	280	0	40	200	0	20	335,5	1	27

15	11	220	0	250	250	0	240	252,2	0	240,2

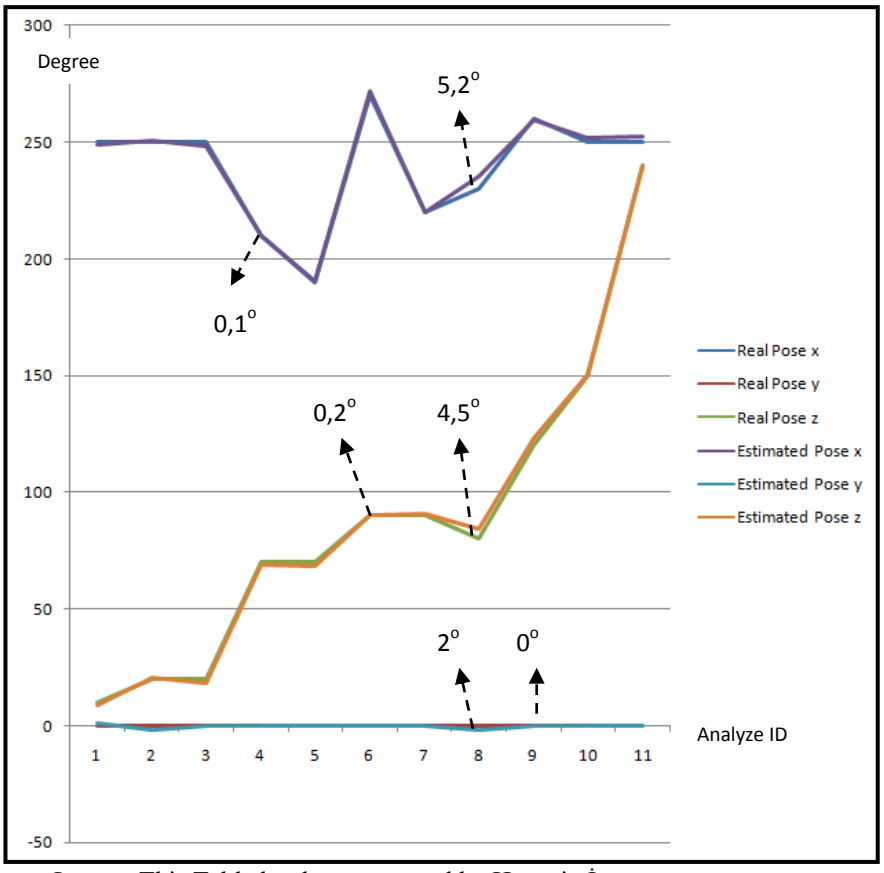
Source: This Table has been prepared by Hüseyin İnan

Table 5.2: Rotation difference between and estimation error for Buda object

Analyze Number	Iteration Number	Difference Between Initial Estimate and Real Pose (angle)			Estimation Error (angle)		
		x	y	z	x	y	z
1	10	30	0	0	1,4	1,2	1,2
2	17	30	0	10	0,9	1,7	0,7
3	6	30	0	10	1,7	0	1,5
4	10	20	0	0	0,1	0,2	0,7
5	8	20	0	0	0,3	0	1,4
6	11	40	0	0	1,9	0,3	0,2
7	fail	40	0	0	20,5	7	0,1
8	9	0	0	10	0,4	0	1
9	9	10	0	20	5,2	2	4,5
10	31	20	0	20	0,3	0,3	3,4
11	fail	20	0	20	11,5	3	31,2
12	15	30	0	10	1,8	0,3	0,4
13	fail	80	0	20	0,5	3,5	23
14	fail	80	0	20	135,5	1	7
15	11	30	0	10	2,2	0	0,2
Average error					1,47	0,55	1,38

Source: This Table has been prepared by Hüseyin İnan

Figure 5.8: The graph for representing differences between estimated poses and ground truth for buda object

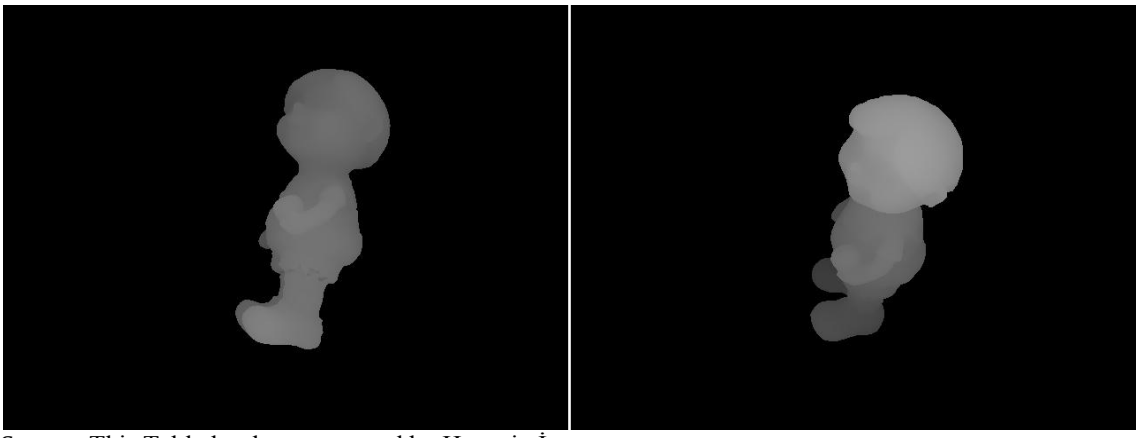


Source: This Table has been prepared by Hüseyin İnan

Addition to Buda head, we analyze our method by a boy statue that can be seen in Figure 5.9. Estimation results can be seen in table 5.3. In total, 11 tests is made for boy statue. Two of them are failed. Like Buda head, boy object's pose also tried to be estimated from variety poses. The algorithm manages to find true poses of the boy object by iterating between 6 and 41.

Images in Figure 5.9, corresponds to analyze 2 in Table 5.3 and Table 5.4. Difference between initial pose estimate and real pose is 40 degree in axis x. In addition, errors in degree, are 0.8, 0.2, 0.4 that are respectively axis x, y and z.

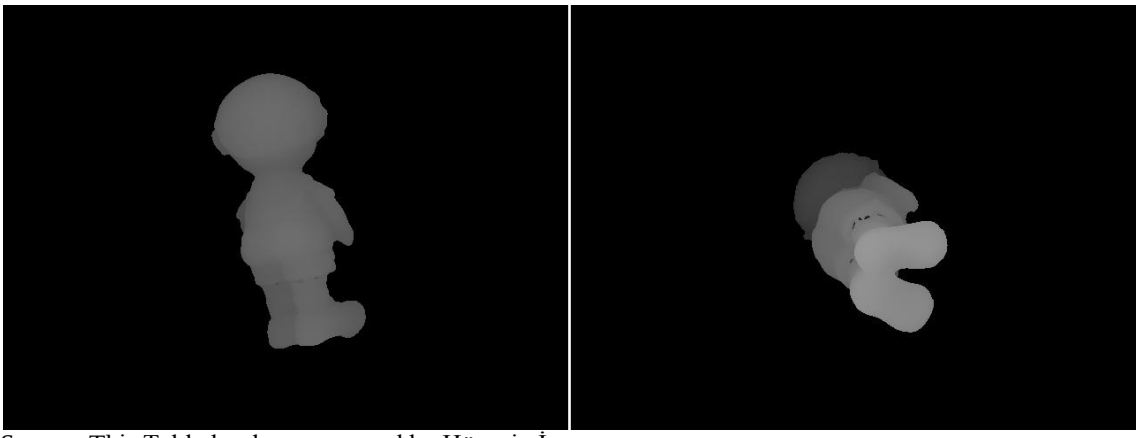
Figure 5.9: Given pose (left) and real pose of the object for analyze 2



Source: This Table has been prepared by Hüseyin İnan

Figure 5.10 shows the initial pose estimation and real pose of the analysis 6 and 7. Differences between poses are 60, 0 and 40 respectively for axis x, y and z. The pose of left image is can be estimated by using right pose as initialization. However, the pose of right image is estimated untruly.

