MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

OBJECT DETECTION AND RECOGNITION WITH UNMANNED AERIAL VEHICLE

İBRAHİM EREN POSTACI

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor

Prof. Dr. Cem ÜNSALAN

ISTANBUL, 2020

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

OBJECT DETECTION AND RECOGNITION WITH UNMANNED AERIAL VEHICLE

İBRAHİM EREN POSTACI

525017017

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor

Prof. Dr. Cem ÜNSALAN

ISTANBUL, 2020

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

İbrahim Eren POSTACI, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended his thesis entitled **"Object Detection and Recognition with Unmanned Aerial Vehicle"**, on February 3, 2020 and has been found to be satisfactory by the jury members.

Jury Members

Prof.Dr. Cem ÜNSALAN (Advisor)
Marmara University .....(SIGN) .....

Assoc.Prof. Engin MAŞAZADE (Jury Member)
Marmara University .....(SIGN) .....

Assoc.Prof. Serkan TOPALOĞLU (Jury Member)
Yeditepe University.....(SIGN).....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that İbrahim Eren POSTACI be granted the degree of Master of Science in Electrical and Electronics Engineering in Department of Electrical and Electronics Engineering on **04.03.2020** (Resolution no: **2020/08-02**

Director of the Institute

Prof. Dr. Bülent EKİCİ

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Dr. Cem ÜNSALAN who gave me the opportunity to study a meaningful and practically satisfactory problem during my graduate studies and helped me solving a valuable problem for his invaluable help during the preparation of this thesis.

I present my eternal thanks to my esteemed family who has been with me at every stage of my work and gave me morale.

JANUARY, 2020 **İBRAHİM EREN POSTACI**

TABLE OF CONTENTS

ÖZET
İNSANSIZ HAVA ARAÇLARI İLE NESNE TESPİTİ VE TANIMASI

Teknolojinin hayatımızdaki öneminin gittikçe artması ile birlikte bilgisayarlı görü konusu çok önemli bir alana sahip olmuştur. Bu teknik, özellikle arama ve kurtarma operasyonlarında, tarım uygulamalarında, makine öğrenmesi alanında, mobil uygulamalar ve otonom araçlar konularında sıkça kullanılmaktadır. Aynı şekilde insansız hava araçları da teknolojinin ve bilgisayarlı görü konusunun ilerlemesiyle birlikte birçok farklı alanda kullanılmaktadır.

Tez kapsamında insansız hava aracı ile havadan çekilmiş görüntüler ile birlikte araç tespiti yapılmaktadır. Bizim çalışmamızın esas yapısı 'Implicit Shape Model (ISM)' çerçevesi etrafında üç farklı yöntem oluşturularak tespit işlemi gerçekleştirilecektir. ISM yöntemi bir nesneye ait parçaları nesne üzerinde tespit ettikten sonra o parçanın nesnenin merkezine doğru yönelerek birikim oluşturmasına dayanır. Bu şekilde nesnenin merkezindeki birikim sayesinde tespit işlemi gerçekleştirilir. Bu ekosistemi üç farklı nesne tespit yöntemi ile gerçekleştirip, bu sistemler harici oluşturulan farklı yöntemler ile test edilecektir.

Tez içeriğinin ilk bölümünde, ana konular olan insansız hava araçları, bilgisayarlı görü tekniklerinin analizi, yazılımsal olarak hangi ortamlardan yararlanıldığı ve literatür taraması ile benzer çalışmalar hakkındaki bilgilere değinilmiştir. İkinci bölümünde ise bahsi geçen üç sistemle ilgili bilgilendirmeler yapılmıştır. Yöntemlerin nasıl çalıştığına dair incelemelerde bulunulmuştur. Üçüncü bölümde bu üç yöntemle ilgili teknik test sonuçları ve karşılaştırmalar yapılmıştır. Son bölümde ise yöntemle ilgili değerlendirmeler yapılmıştır.

OCAK, 2020 **İBRAHİM EREN POSTACI**

ABSTRACT

OBJECT DETECTION AND RECOGNITION WITH UNMANNED AERIAL VEHICLE

Computer vision has become a very important area with the increasing importance of technology in our lives. It is used frequently in search and rescue operations, agricultural applications, machine learning, mobile applications and autonomous vehicles. Likewise, unmanned aerial vehicles are used in many different fields with the advancement of technology and computer vision.

Within the scope of the thesis, vehicle detection is performed with images taken from the air with the help of unmanned aerial vehicles. The main structure of our study will be determined by using three different methods around 'Implicit Shape Model (ISM)' system. ISM method; after identifying the patches of the object, it is based on the accumulation of these patches to the center of the object. In this way, the detection is carried out thanks to the accumulation in the center of the object. This ecosystem will be realized with three different object detection methods and these systems will be tested with different externally generated methods.

In the first part of the thesis content, the main topics are unmanned aerial vehicles, analysis of computer vision techniques, which environments are used in software and information about similar studies with literature review are mentioned. In the second part, information is given about the three systems mentioned. Investigations are made on how the methods work. In the third part, technical test results and comparisons are made about these three methods. In the last section, evaluations about the method are made.

JANUARY, 2020 **İBRAHİM EREN POSTACI**

SYMBOLS

σ : Laplacian of Gaussians scaling parameter

k_σ : Laplacian of Gaussians scaling parameter

$G_x(x, y)$: Image x direction gradient component

$G_y(x, y)$: Image y direction gradient component

$M(x, y)$: Image magnitude component

∂f_x : Two-Dimensional Gaussian function x direction derivative

∂f_y : Two-Dimensional Gaussian function y direction derivative

S : Threshold value for NormL1

p_{1i} : First target point

p_{2i} : Second target point

d_i : Train image i*th* keypoint distance value

a_i : Train image i*th* keypoint angle value

y_k : Test image k*th* keypoint y direction point value

x_k : Test image k*th* keypoint x direction point value

y_{knew} : Test image k*th* keypoint new x direction point value

x_{knew} : Test image k*th* keypoint new x direction point value

μ : Mean of sequence

ϑ : Variance of sequence

$F(x, y)$: Two-Dimensional Gaussian function

pix_{tp} : The remaining pixels result of logic and process

pix_{gt} : The labeled pixels with ground truth

pix_{fp} : The mismatched pixels result of logic and process

pix_{all} : The all pixels found in the method

ABBREVIATIONS

OpenCV **:** Open Source Computer Vision Library

UAV **:** Unmanned Aerial Vehicle

SVM **:** Support Vector Machine

HOG **:** Histogram of Oriented Gradients

KNN **:** K – Nearest Neighbour

GBT **:** Gradient-Boosting Trees

ISM **:** Implicit Shape Model

GMSR **:** Gradient Magnitude Based Support Regions

FAST **:** Features from Accelerated Segment Test

SURF **:** Speeded-Up Robust Features

SIFT **:** Scale-Invariant Feature Transform

MLE **:** Maximum Likelihood Estimation

DoG **:** Difference of Gaussians

LoG **:** Laplacian of Gaussian

IIR **:** Infinite Impulse Response

LSD **:** Line Segment Detection

LSR **:** Line Support Region

BRIEF **:** Binary Robust Independent Elementary Features

ORB **:** Oriented FAST and Rotated BRIEF

TPR **:** True Positive Rate

FPR **:** False Positive Rate

ROC **:** Receiver Operating Characteristic

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Computer vision is a field of computer science that works on enabling computers to see, identify and process images in the same way that human vision does, and then provide appropriate output. It is like imparting human intelligence and instincts to a computer. In reality though, it is a difficult task to enable computers to recognize images of different objects. Computer vision is closely linked with artificial intelligence, as the computer must interpret what it sees, and then perform appropriate analysis or act accordingly. Computer vision application is getting popular today in different fields.

As for the technical details of computer vision algorithms, these algorithms can be implemented by using object oriented software such as Python, C ++ and MATLAB. We have written algorithms in terms of object orientation as well as spelling language, and this is the reason we prefer the MATLAB software language. MATLAB is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. MATLAB is simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. MATLAB supports modules and packages, which encourages program modularity and code reuse. The MATLAB interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Debugging MATLAB programs is easy: a bug or bad input will never cause a segmentation fault.

Computer Vision Toolbox is the most important tool of MATLAB software language, which includes computer vision algorithms. The tool algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality. The results are better seen in systems created with the

help of Computer Vision Toolbox. It has a very advanced structure especially in three dimensional analysis. This feature makes it easier to analyze the results.

Another tool to be used in this thesis is unmanned aerial vehicles. An unmanned aerial vehicle (UAV) is a type of aircraft that operates without a human pilot onboard. Recent technologies have allowed for the development of many different kinds of advanced unmanned aerial vehicles used for various purposes. An unmanned aerial vehicle is also known as a drone. Over just a few decades, the unmanned aerial vehicle has gone from a science fiction concept to an everyday reality. Many of these applications were developed in the military, and the general public has seen UAVs evolve as spy or reconnaissance vehicles used during wartime. However, recently consumers have also seen the development of these types of aircraft for public markets. Remote control airplanes and helicopters can also be classified as UAVs when they have particular kinds of performance and remote control capabilities. One example is the proposed use of flying robots, or UAVs, to deliver packages. Although the technology has become relatively common, the UAVs are often still associated with military objectives. The use of UAVs for surveillance is a controversial topic. Other lower-profile uses for unmanned aerial vehicles include firefighting and police use, as well as other kinds of domestic surveillance.

If we look at the search and rescue drones, which are closely related to the thesis, A search and rescue drone is an unmanned aircraft used by emergency services, such as police officers, firefighters or volunteer rescue teams, ideal for searching over vast areas for missing persons and crime victims in need of rescue and in any environment.

UAV scan provide real-time visual information and data in the aftermath of an earthquake or hurricane. They can also become an eye in the sky to locate a lost person in the mountain for example. When a disaster or incident threatens lives and livelihoods, emergency responders need information and real-time imagery in order to make better decisions and save time. UAVs can provide situational awareness over a large area quickly, reducing the time and the number of searchers required to locate and rescue an injured or lost person, greatly reducing the cost and risks of search and rescue missions.

The possibilities for helping ensure public safety are endless. The main important approach in human search and rescue operation is the infrared (IR) thermal imaging

camera that can detect human body heat. This capability greatly increases the ability to find people or objects at night that may be hidden, even during daytime operations.

If we enter into the general approach of this thesis, its main objective is to real-time objects detection and recognition with an autonomous drone. This process will be realized by using two main approaches in object detection and identification process. These two main processes are Histogram of oriented Gradient (HoG) and implicit shape model. There are studies on these two approaches and there are articles sharing at the academic level. First, we will examine the work done with the pedestrian detection algorithm. Then, we will examine the implicit shape model approach and give information about the part about UAVs.

If we look at the first study of detection and recognition with pedestrian detection algorithms, this study is based on the body, arms, head and legs of the human body [4]. In this study, train data sets are taken through Support Vector Machine and feature extraction is done through HoG. If we define Support Vector Machine (SVM) and HoG, SVM is machine learning algorithm that analyzes data for classification and regression analysis. In this study, SVM is a supervised learning method that looks at data and sorts it into one of two categories. In study, an SVM outputs a map of the sorted data with the margins between the two as far apart as possible.

SVMs are used in text categorization, image classification, handwriting recognition and in the sciences. This image is also blurred when the results are more likely to be efficient. In addition, HoG is a feature descriptor used in computer vision and image processing for the purpose of object detection. In this study, the technique counts occurrences of gradient orientation in localized portions of an image and this method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. However, in this algorithm, the human color and the figure in the picture to detect the addition of the Skin Color Detection algorithm can be improved by adding a little more.

In a different study, feature extraction for body detection and SVM and K-Nearest Neighboor (K-NN) algorithms were used for recognition [5]. In this study, a k-nearest-

neighbor algorithm, often abbreviated K-NN, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in. The K-NN is an example of a "lazy learner" algorithm, meaning that it does not construct a model using the training set until a query of the data set is performed. Feature extraction for shape and color feature made it possible in four steps. In the second stage, the data obtained by SVM were obtained and compared with K-NN and obtained a result. Here, this study make a determination process by taking the steps indicated for body fixation. In fact, a study was created in the same logic as the second study.

Wu and Nevatia did combining body parts with Bayesian technique in the approach for human detection among the community [6]. If we can introduce to define bayesian classifier, bayes classifier is popular in pattern recognition because it is an optimal classifier. It is possible to show that the resultant classification minimizes the average probability of error.

Bayes classifier is based on the assumption that information about classes in the form of prior probabilities and distributions of patterns in the class are known. It employs the posterior probabilities to assign the class label to a test pattern; a pattern is assigned the label of the class that has the maximum posterior probability. The classifier employs Bayes theorem to convert the prior probability into posterior probability based on the pattern to be classified, using the likelihood values. In this chapter, we will introduce some of the important notions associated with the Bayes classifier. The human body is determined by combining certain body parts (which are leg, head and body) with the Bayes technique. The aBoosting technique is applied to teach the mentioned body parts to the algorithm. The term ‘Boosting’ refers to a family of algorithms, which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor.

In a more recent study, the Gradient Boosting Trees method, which is designed for pedestrian detection, has been proposed, which allows for more efficient use of existing object detection OpenCV algorithms [7]. Gradient boosting is a machine learning

technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

To compare the efficiency here, the Gradient Boosting Trees method has been compared with the SVM model. Most of the available versions of GBT are limited to small dimensions of processed data or supported classes of the problem (regression only). In this study, a fast and flexible version of the algorithm has been developed. It uses the OpenCV decision tree application, which allows it to work effectively with mixed type data (both categorical and numeric) and to handle missing values.

Geisman and Schneider did two-stage approach to identify pedestrians in a moving vehicle [8]. The system combines the advantages of the two family members into two stages: the first stage uses a quick search mechanism based on simple features to detect interesting areas. The second stage uses a more expensive but also more accurate array of properties to classify by pedestrians in these regions. In the first stage, a detection mechanism based on Haar properties is used. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Viola and Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. The second stage classifies incoming samples using HOG-based properties and a linear SVM.

Implicit Shape Model (ISM) has been proposed in the article which contains an approach that performs object recognition and segmentation in one operation [9]. In traditional approaches, object recognition and segmentation are not performed at the same time. The process that performs both recognition and segmentation in one operation makes the ISM approach. As a result of this recognition and segmentation process, it produces accurate measurements of the location and reliability of the pixels so resulting in a reliable method.

This process is performed in two stages. The first is the Codebook of Local Appearance, which contains the information of the local locations of the objects, and the second is the ISM approach, which specifies where the object entries in the Codebook of Local Appearance can occur. This model do not try to define an explicit model for all

possible shapes a class object may take, but instead define “allowed” shapes implicitly in terms of which local appearances are consistent with each other.

Liebe, Schiele and Leonardis did an interface model that is proposed between the visual information that can be easily obtained from the image and the higher-level semantic information that can be used by reasoning processes [10]. This cannot be done solely on the basis of visual similarity, as the appearance of the object parts can vary considerably. Rather, this article also proposes to use co-location and co-activation with weak and top-down constraints, such as compliance, as guidelines for learning the appearance of local object parts. At a later stage, Bayesian network was used to apply these hypotheses to complex scenes.

Gall, Razavi and Gool did multi-class object detection by Random Forest method [11]. In this study they describe the general framework of random forests for multi-class object detection in images. It also contain that object detection is formulated as a combined regression and classification problem. When the detection problem becomes a distribution estimation problem, the random forests allow to learn features and descriptors that are optimal for estimating the distributions with low uncertainty.