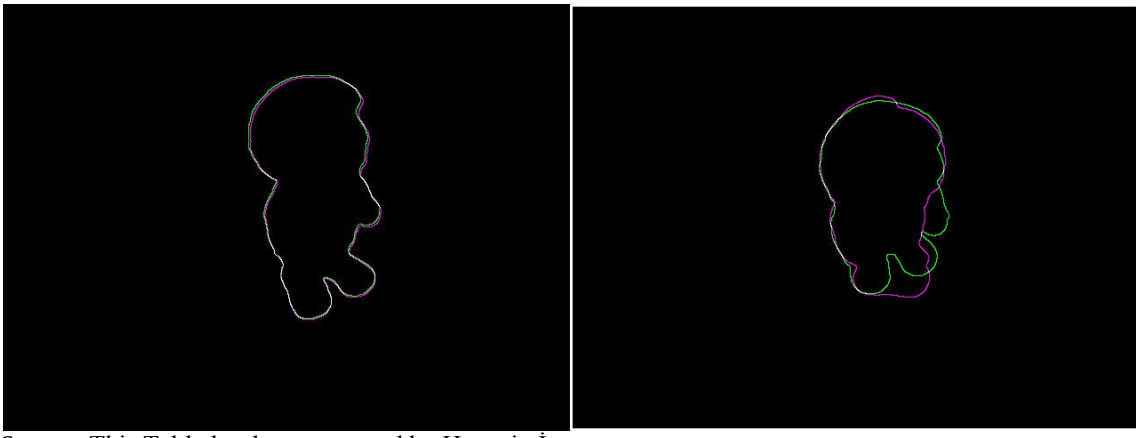
Figure 5.10: Given pose (left) and real pose of the object for analyze 6 and 7



Source: This Table has been prepared by Hüseyin İnan

In Figure 5.11, right image belongs to analyze 9, green contour is input image's contour and purple one is estimated contour. Left image illustrates analyze 11. Estimation of our algorithm is wrong and it stuck in local minimum.

Figure 5.11: Right image pose estimation result for analyze 9 and the left one for analyze 11



Source: This Table has been prepared by Hüseyin İnan

Table 5.3: Given poses, initial poses and estimated poses for the object boy

	Iteration Number	Initial Estimate (angle)			Real Pose (angle)			Estimated Pose (angle)		
		x	y	z	x	y	z	x	y	z
1	10	230	0	0	270	0	0	270,8	0,2	0,4
2	36	280	0	70	230	0	80	229,3	0	79,9
3	11	190	0	120	220	0	120	219,9	0,1	120,9
4	7	220	0	120	190	0	120	192,2	1	119,7
5	fail	280	0	210	340	0	250	248	36	228,6
6	27	340	0	250	280	0	210	279	-1,7	211
7	6	200	0	340	250	0	350	248	0	349,6
8	10	250	0	350	200	0	340	201,6	0,3	341
9	41	200	0	350	210	0	320	209,9	0,2	320,5
10	25	190	0	330	240	0	310	239,1	-0,2	311,7
11	fail	240	0	310	190	0	330	254,4	-12,3	206,2

Source: This Table has been prepared by Hüseyin İnan

Table 5.4: Rotation difference between and estimation error for boy object

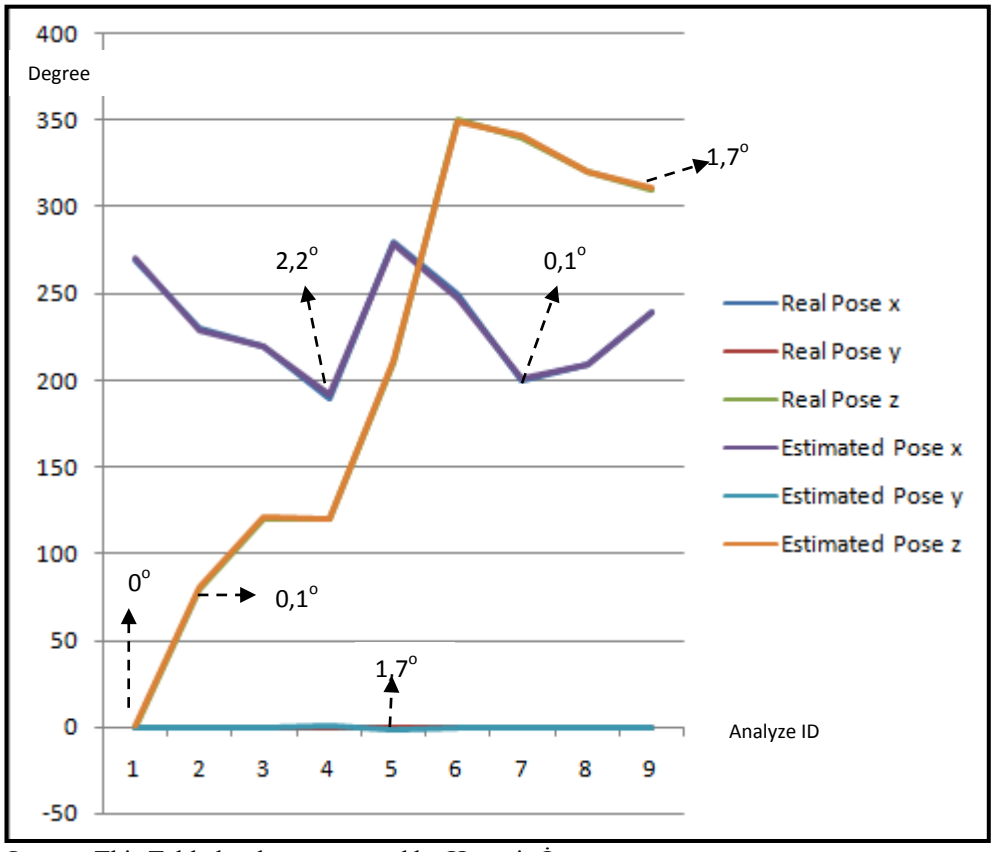
	Iteration Number	Difference Between Initial Estimate and Real Pose (angle)			Estimation Error (angle)		
		x	y	z	x	y	z
1	10	40	0	0	0,8	0,2	0,4
2	36	50	0	10	0,7	0	0,1
3	11	30	0	0	0,1	0,1	0,9
4	7	30	0	0	2,2	1	0,3
5	fail	60	0	40	92	36	0,4
6	27	60	0	40	1	1,7	1

7	6	50	0	10	2	0	0,4
8	10	50	0	10	1,6	0,3	1
9	41	10	0	30	0,1	0,2	0,5
10	25	50	0	20	0,9	0,2	1,7
11	fail	50	0	20	44,4	12,3	123,8
Average error					1,04	0,4	0,7

Source: This Table has been prepared by Hüseyin İnan

In Figure 5.12, the result of the pose estimations of Buda can be compared with ground truth. In graph, only truly estimated poses are illustrated.