Ünsalan and Sırmaçek that detects buildings with satellite images, vector analysis was performed from the image before the detection [12]. Four different methods are used to get these vectors. These methods are These vectors then modeled the locations of buildings as random variables. And then they estimate them as probability density functions (pdf). Then, where these estimates are concentrated, the building is detected.

Ünsalan and İlsever did HoG method for detection in a different study that detect the building with satellite images [13]. Four steps have been proposed to calculate HoG detectors. This calculation is examined around Gamma / Color normalization, Gradient computation, Spatial / Orientation binning, Normalization and descriptor blocks headers. Then the calculated descriptors were traced with the SVM. While doing this, it is provided to learn the buildings targeted from several different images.

Razavi and Gall did latent Hough transformation for object detection [14]. In this study, the object is voted according to the position in the image. Although these votes are useful for the detection of the object in the image, they also produced incorrect values for the position, color and shape of the object. This study aims to strengthen the correct values

with latent variables. The linearity of the methods based on Hough transformation was used to train the latent values.

Abughalieh and Sababha did a general monitoring and guidance system has been established with weight based sensors [15]. In this system, non-static sensors are used for speed detection and in addition for UAVs, color filter is used for object detection and recognition. FAST, SIFT and SURF algorithms were tested and implemented for key points. In other words, the system is based on motion detection and the dominant feature of the object. Color histogram calculations were made to determine dominant characteristics.

Chen and Meng did the studying of vehicle detection with UAVs, the combination of SIFT and Implicit Shape Model (ISM) has been proposed as a method [16]. First, the key points in the test image were determined using the SIFT algorithm. Then ISM was used to determine the object properties in the image. Finally, the key points were selected with the Support Vector Machine (SVM). The most critical point in this study is to record the unchanging points with the codebook and to identify the vehicle with key points. According to the results of the article, it achieved a success rate of around 94%, which is a very important result.

Jüngling, Becker and Arens did ISM and SIFT algorithms in the object detection and recognition process unlike the above study, a cascade structure is proposed [17]. For example, people with bags and people without bags were used in the experiment. The nature of the objects in the nature of the common aspects of the work and want to identify them separately in a study, ISM and SIFT algorithm that we obtained from the codebook is used. Each detected point is voting toward the center of the object. It is also a real-time transaction that uses a simple Kalman Filter in its motion estimates.

Seo did a prediction method to simplify the complex operations of algorithms and to obtain a more efficient result for detection [18]. With the estimation method developed in this study, the HoG algorithm will be used for the detection process after the system is made efficient. First, the noise of the picture is cleared with the help of Gauss Filter. Canny edge algorithm is applied to the cleaned image. Then, with the formula called BITCOUNT, the edges of the edge filter are eliminated by inserting this formula into the formula, which is made here. Then, with the remaining edges of the operation, the HoG

algorithm performs the detection process. As a result, the estimation and the use of HoG observes that the results are better than HoG algorithm.

Rematas and Leibe did developed a more efficient object recognition and detection process, a cascade Hough Forest ISM method was proposed [19]. The Hough Forest method consists of an effective adaptation of the random forest method to Hough Transform. The Hough Forest method gives more successful results compared to the object detection and recognition processes based on the Hough theory. This method has been developed around two main headings. First, the probability segmentation for the object has been improved more efficiently. Secondly, it has been developed that these segmentations can be used more effectively in object selection. At the same time, they have developed an efficient cascade selection structure.

Barinova, Lempitsky and Kohli did developed a new probabilistic structure by utilizing the Hough Transform, a straight line detection and pedestrian detection study was conducted [20]. Train method and selection process used in ISM and Hough Forests methods. This method provides better energy usage, more precise results and easier integrable environment than Hough Transform. Instead of solving the local peaks in the Hough image, the method simplifies the process by selecting a global (global) maximum. According to the results, this method has been shown to be more successful than traditional Hough-based determinations.

Pan and Chu did probabilistic pairwise voting method for object detection is proposed. This method is mainly based on circular object detection method [21]. Conventional circular object detection two main groups which are voting-based and maximum likelihood estimation (MLE). These methods do not give good results both in terms of process complexity and noise. The method proposed in the article has yielded significant results in terms of improving these results. The study was divided into three main groups. First, center, radius and circle parameters were found for circular object detection. Secondly, image noise, occlusion and deformation were examined. Finally, an efficient algorithm for multiple circular object detection has been developed.

In another study with object detection, the hierarchical Implicit Shape Model structure was proposed [22]. Implicit Shape Model structures were described in the previous paragraphs. In this study, unlike the others, a two-stage detection process is

performed. First, local ISMs are used to model object particles. Secondly, the structure of the components related to the object center is modeled by the global ISM. The proposed approach is applied to some data sets. Then, the performance of the algorithm is mixed with comparable methods. The results show that the method has good performance.

2. METHOD

2.1 SHAPE REPRESENTATION

2.1.1 Implicit Shape Model

In this part of the thesis, we will talk about three different approaches of the subject discussed in the thesis. The subject we have discussed in this thesis is the results obtained from three different approaches to the Implicit Shape Model (ISM). We think that the ISM method mentioned in the introduction part should be explained in more detail.

Implicit Shape Model is a study proposed by Liebe [30]. The main point of this study is the structure that enables the object or objects to define the objects implicitly together with their identifiers. By identifying the object with any identifier for this study, it provides the possibility to make identification on another image through the ecosystem of this structure. Implicit Shape Model structure can be examined under two main headings. One of them can be explained as Codebook structure and Hough voting process.

Codebook structure is a method developed for object definition. According to this method, there are related points on the image of the train. These points may be the techniques used in image processing, in which the Harris technique is used. Once these related points have been found, the image fragments around those points are stored. This way the parts of the object are stored in this way. For matching, these parts are expected to match in the test image.

Hough Transform is a work proposed by Hough [1]. In Hough voting process, matching parts are provided to vote towards the center of the object. This structure is related to the Hough transform structure. The object of the technique is to find defective examples of objects in a particular shape class by a voting procedure. This voting procedure is performed in a parameter field in which the object candidates are obtained as local maxima in a collector field explicitly generated by the algorithm for calculating the Hough transformation. Thus, it was ensured that the parts in the fixing process were collected in certain local areas and a voting process was formed.

This transform was originally created for line detection only. This transformation was then used for the detection of circles, ellipses, and even irregular shapes, which was called the "Generalized Hough Transform" [2].

Applying the Implicit Shape Model, patches created by cutting from the relevant points are stored in one place. Then these patches are searched for in the test image. Matching patches are stored and probabilistic voting is performed. Vote space is created and the most accumulated local point is determined by the Hough technique. And then, according to this back projection hypothesis, the object is tried to be implicitly validated. Purification of the hypothesis is achieved with the help of uniform sampling.

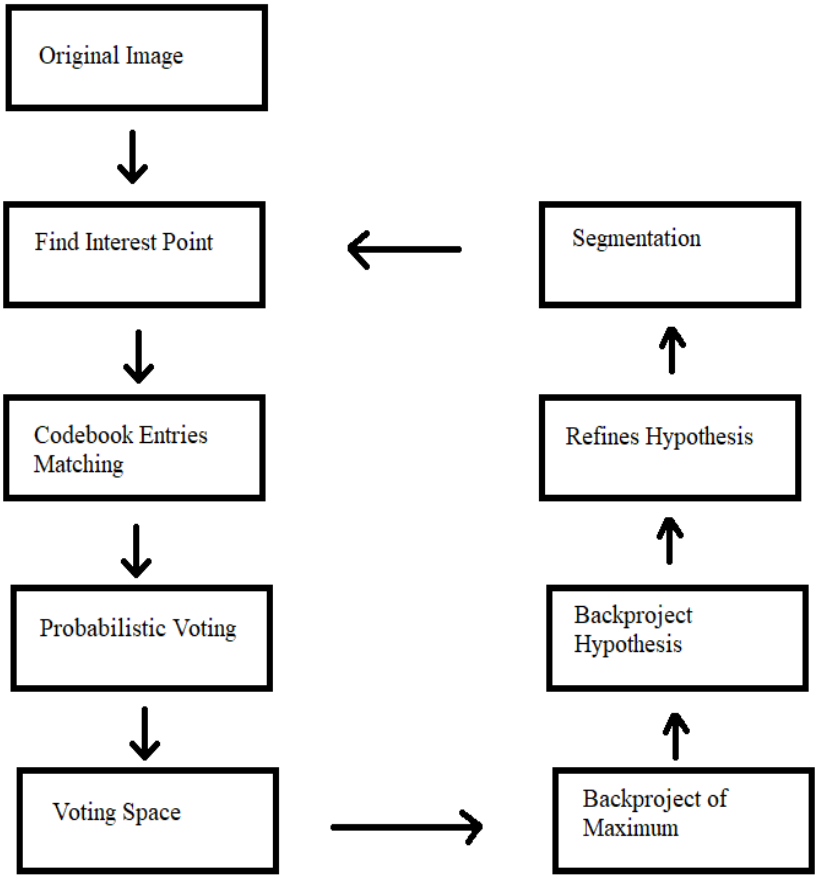


Figure 2.1: ISM ecosystem

ISM ecosystem covers the properties of the train images in an implicit manner and enables the object to be found in different test images. Here, of course, the methods used as object detection and recognition methods are used. Object detection and recognition can be used instead of codebook entries in the ISM ecosystem. Because, in this part, the test parts are matched over the image parts of the train images. We want to do is to detect object properties by using object detection and recognition algorithms instead of codebook entries. If these features match in the train and test sections, we will continue the operations in the remaining ISM ecosystem. Methods that we will use in our study are Speeded-Up Robust Features (SURF), Line Segment Detection (LSD) and Gradient Magnitude Support Region (GMSR). Continuing from the ISM ecosystem, after finding the features and descriptions of the objects we named above this methods, we stored the properties of the properties found in the train image. Since these properties are point based in the SURF method, we have stored the direct coordinates in this method. In the LSD method with GMSR, the locations found as properties are not points but areas. So, we kept the centers of gravity of these areas.

2.1.2 A Probabilistic Framework to Detect Buildings

In the method proposed by Ünsalan and Sırmaçek [12], a probabilistic framework was created using local feature vectors. This structure is proposed for detecting buildings from aerial and satellite images. Four different methods have been proposed for local feature vector extraction. These methods; Harris Corner Detection, Gradient Magnitude Based Support Regions (GMSR), Gabor Filtering in Different Orientations and Features from Accelerated Segment Test (FAST). This is an observation for the probability density function (pdf) to be estimated with four local vector inferences. Then find the corresponding pdf value using the kernel density estimation method. Thus, the locations of the detected objects are represented by a random variable.

After obtaining the vectors with the four different feature extraction methods mentioned, it is necessary to customize it on the object. We need to make this customization both for vectors in different locations and because we don't know how many objects there are. The kernel density estimation method is used for this

customization. Then the probability framework structure is used. Since these vectors are structured separately for each method, the probabilistic framework method combines them all to obtain a better result in terms of location.

Gaussian structures are placed in the locations where these vectors are located. Gaussian structure is formed by exponential function with concave quadratic function [23]. We used two-dimensional function in this thesis, because our videos and images are two-dimensional.

Two-dimensional Gaussian function is,

$$f(x,y) = C \exp\left(-\left(\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}\right)\right) \quad (2.1)$$

If we explain the values in this formula, C is constant amplitude, x_0 and y_0 is the center point, σ is controlling its width of the Gaussian function [24]. As can be understood from this formula, Gaussian function takes the highest value at the center point. Then, the number of vectors in the results obtained is different, we see some less and some more Gaussian structuring.

Two methods have been proposed for the Probabilistic Framework. The first finds the pdf of the detected object by gathering the vectors from all methods into a single group. The second method is based on the decision structure. Here, a final pdf is obtained by mixing the predicted pdfs in different ways. When mixing estimated pdfs, a weight is assigned to each of them, which is directly proportional to the maximum mode values. By pinpointing object positions from the estimated pdf, the mode is tagged with the maximum value as a building. By normalizing four different pdfs in this way, the final pdf estimate is obtained.

In our approach, inspired by these studies, we took parts from both studies. In more detail in the following chapters, we would like to present a summary of our thesis here. In our study, we use the object detection and recognition algorithms of train images at first, which we explained in the next section. These structures correspond to codebook entries in the ISM ecosystem. If this structure is the SURF method, we obtain the angle and distance values that are directed towards the center of each coordinate value. But in the field-based study, we take the center of gravity of the fields and obtain the angle and distance values that carry them towards the center. We match the descriptors obtained with the descriptors obtained in the test image. For this matching process, we use the structure that the SURF structure has already developed for its identifiers. For field based studies such as GMSR and LSD, we calculate and use the mean and variance values of these fields for matching process. We make them accumulate in the center of the object with the angle and distance values that we find.

After finishing the voting process, we place Gaussian structures in the locations. We apply this process in line with the information we received from probabilistic framework. The situation we expect is to see a large Gaussian structure after finding the object. Although it also finds different locations that are not in the center of the object, we have identified the object because we have achieved a higher Gaussian in the center of the object thanks to this structure.

2.2 FEATURE EXTRACTION

2.2.1 Speeded-Up Robust Features (SURF)

SURF algorithm is very similar to SIFT algorithm. The differences between the SURF structure that we use will be examined. First, the SIFT structure is explained. Then, the differences between SURF structure are examined. Then the application of the method to the system is explained.

Firstly, the SIFT structure is an object descriptor published by Lowe [25] and can be found unchanged under effects of uniform scaling, orientation, illumination changes, and partially invariant to affine distortion of an object given in the train image. It provides a good result by reducing the share between the scaling and the noise of the image, thanks to its local variables, because it generates a large number of keypoints and descriptors for an object. To reason that the SIFT feature identifier is invariant for uniform scaling, orientation, and illumination values. This invariance structure is essentially the work created by Serre and Kouh [26]. Around the four main headings, we can determine how this invariance structure is formed. Four main topics which Lowe's [27] defined are, scale-space extrema detection, keypoint localization, orientation assignment, keypoint descriptor.

First of all, scale-space extrema detection is implemented for invariance of the points to be determined for the object against scale and orientation effects. This process is carried out with the Difference of Gaussian (DoG) method, which is less costly approach of Laplacian of Gaussian (LoG). LoG are scale-space filter and it produces σ scaling parameter. We can reduce and increase the scale with this parameter. On the other side, DoG structure produces two main scaling parameter, σ and k_σ. When the DoG process is finished, it produces a pyramid at different octaves. It compares the points at different scales to find the local maximum value. The scale with the local maximum value is the dominant scale and the probability is chosen considering that the keypoint is best represented at that scale. For the initial values in this variant is number of octaves 4, number of scale levels 5, initial σ 1.6 and initial k_σ 1.4142. This event does not actually make a difference if the interest points are too small. But if the interest points are on large

scales, then this method significantly reduces the scale effect on the descriptor. It prevents scale and orientation with DoG operation, and ensures accuracy, stability, scale & rotational invariance.

Secondly, Taylor series expansion method is used to increase the accuracy of the captured potential interest points. If the threshold value in Taylor series expansion is below 0.03 specified in the article, these values are discarded. This increases the accuracy of keypoint localization. This structure is also used in structures such as the Harris corner detector, which basically eliminates the low probability interest points and takes into account strong interest points.

Thirdly, a field is created for the rotation, taking into account the neighbors of the potential keypoint. Then the gradient magnitude and direction of these fields are calculated. 36 structures are created to cover all angles, including 360 degrees. The histograms of these structures are calculated and finding the highest value. If the peaks of the other areas are 80% or more above this highest peak, the orientation is complete. After the operation minimizes differences in orientation and rotation. As a result of this process keypoints are strengthened against orientation and rotation effects and a more pure result is provided.