Figure 5.12: The graph for representing differences between estimated poses and ground truth for boy object



Source: This Table has been prepared by Hüseyin İnan

In addition to, Buda head and boy statue, we also use angel statue to test our pose estimation method. Figure 5.2 shows mesh of the angel object. Like other objects, the algorithm is tested from variety angles of angel statue. 8 analysis are made for testing the method. 3 of them is failed to estimate pose of the object.

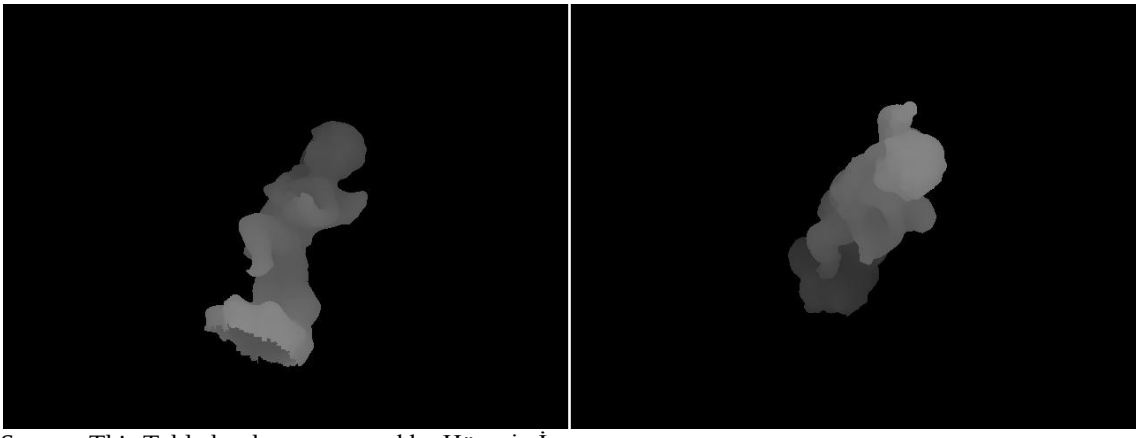
In Figure 5.13, angel statues poses are for analysis 2 in Table 5.6 and Table 5.7. Right Image's pose is estimated by using left image's pose as initial pose. The differences between poses are 80, 0, 20 in order of axis x, y and z. And the pose estimation error is o for axis x, 1 for axis y and 0,6 for axis z.

Figure 5.13: Given pose (left) and real pose of the object for analyze 2

Source: This Table has been prepared by Hüseyin İnan

In Figure 5.14, analyze 5 poses are shown, In that test left image's pose is try to be estimated by using parameters of right image's pose. However, our method fails to estimate pose of the angel. Degree differences of the poses are 80, 0, 0 in order for x, y and z-axis. If we look at in contour perspective, there is really no correlation between the two pose, so it is expected to failure of our method.

Figure 5.14: Given pose (left) and real pose of the object for analyze 5

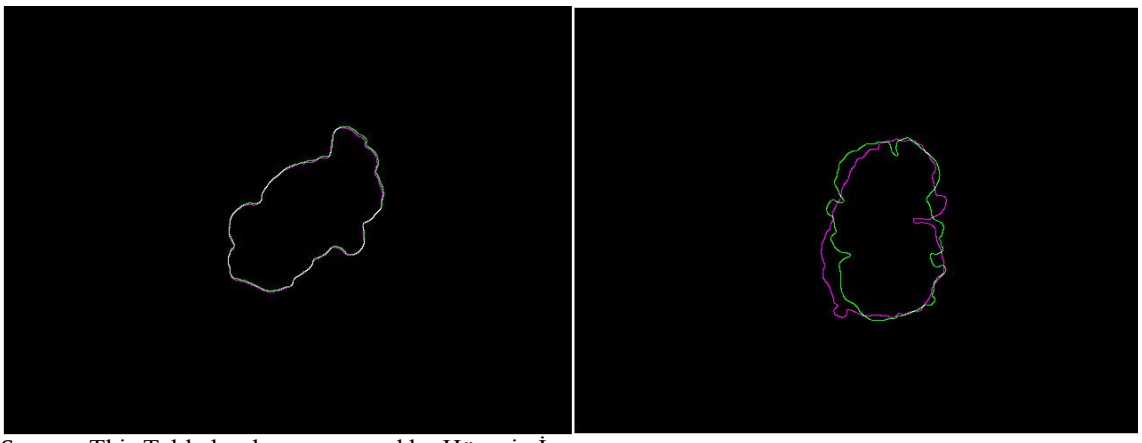


Source: This Table has been prepared by Hüseyin İnan

In Figure 5.15, green contours stands for query image contour and purple is projected contour. In right image, which belongs to analysis 7, contours are pretty aligned and so

estimated pose is accurate. Degree differences are 30 for x, 0 for y and z-axis. On the other hand, in left image which is end it up pose estimation for analysis 8 contours are so different. Algorithm is stuck in local minimum here. Variation of poses in degrees are 100, 0 and 30 in axis x, y and z respectively.

Figure 5.15: Right image pose estimation result for analyze 7 and the left one for analyze 8



Source: This Table has been prepared by Hüseyin İnan

Table 5.5: Given poses, initial poses and estimated poses for the object angel

	Iteration Number	Initial Estimate (angle)			Real Pose (angle)			Estimated Pose (angle)		
		x	y	z	x	y	z	x	y	z
1	11	210	0	10	270	0	10	272,2	-0,3	10,2
2	12	190	0	30	270	0	10	270	1	10,6
3	fail	210	0	80	290	0	70	-	-	-
4	9	210	0	240	270	0	240	270,3	-0,7	240,7
5	fail	300	0	210	220	0	210	-	-	-
6	14	220	0	210	300	0	210	298,6	0,8	209,3
7	37	230	0	100	200	0	100	199,8	0,2	99,6
8	fail	330	0	160	230	0	190	32,6	6,2	314,3