In the fourth chapter, the method of creating keypoint descriptors is specified. A vector structure was formed by calculating the local image gradients of the areas where the keypoints were created with their neighbors. There are 128 values in this vector. With this transformation, local shape distortion and illumination changes are aimed to be prevented.

SURF was proposed by Mr and Tuytelaars [3] and was created faster than the SIFT method. SURF was first used Laplacian of Gaussian structure with the help of box filter as opposed to Difference of Gaussian structure in SIFT. With this approach, the calculation between the images is made easier and faster. He used a 6s intensity wavelet transform for orientation. This wavelet trasform responds quickly. This gives a better result in terms of speed and cost than SIFT. Finally, the size of the identifiers is 64 bits. The result of experiments after all these calculations SURF is three times faster than SIFT. However, in terms of image scaling, SIFT gives better results than SURF.

In general, object detection and identification can be done by using local features in this method. However, because of the high dimensionality problem in this method, it has been tried to be prevented by certain algorithms. To avoid this problem, k-d algorithm is used which can detect neighbors by using a limited number of calculations. This search has the best matching keypoints in the test image. Then the keypoint that matches the Euclidean Distance mentioned in the introduction is marked. With this method, it increases the matching speed and efficiency in the SURF algorithm. For a single feature, this method may well ensure that the result works correctly. However, there may still be a possibility of mismatch on messy and scattered images. In order to avoid this distress, a cluster structure for scale, localization and orientation was created for correct matches. Thus, accurate and exact matches give a healthier result thanks to this cluster structure. This cluster structure is formed by generalized Hough transform. First, a least squared estimate is made for this created set structure and unrelated sets are discarded. Then, the calculations for the remaining clusters are made with the suitability and the number of mismatches. Hough Transform aims to create a realistic model, a more efficient match with Linear least squares method and finally to verify the ecosystem with Bayesian approach.

To summarize the SURF structure, in object detection and recognition process, the keypoints that it generates will be invariant against the scale, orientation, rotation and illumination factors of the image in which the object is located. Therefore, it provides a great advantage in object recognition and detection at large scales and in scattered and complex images.

Train images are all photographed from above with drone. Our object is that the cars are exactly in the middle of the train image. When we apply the SURF method to each train image, we obtain keypoints at different locations and numbers. We keep the locations of the keypoints in each train image. Since the center image of our object in the train image is 64x64 and located in the middle of the image, we have determined 32 coordinate values for the x and y directions as the center.

(a) Original Image

(b) Convert Grayscale Image

(c) Applying SURF Method to Grayscale Image

Figure 2.2: SURF method process

2.2.2 Gradient-Magnitude-Based Support Regions (GMSR)

Ünsalan did Gradient-Magnitude-Based Support Regions (GMSR) which is a support gradient extraction method [28]. In this study, a different approach has been presented in subjects such as pattern recognition and remote sensing communities. Unlike other field-based identification systems (such as Line Support Region (LSR)) it is less costly. It is faster because it requires less calculation than LSR. It can also be represented by curves rather than a straight-line approach to represent areas.

To describe the structure of GMSR, since this is a gradient-based approach, it is necessary to first find the gradient components on the image. In our study, we created the gradient components with the help of Gaussian structure. In this structure, we took the derivative of Gaussian structure. When you take the derivative, the equation came out.

$$\partial f_x = \frac{-2x}{\sigma} e^{-\frac{x^2}{\sigma}} \tag{2.2a}$$

$$\partial f_y = \frac{-2y}{\sigma} e^{-\frac{y^2}{\sigma}} \tag{2.2b}$$

The functions f_x and f_y described here are derivatives of the two-dimensional Gaussian structure with respect to x and y. In our study, instead of using the equation derived from y, we rotated the image 90 degrees and used its derived equation according to x. So, at first, we obtained the $G_x(x, y)$ component by convolving the structure of Gaussian's derivative according to x with the image, then rotating the image 90 degrees, and we obtained the $G_y(x, y)$ component by convolving the structure of Gaussian's derivative according to y with that image. Thus, we found both $G_x(x, y)$ and $G_y(x, y)$ gradient components. Then we took the square root of the squares of these structures respectively and acquired the magnitude structure. Finally, we applied a threshold value to this magnitude structure and we have GMSR fields of our object.

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad (2.3)$$

The GMSR fields in the image were found in this section. After giving a certain threshold to take advantage of these fields, these fields will be able to detect the objects in the image and form their fields. In other words, these fields will be our object identifiers.

After finding GMSR areas, we keep the coordinate values corresponding to the areas we have detected on the object. We keep the mean and variance values of the pixels corresponding to these coordinate values. These values are found in an important part to compare the mean and variance values that we will find in the test section. The areas we found can be represented mathematically in arrays. The variance and mean of a finite sequence can be represented as follows.

$$\mu = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) \quad (2.4a)$$

$$ϑ = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2 \quad (2.4b)$$

In the particular of these equations, '*n'* represents the number of elements in the array, 'μ' represents the average of the array, and ' ϑ ' represents the variance of the array.

Since the identifiers we have identified for the train section are fields, we find the coordinate values corresponding to the centers of gravity of these fields. Then we look at the two parameters between these centers of gravity and the center of the object. These two parameters are angle and distance information for each line. This angle and distance information will be required after matching process for the voting process.

In the test section, we apply the GMSR method for the test image. We store the pixel values of the fields we have obtained in arrays. We store the mean and variance values of these fields in another array variable for matching process. In the end, we will keep the centers of gravity for these areas, as it will be necessary for voting.

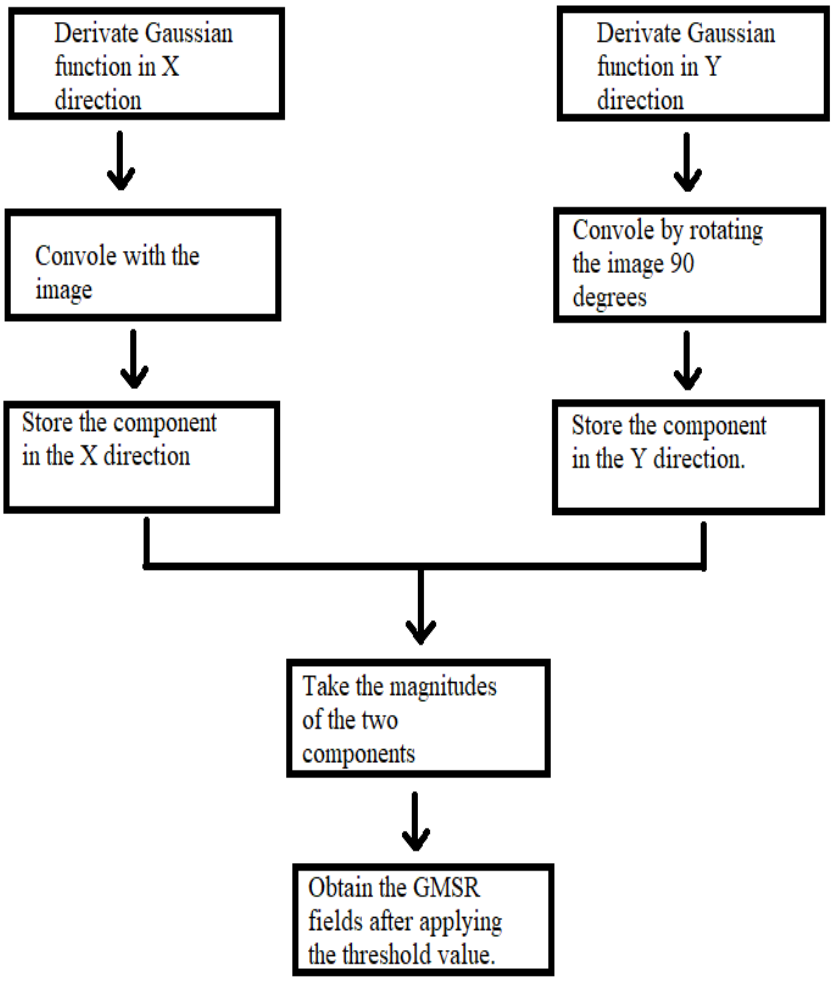
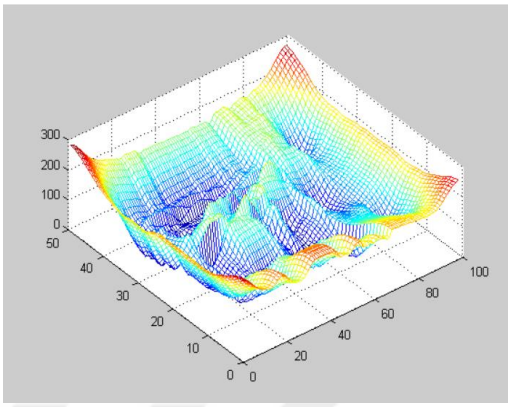
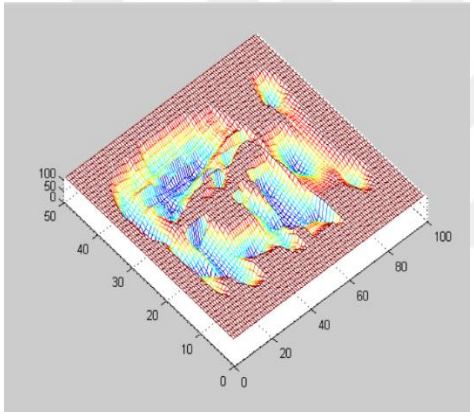


Figure 2.3: GMSR ecosystem

(a) Original image



(b) Image GMSR Region



(c) Threshold GMSR region

(d) GMSR Label

Figure 2.4: GMSR method process

2.2.3 Line Segment Detection (LSD)

Line detection studies is a very old field. Haralick's [30] work on a narrow ridge with pixel grouping is the basis of the line segmentation study. Some of the object detection and recognition studies are based on the fastest changing locations of pixels. The fastest changing pixels in images are edge regions. Likewise, line segmentation can be included in these studies. In the study conducted by Burns, Hanson and Riseman [31], line segmentation was performed via line support region. In this study, it is a segmentation performed through gradient orientation. When the other edge and line detection methods are considered, the detection process is done via gradient magnitude. On the basis of this, this position is one of the places where the gradient changes most rapidly. Therefore, we can find the gradient size of these values and the locations of rapid change with this value. However, the method proposed in this study proposes to make a segmentation by grouping the pixels over the gradient orientation.

In the Burns et al study, line support region defines straight regions that share the same picture gradient angle. Segmentation was carried out in three steps. By grouping the pixels with the same gradient angle values, dividing them into line support regions with a certain threshold value, determining the value representing the best approach for each grouped region, and finally making the segmentation relative to the grouped regions.

Although segmentation gives a good result in terms of location, the threshold value problem still exists. In the study developed by Desolneux, Moisan and Morel [32] for the solution of this problem, the control of the segmentation is controlled by false positive values. The “A contrario model” allows the segmented lines to have a higher level of accuracy.

Briefly, the line support regions method is introduced using the gradient orientation method. This method uses gradient angles unlike previous methods. In previous studies, segmentation was performed based on gradient magnitudes. Although this method performed a perfect segmentation, in some cases it made extra detection. To prevent this, there were difficulties in determining the threshold value. With the

improvement method proposed by Desolneux et al., a validation criterion was developed for this detection method.

In the train section, we apply Line Support Detection method to our images. Since these lines have a very narrow area, we provide dilation process to expand their areas. Dilation process belongs to the group of morphological operations. Morphological operations are the operations made with the shape of the object. These operations can remove noise on the image, enlarge or reduce the shape of the object, and "Isolation of individual elements and joining disparate elements in an image". Dilation is done with the aim of enlarging the object by expanding the pixels in the image with a kernel.

When we continue to train part, we keep the coordinate values corresponding to the lines we have detected on the object. We keep the mean and variance values of the pixels corresponding to these coordinate values. These values are found in an important part to compare the mean and variance values that we will find in the test section.

Since the identifiers we have identified for the train section are fields, we find the coordinate values corresponding to the centers of gravity of these fields. Then we look at the two parameters between these centers of gravity and the center of the object. These two parameters are angle and distance information for each line. This angle and distance information will be required after matching process for the voting process.

In the test section, we apply the Line Support Detection method for the test image. Since the lines we have obtained have a narrow area, we apply dilation to expand these areas. Then we store the pixel values of the fields we have obtained in arrays. We store the mean and variance values of these fields in another array variable for matching process. In the end, we will keep the centers of gravity for these areas, as it will be necessary for voting.

(a) Original image

(b) Applied LSD Method

(c) Image with LSD

(d) Dilate LSD Lines

Figure 2.5: LSD method process

2.3 OBJECT DETECTION

2.3.1 Feature Matching

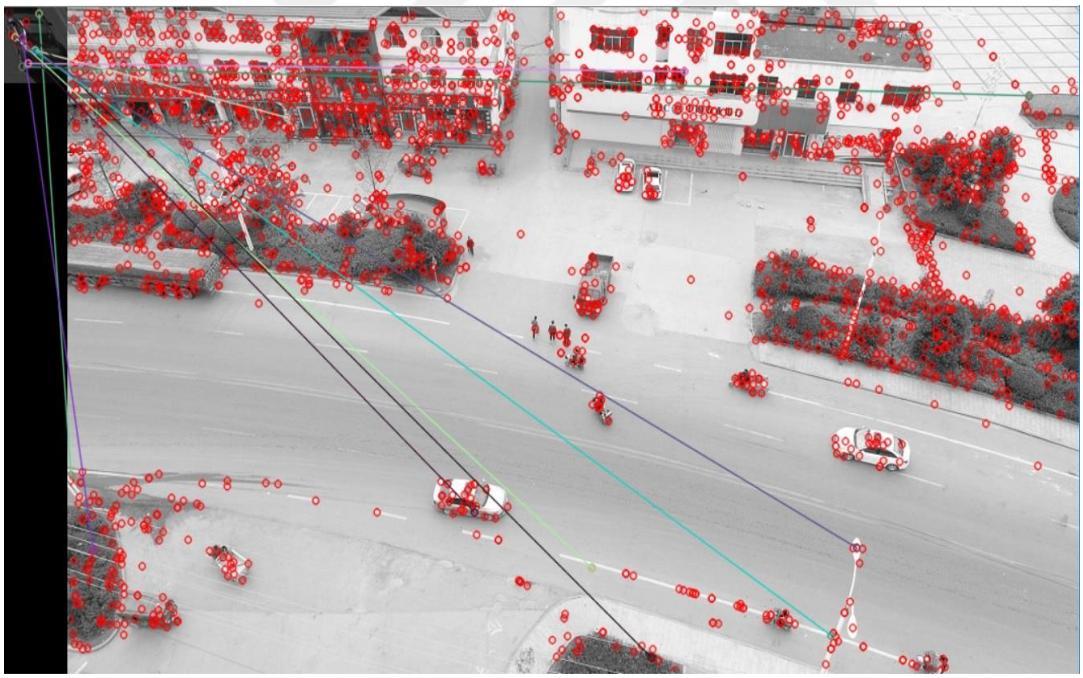
We calculate the object identifiers and descriptors we found in the test image and matching the train images' identifiers and descriptors with NormL1 method. It is minimizing the sum of the absolute differences or we can say threshold value (*S*) between the first target point (p_{1i}) and the second target point (p_{2i}):

$$\sum_{i=1}^{n} | p_{1i} - p_{2i}| < S \quad (2.5)$$

After giving NormL1 a threshold, we determined in our study, we store the matching identifiers and descriptors. Then, for each train image we first find the coordinates, centers of gravity, angle and distance and the statement values. Then we capture the features we matched via NormL1 with the values we store and vote on the test image.