Source: This Table has been prepared by Hüseyin İnan

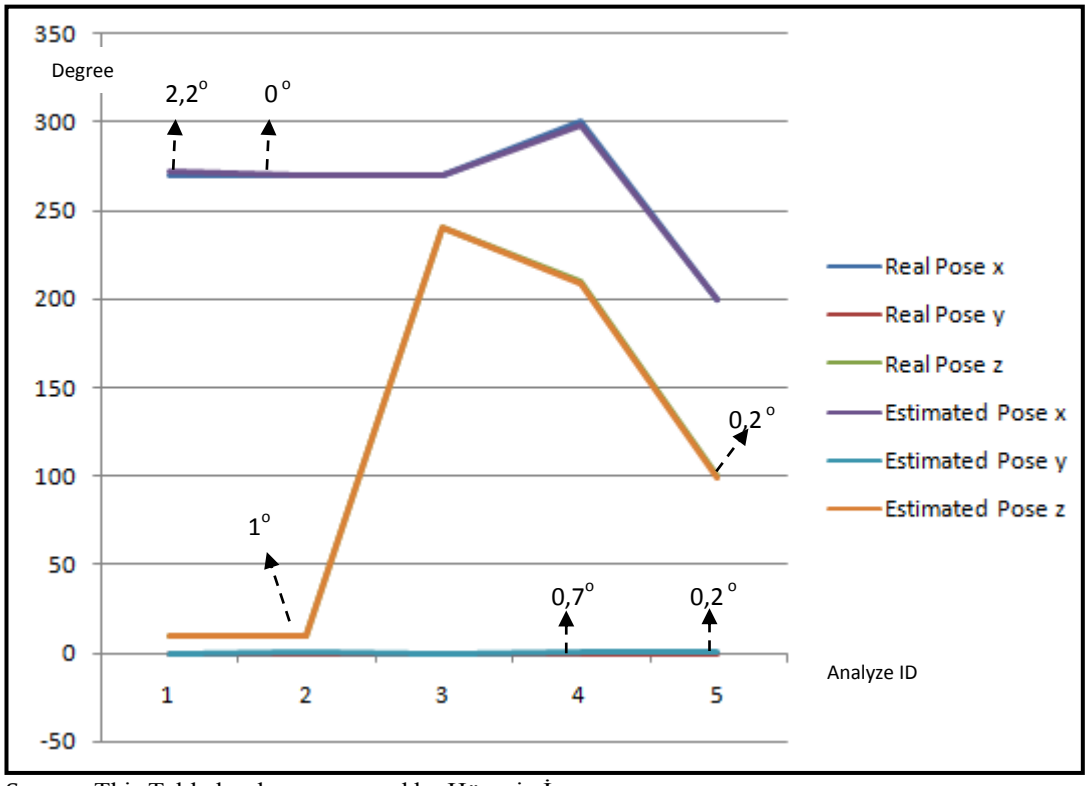
Table 5.6: Rotation difference between and estimation error for angel object

	Iteration Number	Difference Between Initial Estimate and Real Pose (angle)			Estimation Error (angle)		
		x	y	z	x	y	z
1	11	60	0	0	2,2	0,3	0,2
2	12	80	0	20	0	1	0,6

3	fail	80	0	10	-	-	-
4	9	60	0	0	0,3	0,7	0,7
5	fail	80	0	0	-	-	-
6	14	80	0	0	1,4	0,8	0,7
7	37	30	0	0	0,2	0,2	0,4
8	fail	100	0	40	-	-	-
Average Error					0,8	0,6	0,5

Source: This Table has been prepared by Hüseyin İnan

Figure 5.16: The graph for representing differences between estimated poses and ground truth angel object



Source: This Table has been prepared by Hüseyin İnan

Insofar query object in the image and projected mesh are assumed in same scale. However in real world, it is expectable to our interest objects are bigger or smaller than our projected mesh. Thus, for testing scale invariance of our algorithm, we do two different test groups. In first one query object is bigger than our projected object. In second, one projected mesh's scale is bigger than input object that's pose is tried to be estimated.

Firstly we are going to analyze the situation that interested object is bigger than projected object. We make 9 tests and 3 of them is failed to find the pose of the boy statue.

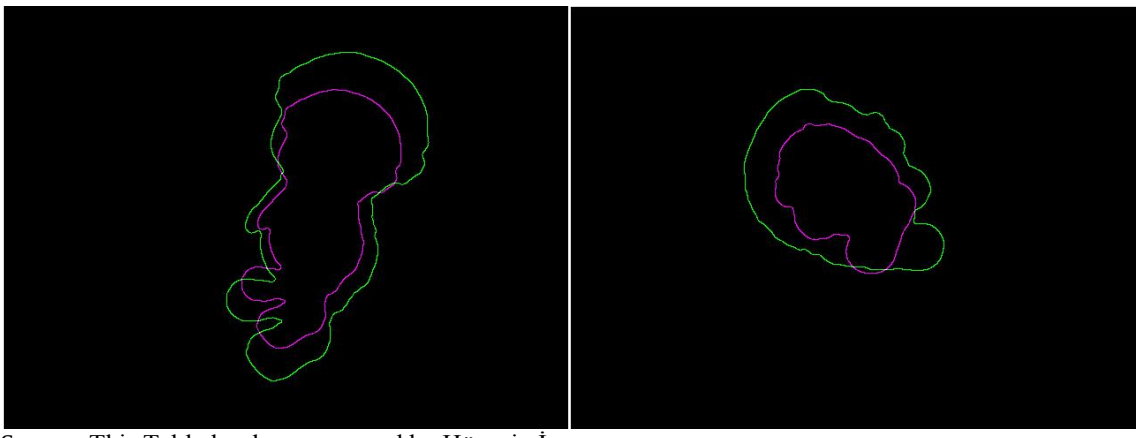
In Figure 5.17, the analysis 5 poses are shown. Object's pose in the left try to be estimated by using pose of boy in the right image. The differences between poses in degrees are 40, 0 and 30 for degrees x, y and z in order.

Figure 5.17: Given pose (left) and real pose of the object for analyze 5

Source: This Table has been prepared by Hüseyin İnan

In Figure 5.18, a green contour belongs to input image and projected mesh's contour is purple. The right image the result of analysis 9 that is achieved to estimate pose of the object. Whereas, Left Image is one of the cases that our method fails to estimate true pose of Buda. The algorithm is stuck in local minima.

Figure 5.18: Right image pose estimation result for analyze 9 and the left one for analyze 8



Source: This Table has been prepared by Hüseyin İnan

Table 5.7: Given poses, initial poses and estimated poses for the scaled object boy

	Iteration Number	Initial Estimate (angle)			Real Pose (angle)			Estimated Pose (angle)		
		x	y	z	x	y	z	x	y	z
1	16	260	0	20	230	0	20	229,8	-0,8	19,1
2	fail	270	0	90	220	0	70	-	-	-
3	fail	220	0	70	270	0	90	270,3	-0,7	52,1
4	7	190	0	270	230	0	270	229,5	0,6	270,9
5	16	300	0	120	340	0	150	340,7	0	150,1
6	16	340	0	150	300	0	120	300	0	119,4
7	fail	250	0	210	190	0	250	-	-	-
8	17	190	0	250	250	0	210	249,2	0,1	210
9	30	240	0	350	230	0	60	230,3	-0,6	60,3

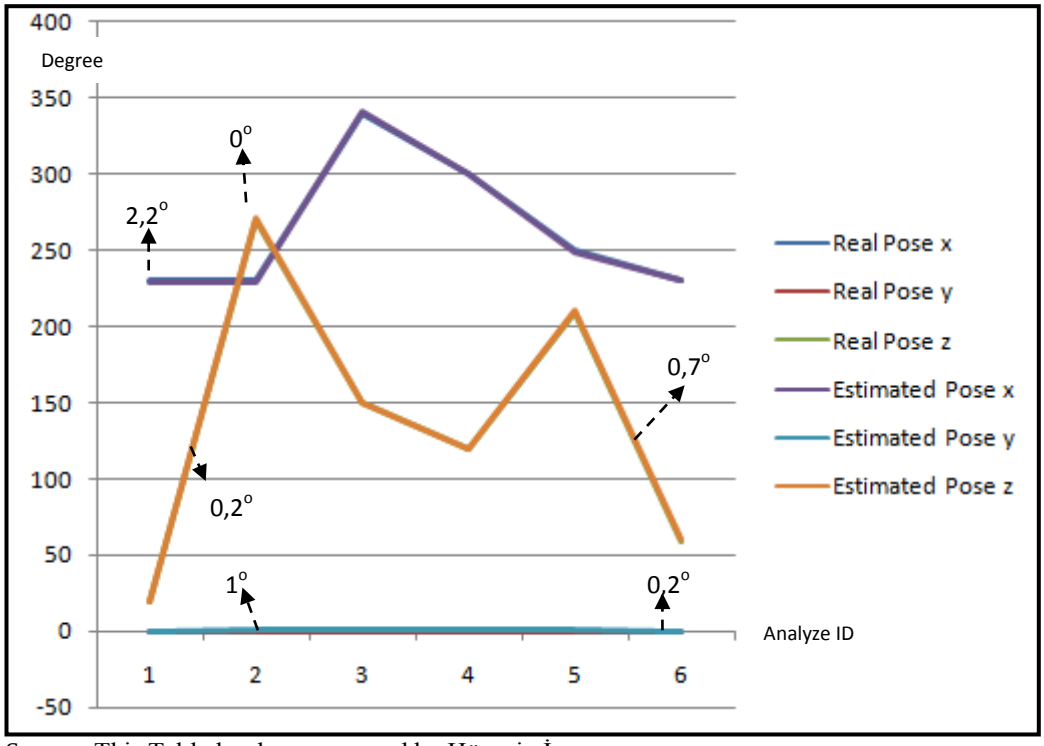
Source: This Table has been prepared by Hüseyin İnan

Table 5.8: Rotation difference between and estimation error for scaled boy object

	Iteration Number	Difference Between Initial Estimate and Real Pose (angle)			Estimation Error (angle)		
		x	y	z	x	y	z
1	16	30	0	0	0,2	0,8	0,9
2	fail	50	0	20	-	-	-
3	fail	50	0	20	0,3	0,7	37,9
4	7	40	0	0	0,5	0,6	0,9
5	16	40	0	30	0,7	0	0,1
6	16	40	0	30	0	0	0,6
7	Fail	60	0	40	-	-	-
8	17	60	0	40	0,8	0,1	0
9	30	10	0	70	0,3	0,6	0,3
Average Error					0,4	0,35	0,5

Source: This Table has been prepared by Hüseyin İnan

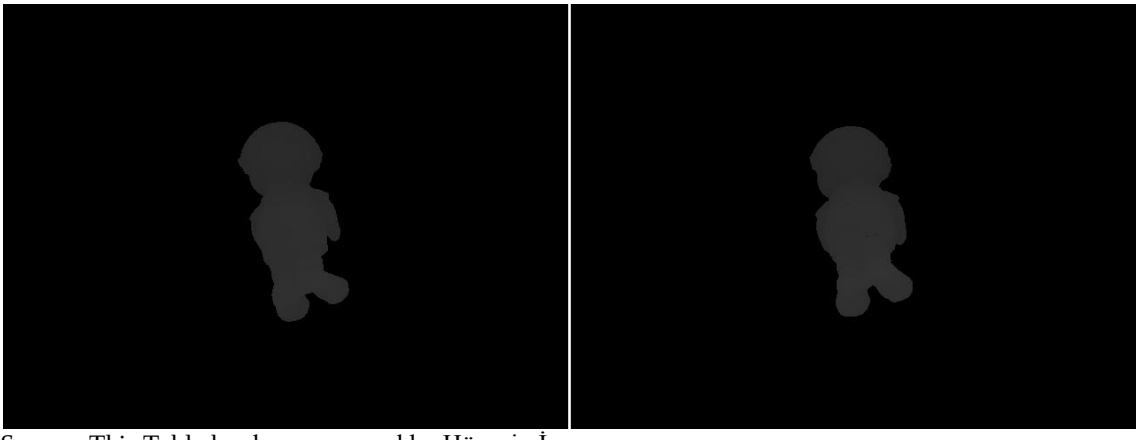
Figure 5.19: The graph for representing differences between estimated poses and ground truth for scaled boy object



Source: This Table has been prepared by Hüseyin İnan

The other case for scale invariance test is query object is smaller than projected mesh. We do nine analyses and in eight of them, we achieved to estimate true pose, and other pose estimation test fails.

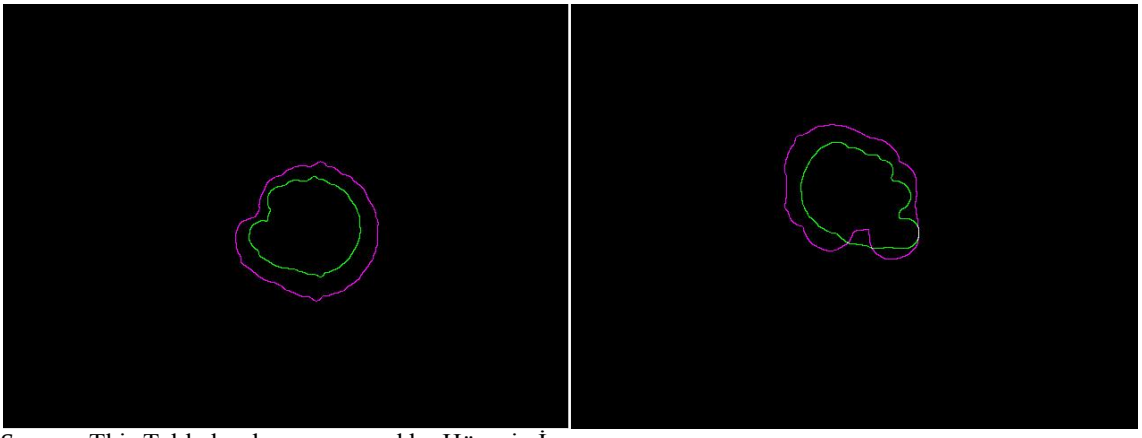
Figure 5.20: Given pose (left) and real pose of the object for analyze 7



Source: This Table has been prepared by Hüseyin İnan

In Figure 5.20, the right image is initial guess of analyze 7 and tried to estimate left image pose. Differences of rotation degrees of poses are 20 in axis x, 0 in axis y, and 10 in axis z.

Figure 5.21: Right image pose estimation result for analyze 9 and the left one for analyze 1



Source: This Table has been prepared by Hüseyin İnan

Left image in Figure 5.21, is estimated boundary (purple) and input image contour (green) are shown. Since contours correlated correctly, pose of the object's estimation is achieved. On the other hand, in left image that's are estimated contour (purple), and input contour (green) are illustrated, is estimated pose false in analysis 1. Degree differences between initial pose and real pose is 60, 0, 40 in axis x, y and z respectively.