We apply the SURF method to the different test image which is not include the our train images and also find keypoints for the test image. We keep the locations of the keypoints found. We apply the matching process for the keypoints we found in the test image and the keypoints we found in the train images. The pairing process is based on the NormL1 theorem. With this theorem, we mark the most similar keypoints with the help of a threshold value. We have locations of matching keypoints in the test image, angle and distance information of the keypoints of the matching train image.

(a) Original Image



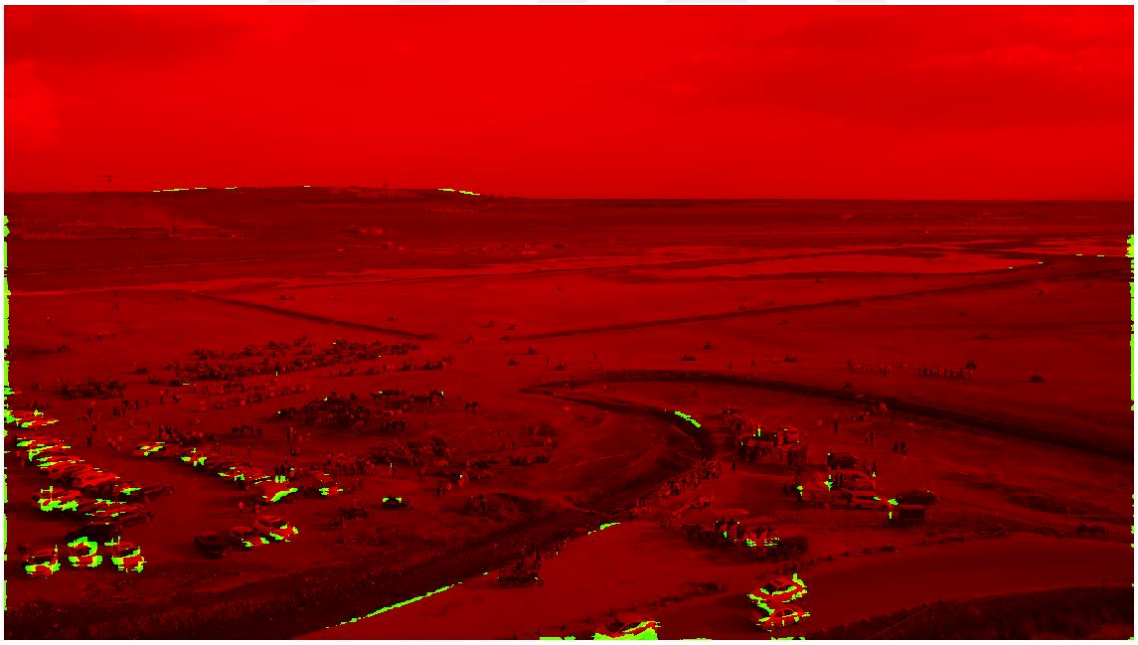
(b) SURF Matching

Figure 2.6: SURF matching process

In GMSR method, we used NormL1 which we mentioned in the previous part in the matching section of the fields obtained from both test and train images. The parameters we compared for matching these fields are mean and variance statistical values.

In light of mean and variance equations, we calculate the mean and variance of the fields in each train image and the mean and variance of the fields in the test image. We then compare these values individually with the threshold values in the NormL1. These threshold values are applied for both variance and mean values. Then the fields that pass through these thresholds are matched. We tag these matching areas in a separate sequence on both train images and test images. In this way, we have areas of matching train and test images left.

(a) Original Image



(b) GMSR Matching

Figure 2.7: GMSR matching process

In the matching process of the LSD method, we apply the same process as the GMSR.

(a) Original Image

(b) LSD Matching

Figure 2.8: LSD matching process

2.3.2 Voting Process

In this part, we explain why we should vote towards the center of the object. The reason we voted towards the center of the object, we want to create the clustering by directing the identifiers and coordinates of the detectors that we find in different train images towards the center of the object. When we look for different train images in different test images, it is very unlikely that we will naturally find all the identifiers we find in a train image in the car in the different test images. Suppose we matched an identifier of the car in the test and an identifier in the image of a train. We have already indicated that the identifier will be directed by doing the voting process. In this way, we create an accumulation in the center of our object without the need for all identifiers in all train images. We will need some parameters when doing this vote. Because if we want to move a point to another point in a two-dimensional space, we need the angle and distance parameters of that point.

When we do this, we know the coordinates we get from the features, we also know the center of the object. The rest is to calculate the angle and distance. Formulas for finding angle and distance between two points are based on basic geometry. In a two-dimensional space, the distance between two points, the square of the change in the x direction and the square of the change in the y direction, is added and the square root of this result is obtained. On the other hand, if we want to find the angle between two points in two-dimensional space, we have to divide the difference in the y direction by the difference in the x direction and take the inverse tangent of the result.

We keep the angles and distances given above. Thus, we know the angle and distance information of the features in each train image. Then we will have to find a statement that will move these points to the center. Since the coordinate points for this statement are two expressions (x and y direction), we had to formulate them with two different expressions. These expressions occur for the x direction as follows. It is calculated by multiplying the distance value and the cosine value of the angle for the x coordinate value of the specified point. Then we take the absolute value of this result. The y direction of these expressions occurs as follows. Then the statement is completed by adding or subtracting the specified point with the value in the x direction. The distance we have stored for the y coordinate value of the specified point is obtained by multiplying

the sine value of the angle. And then we take the absolute value of this result Adding or subtracting the specified point with the value in the y direction completes the statement in the y direction. We decide the addition and subtraction process here. If the x direction of the point is lower than the center value, we perform addition, and if its position is higher, we perform subtraction. This event is formulated as follows.

$$x_{1new} = x_1 \pm d_1 \times \cos(a_1) \quad (2.6a)$$

$$y_{1new} = y_1 \pm d_1 \times \sin(a_1) \quad (2.6b)$$

If we name the expressions here, x_{1new} and y_{1new} coordinate values we move to the center, x_1 and y_1 we process our train points (before moving), d_1 and a_1 of the specified point (x_1 and y_1 coordinate) are the distance and angle values we found before. As a result of this process, we see that the new points formed are accumulated in the center of the object in the train image. Voting processing in the train image is an experiment. The actual voting process is the one in the test image. We will already explain it in the following paragraphs.

In SURF method, in the light of this angle and distance information, new locations of these points will be found on the test image. Hereby voting process on the test images completes.

In the GMSR method, is to find the angle and distance values for the voting process, which is one of the important parts of the ISM structure. As can be seen from this method, these structures do not rely on a single coordinate plane, such as the SURF method. The reason for this is that these structures consist of areas. The process we apply for this is to find the centers of gravity of the areas that we obtain. The reason for this is to reduce the structure to a single coordinate value for voting and to find the angle and distance values of these structures extending towards the center of the object. After finding the centers of gravity in this way, we have already mentioned the formulas for the angle and distance values in the voting process in the first section. Likewise, after applying that process to these structures, we have stored the angle and distance

information for each area for voting. This is the process for a single train image. We applied this to the number of train images that we determined. For each train image that we have, we have stored the angle and distance values of their areas with a cellular structure. In the test image section, we applied the same procedure to a single train image. In this process, we have GMSR fields. We also kept the centers of gravity for the voting process in these areas.

In the voting process of the LSD method, we have already saved tag values for matching fields. For the matched fields in the test image, we take the angle and distance values of the matched fields in the train section. Then, in the test image, we provide the center of gravity of the matching areas in the train section to canalized the matching areas within the angle and distance information. At the end of the day, we have completed the voting process.