Table 5.9: Given poses, initial poses and estimated poses for the scaled object boy

	Iteration Number	Initial Estimate (angle)			Real Pose (angle)			Estimated Pose (angle)		
		x	y	z	x	y	z	x	y	z
1	32	190	0	250	250	0	210	246,5	-1,2	212,4
2	fail	230	0	240	310	0	240	-	-	-
3	17	290	0	250	260	0	250	260,6	-1,1	249,6
4	23	230	0	350	220	0	300	220,1	0,4	299,9
5	16	220	0	300	230	0	350	229,3	0	350,5
6	7	280	0	230	240	0	210	240,9	0,7	210,4
7	9	220	0	260	200	0	250	200	-0,7	250,8
8	8	200	0	250	220	0	260	220,7	-0,3	259,2
9	20	210	0	110	190	0	110	190	-0,5	111,8

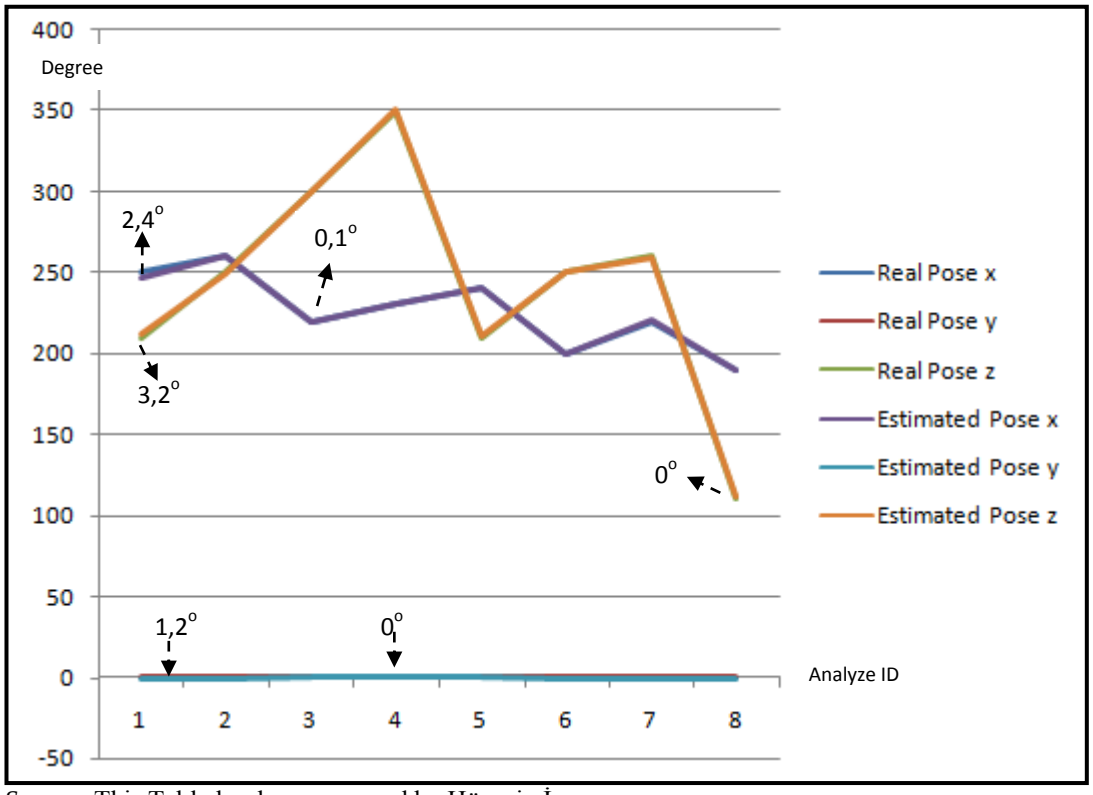
Source: This Table has been prepared by Hüseyin İnan

Table 5.10: Rotation difference between and estimation error for scaled boy object

	Iteration Number	Difference Between Initial Estimate and Real Pose (angle)			Estimation Error (angle)		
		x	y	z	x	y	z
1	32	60	0	40	3,5	1,2	2,4
2	fail	80	0	0	-	-	-
3	17	30	0	0	0,6	1,1	0,4
4	23	10	0	50	0,1	0,4	0,1
5	16	10	0	50	0,7	0	0,5
6	7	40	0	20	0,9	0,7	0,4
7	9	20	0	10	0	0,7	0,8
8	8	20	0	10	0,7	0,3	0,8
9	20	20	0	0	0	0,5	1,8
Average Error					0,8	0,6	0,9

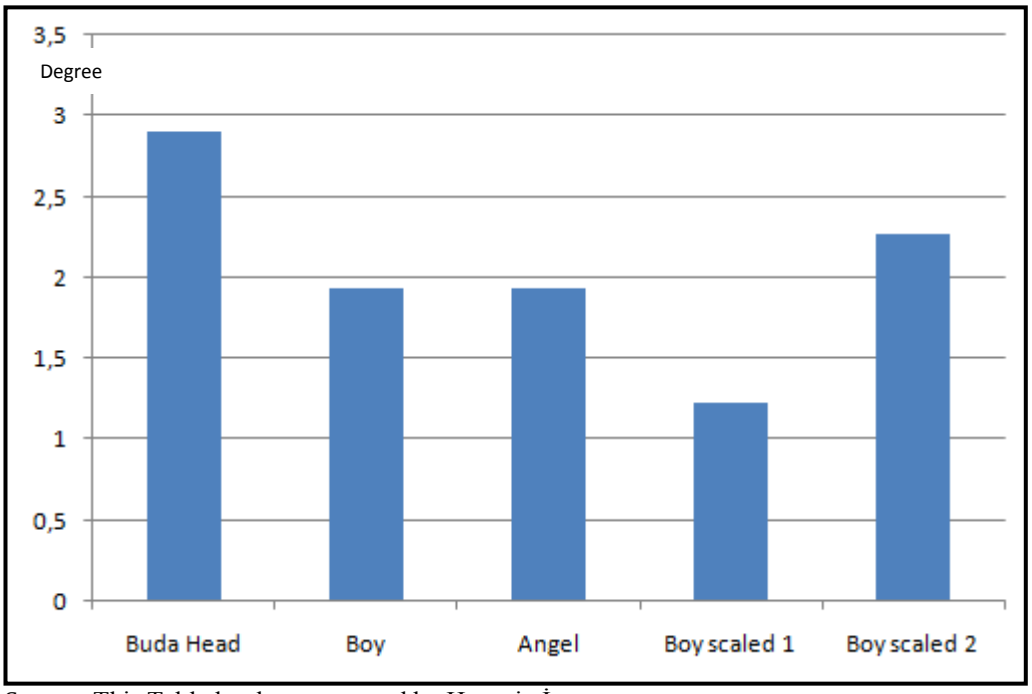
Source: This Table has been prepared by Hüseyin İnan

Figure 5.22: The graph for representing differences between estimated poses and ground truth for scaled boy object



Source: This Table has been prepared by Hüseyin İnan

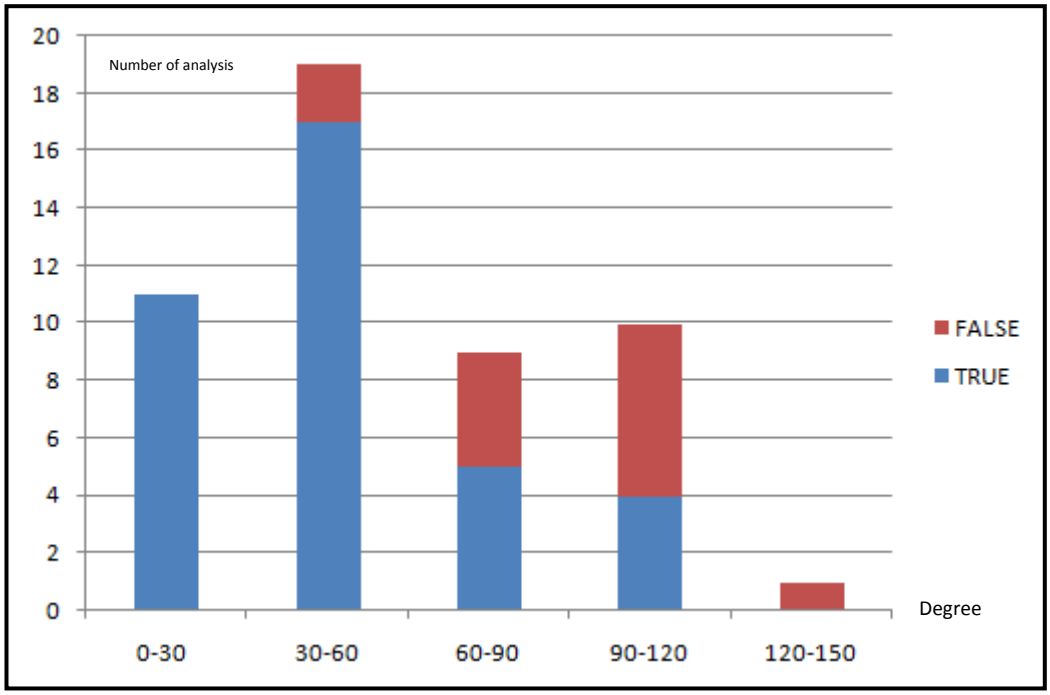
Figure 5.23: Total errors for each test groups



Source: This Table has been prepared by Hüseyin İnan

In Figure 5.23, total average errors based on test cases are illustrated. Averages of error in each axis of the cases are calculated. Then they are summed to get the error metric. The column of the graph shows error metric in degree and row of the graph stands for test cases. If the graph is examined, the largest error belongs to Buda head. Main reason of that, contour of the Buda head is not change sufficiently. It causes to algorithm to cannot discriminate variety poses of the Buda head. The boy statue has smallest error, because it contrasts to Buda head, its poses provide descriptive projected contours. Angel's error metric also similar with the boy statue. In addition, by the help of two classes we can analyze the scale factor to pose estimation. It seems that when object's scale of input image is bigger than projected object, pose estimation accuracy also increases. In contrast, when scale of the query image is smaller than projected mesh error of pose estimation also increase.

Figure 5.24: True and false estimation ratios base of given and real pose estimation rotation differences.



Source: This Table has been prepared by Hüseyin İnan

In Figure 5.24, ratios of true and false poses are illustrated according to difference between initial given pose and real pose. As shown in graph, if given pose is close (between 0 and 30 degrees) to real pose, our method achieved to estimate pose without any error estimation. If difference degree is between 30 and 60, it is also successful to finding true pose, but there is also one false estimate. Next, if variance is between 60-90 degrees, our method more likely to find true pose, even though false estimation probability is also high. Furthermore, If the difference is between 90 and 120, although the probability of finding true poses is sufficient, false estimations overweight the true estimations. Lastly, there is no true pose estimation is analyzed if difference is between 120 and 150 degree.

6. CONCLUSION REMARKS

6.1 CONCLUSION

In this thesis, we have studied estimating pose of the rigid objects with a prior 3D mesh data from 2D images. 3D model that we used is obtained by using Kinect. Only contours of the image and model are used as feature. We analyzed our proposed algorithm with 3 different objects boy, angel and buda and also for 3 different scales for boy object. Then we analyze our results in terms of rotation error. The results of our analysis are promising. The rotation error between ground truth and real pose is small enough to compute with state of the art algorithms. Additionally, we manage to estimate pose of the objects with different scales in output image. One of the drawbacks of our algorithm is that it is time consuming. It takes around between 100 minutes and 120 minutes. However, because of the limited time issues, there is no comparison data between SOTA and our algorithm.

We aim to enhance a pose estimation method for making it robust to scale variances and bad initial given pose parameters by using only Kinect-scanned textureless model. The average rotation error of the all analysis is 2 degrees. If the object that is pose is tried to be estimated is so similar in different views, the error rate is also increase. When degree differences between initial pose and estimated pose are analyzed, our algorithm works fine for rotation differences between 0 and 60 degrees and it works fine for differences 60 to 90 degrees with some false estimations.

6.2 FURTHER RESEARCH

Although the result of our pose estimation method is sufficient, there are some parts of the algorithm that can be improved. For the algorithm, establishing 2D-2D point correspondences between contours is most time consuming part of the algorithm. Indeed, it is the step that our algorithm does not work real time.

Additionally, our algorithm is not tested for occlusion. To be robust to occlusion, The ACO algorithm's performance is important, because it is the root of point correspondence between input image and 3D object. Similarly, active contours method for detecting boundary of the input image fragile for cluttered scenes. Another boundary extraction algorithm can be used for contour extraction.

Furthermore, because of our model is constructed by Kinect version 1.8 so details of the object are gone. However, using just boundaries for pose estimation is not feasible for some rigid objects. Therefore, addition to contours of the object, strong edges of the object can be also used to enhance performance of the method. For detecting edges of the mesh, depth image projection of the mesh can be used. By applying edge detection to depth image, we can get similar edges with the input image if the object is not textured. If the model is obtained by Kinect version 2.0, maybe the details of the created mesh are sufficient to use the edges.

Finally yet importantly, deformable part models methodology can be adapted to our method to make it more robust to occlusions. To put it another way, instead of establishing correspondences between contours by using whole contour, departing the contour in a logical way to compare only relevant segments of the contour. In that case, even if other object occludes one segment of the contour, by the using correspondences from others sources, true pose estimation can be still obtained.

PUBLICATIONS

3D Reconstruction of Saltanat Gate in Dolmabahce Palace. Övgü Öztürk Ergün, Bo Zheng, Bahtiyar Kaba, Hüseyin İnan, Masataka Kagesawa, Katsuhi Ikeuchi. *International Conference on Cultural Heritage, Euromed 3-8 November 2014, Cyprus.*

Intelligent Interactive Applications for Museum Visits. Kemal Egemen Özden, Devrim Ünay, Hüseyin İnan, Bahtiyar Kaba, Özgü Öztürk Ergün. *International Conference on Cultural Heritage, Euromed 3-8 November 2014, Cyprus.*

REFERENCES

Books

O. Faugeras and Q. T. Luong, *The Geomety of Multiple Images*, 2001

O. Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision, VolumeII*. Addison-Wesley, 1992.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

Journals

A. Toshev, B. Taskar, and K. Daniilidis, *Shape-based Object Detection via Boundary Structure Segmentation*, IJCV, vol. 99, no. 2, pp. 123–146, 2012.

D. Chrysostomou, A. Gasteratos, L. Nalpantidis and G. Ch. Sirakoulis, *Multi-view 3D Scene Reconstruction Using Ant Colony Optimization Techniques*, Measurement Science and Technology, vol. 23, 11, 2012.

D. Glasner , M. Galun , S. Alpert , R. Basri , G. Shakhnarovich, *Viewpoint-Aware Object Detection and Pose Estimation.* Image and Vision Computing, 30 (12), pp. 923-933, 2012

D. Leng, W. Sun, *Pose Estimation of 3D Rigid Object Based on Tentative Point Correspondences*, in Lecture Notes in Electirical Engineering Volume 128, pp 337-385, 2012.

D. Roller, K. Daniilidis, and H. H. Nagel. *Model-based object tracking in monocular image sequences of road traffic scenes.* International Journal of Computer Vision, 10:257–281, 1993

K. Ottenberg and M. Kölle, *Review and Analysis of solutions of the Three Point Perspective Pose estimation Problem,*International Journal of Competer Vision, 13, 3, 331-356, *1994*

M. Dorigo and T. Stutzle. *Ant Colony Optimization.* MIT Press, 2004.

M. Kass, D. Terzopoulos *Snakes: Active Contour Models,* International Journal of Computer Vision, 1988.

N. Pugeault, F. Wörgötter, N. Krüger: *Visual Primitives: Local, Condensed, and Semantically Rich Visual Descriptors and Their Applications in Robotics.* International Journal of Humanoid Robotics 7(3) 379-405, 2010.

R. Mester, M. Felsberg, *Probabilistic Object Models for Pose Estimation in 2D Images*, in Lecture Notes in Computer Science volume 6835, pp 336-345, 2011

T. Brox, B. Rosenhahn, J. Weickert. *Three-Dimensional Shape Knowledge For Joint Image Segmentation and Pose Estimation.* In Lecture notes in computer science : Vol. 3663. Pattern recognition (DAGM) (pp. 109–116). Berlin: Springer, 2005.

V. Lepetit and P. Fua. Monocular *Model-Based 3D Tracking of Rigid Objects :* A Survey. Computer,1(1):1–89, 2005.

Other Sources

A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski *CAD-Model Recognition and 6DOF Pose Estimation Using 3D Cues*, IEEE Int. Conf. on. IEEE, 2011.

A. G. Buch, J. Barsoe Jessen, D. Kraft, T. R. Savarimuthu, N. *Krüger, Extended 3D Line Segments from RGB-D Data for Pose Estimation*, in Lecture Notes in Computer Science volume 7944, pp 54-56, 2013

Automatic Image Calibration, http://www.roborealm.com/help/Auto_Image_Calibration.php [accessed 01 June 2015]

B. Browning and M.B. Dias, *Evaluating Pose Estimation Methods for Stereo Visual Odometry on Robots,* International Conference on Intelligent Autonomous Systems, 2011

B. Drost, M. Ulrich, N. Navab, and S. Ilic. *Model Globally, Match Locally: Efficient and Robust 3D Object Recognition*. In CVPR, 2010

B. Kaba, *Low-cost 3D Reconstruction of Objects*, Bahcesehir University master of thesis, 2014

C. Bibby and I. Reid. Robust *Real-Time Visual Tracking using Pixel-Wise Posteriors*. ECCV, 2008.

C. Choi and H. I. Christensen. *3d Pose Estimation of Daily Objects Using an RGB-D Camera*. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 3342–3349. IEEE, 2012

C. Harris. *Tracking with Rigid Objects. Active Vision*, 1992.

C. Lu, G. D. Hager, and E. Mjolsness. *Fast and globally convergent pose estimation from video images*. IEEE PAMI, 22(6):610–622, 2000.

C. Wiedemann, M. Ulrich, and C. Steger. *Recognition and Tracking of 3D Objects.* Proceedings of the 30th DAGM, (c):132–141, 2008.

D. Leng and W. Sun, *Pose Estimation of 3D Rigid Object Based on Tentative Point Correspondences*, *Computer Vision, IET* , vol.5, no.5, pp.291,300, September 2011

D. Leng and W. Sun., *Iterative three-dimensional rigid object pose estimation with contour correspondence*, Image Processing, IET , vol.6, no.5, pp.569,579, July 2012

E. Yoruk,; R. Vidal, *Efficient Object Localization and Pose Estimation with 3D Wireframe Models*, Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on, pp.538,545, 2-8 Dec. 2013

H. Zhang and p. Wighton, *Contour Correspondence via Ant Colony Optimization*,Computer, 15th Pacific Conference on , vol., no., pp.271,280, 2007

I. Badami, J. Stückler, S. Bhenke, *Depth-Enhanced Hough Forests for Object-Class Detection and Continuous Pose Estimation.* In 3rd workshop on SPME, 2013

J. Gall, B. Rosenhahn, H.P. Seidel, *Robust Pose Estimation with 3D Textured Models*. In: Pacific-Rim symposium on image and video technology (PSIVT), pp 84–95, 2006

J. J. Lim, A. Khosla, A. Torralba, *FPM: Fine Pose Parts-Based Model with 3D CAD Models*. ECCV (6) pp 478-493, 2014

J. P. Lima, V. Teichrieb, H. Uchiyama, E. Marchand. *Object Detection and Pose Estimation from Natural Features Using Consumer RGB-D Sensors: Applications in Augmented Reality*. IEEE Int. Symp. on Mixed and Augmented Reality (doctoral symposium), *ISMAR'12*, 2012, Atlanta, United States. 2012.

K. G. Derpanis, *Overview of the RANSAC Algorithm*, 2010

K. Pauwels, L. Rubio, J. Diaz Alonso, and E. Ros, *Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues*, in Proc. IEEE Conf. CVPR, Portland, OR, USA, 2013, pp. 2347–2354

L. Dryanovski, W. Morris, R. Kaushik, and 1. Xiao, *Real-time Pose Estimation With RGB-D Camera*,in Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on, 2012, pp. 13-20.

M. Zhu, K. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce and K. Daniilidis, Single *Image 3D Object Detection and Pose Estimation for Grasping*, in International Conference on Robotics and Automation (ICRA), 2014

M.A. Fischer and R.C. Bolles, *Random Sample Consensus: A Paradigm for model fitting with applications to image analysis and automated cartography.* Communication of the ACM, 1981

N. Payet and S. Todorovic, *From Contours to 3D Object Detection and Pose Estimation*, Proc. IEEE Int'l Conf. Computer Vision, 2011

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, *Contour Detection and Hierarchical Image Segmentation*, PAMI, vol. 33, no. 5, pp. 898–916, 2011

P. Besl and N. McKay. *A method for registration of 3D shapes*. *IEEE PAMI*, 14:239–256, 1992.

R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, *Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram*, in intelligent Robots and Systems (JROS), 2010 IEEE/RSJ international Conference on, 2010, pp.2155-2162.

R. B. Rusu, N. Blodow, and M. Beetz, *Fast Point Feature Histograms (FPFH) for 3D Registration*, in ICRA, 2009.

R. Girshick, H.O. Song, T. Darrell: *Discriminatively Activated Sparselets*. In: International Conference on Machine Learning, 2013

S. Hinterstoisser, C. Cagniart, S. Holzer, S. Ilic, K. Konolige,N. Navab, V. Lepetit: *Multimodal Templates for Real-Time Detection of Texture-Less Objects in Heavily Cluttered Scenes*. In: ICCV, 2011

S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, *Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects,* Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2010.

S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, *Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes.*in ACCV 2012

T. Pajdla, J. Matas, *Pose Estimation of Free-Form Objects,* ECCV pp 414-427, 2004

V. Lepetit and P. Fua, *Accurate Non-Iterative O(n) Solution to the PNP Problem.* ICCV, 2007

W. Lee, N. Park, W. Wo. *Depth-assisted Real-time 3D Object Detection for Augmented Reality*. In International Conferance on Artificial Reality and Telexistance , 2011

Y. Iwashhita, R. Kurazume, T. Hasegawa, K. Hara, *Fast Alignment of 3D Geometric Models and 2D Color Images Using 2D Distance Maps*, in fifth Conference on 3D Digital Imaging and Modeling, 2005.