Finally, after the vote is completed, we have a set of coordinates. As the last process, we have to do a Gaussian fit to these coordinates. If we add this function with a Gaussian close to the center, two Gaussian peaks with a different peak value will arise. Because we vote towards the center of the object in our study, it will cause an accumulation in the center. Of course, not only in the center of the object, we have detected, but also in different places come from coordinate points. We will explain in more detail why the coordinates coming from different locations come from, but for example, because windows have the same structure as the top of the cars, they also occur there. However, one or more Gaussian structure or structures with a higher peak value will occur in the center of the object, since it has more coordinates in the center of the car and the Gaussian structure is cumulative. If we determine a threshold value for these Gaussian structures to be formed, the structures in places with one or two identical coordinate values from different locations will be cleaned. That is exactly what we expect in our study.

After the end of the voting process in the SURF method, we keep the new points on the test image. On these points, we will settle the Gaussian structures. In our method, we expect these structures to form multiple Gaussian structures on the resulting test image. When these structures are formed, there will be a very few number of new

locations created by keypoints that vote for different objects, while a large number of Gaussian structures will occur at locations in the center of the car object.

The voting process is created in the plane we specified in the ISM skeleton section for the GMSR method. Here, when we move the centers of gravity as a result of the voting process, we have new coordinates. We fit Gaussian structures at the locations of these new coordinates. All we expect here is that the areas of our object in different train images will detect areas in test images that are completely different but contain the same object. Thus, it is to create a clustering on the detected objects with the help of Gaussian structures.

After the voting process is completed in LSD method, we have to place Gaussian structures in the coordinates we have. With this process, an accumulated area will remain as a requirement of Gaussian structures.

3. RESULTS AND DISCUSSION

In the first part of this section, technical details about the study and system features are mentioned. In this section, the characteristics of test and training data sets and the properties of the computational environment are examined. In the second part, the construction of the experimental system is explained. The ground truth created in this section is created. In the same section, false positive rate and true positive rate measurements are explained.

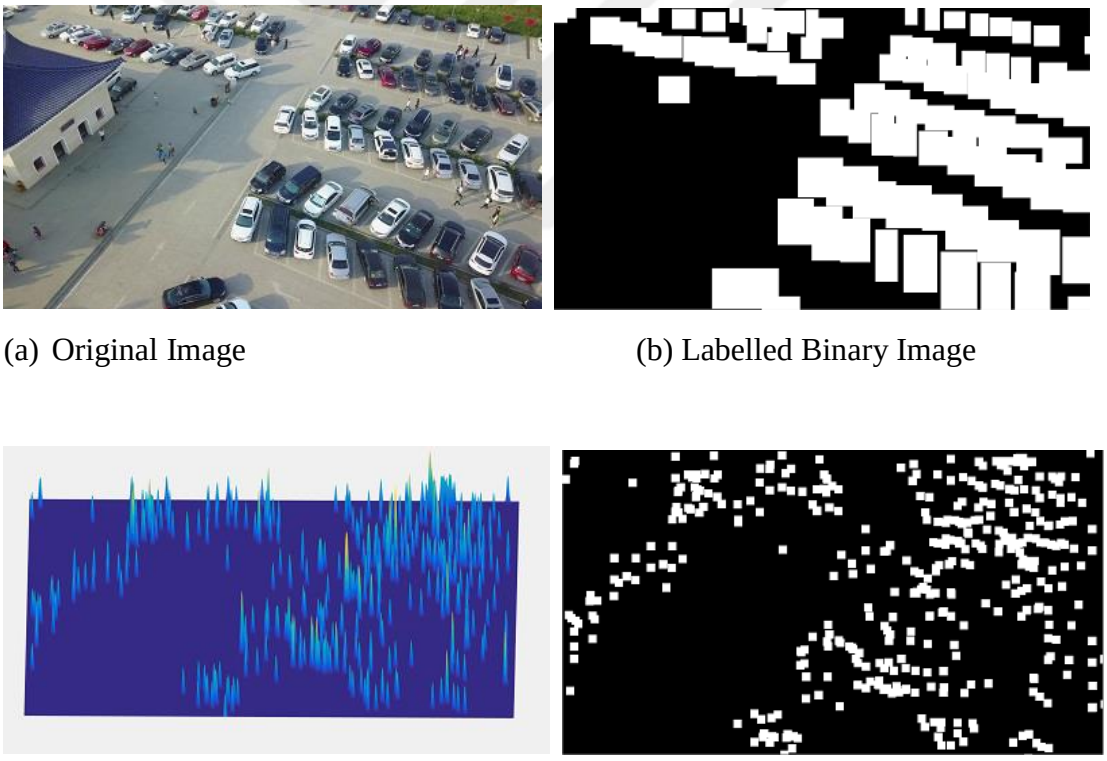
3.1 Data Sets and System Features

Test and training images related to three different methods used in the study were selected differently from each other. In other words, the objects in the training images are not included in the test images. Twenty images taken with UAV were used for training images. There is one car in each training image used. In other words, the methods were trained with a total of 20 car images. The objects in these images are placed in different positions, colors and environments. The dimensions of the training images are 64x64. On the test side, it was taken care that the images taken with UAV were found in different environments. In our study, there are a total of 3874 cars in 100 test images. The dimensions of the test images are based on 1200x675 and 900x675. Particular attention has been paid to the fact that in urban and rural areas, night and day times in time, close and distance in distance, and the use of images with more objects and less objects as the number of objects. VisDrone2018-DET-test-challenge was used as the data set used in our study.

The operations of the methods were performed in MATLAB R2016a version and using computer vision toolbox. The processor is built around Intel i7 and 8 GB RAM. AMD Radeon 530 was used as a graphics card in the study.

3.2 Experimental System

Ground truth was first established in the experimental system. In this process, "Train Image Labeler" system is used in MATLAB2016a version. By the help of this application, objects that appear in the image manually are labeled. The tagged objects are collected in a blank image and the image is reduced to a binary image. On the other hand, the top view of the regions obtained as a result of the voting process was collected in a blank image and reduced to a binary image. These two images were subjected to logic and processing. As a result of this process, the number of objects found, the fields located in the right place and the values of the fields located in the wrong place were found. The visual expression of this description is shown in Figure 3.1.



(a) Original Image (b) Labelled Binary Image

(c) Method Result (d) Binary Image of the Result

Figure 3.1: Experimental process

Pixel based measurement was performed while doing this. According to this measurement, the true positive rate (TPR) is the division of the remaining pixels by the actual reference value to the remaining pixels. On the other hand, false positive error (FPR) value was found by dividing the mismatched pixels to all pixels found in the method. The calculations were repeated around each threshold.

$$TPR = \frac{pix_{tp}}{pix_{gt}} \tag{3.1a}$$

$$FPR = \frac{pix_{fp}}{pix_{all}} \tag{3.1b}$$

In the above equations, pix_{tp} value is represented as the remaining pixels after process logic and, pix_{gt} value is labeled pixels with ground truth, pix_{fp} value is represented as mismatched pixels and pix_{all} value is represented as all pixels found in the method.

Threshold values of TPR and FPR operations are formed in 0.01 step intervals from 0 to 0.06. The values in Table 3.1 are the result of figure 3.1 (a). There are 90 car objects in this image.

Table 3.1. Test Image TPR and FPR Values

Methods	GMSR		LSD		SURF	
Threshold Values	TPR	FPR	TPR	FPR	TPR	FPR
0	1	1	1	1	1	1
0.01	0.95	0.68	0.99	0.65	0.59	0.74
0.02	0.92	0.06	0.96	0.09	0.17	0.04
0.03	0.26	0.01	0.61	0.01	0.03	0.01
0.04	0.12	0	0.32	0	0.02	0
0.05	0.02	0	0.07	0	0.01	0
0.06	0	0	0	0	0	0

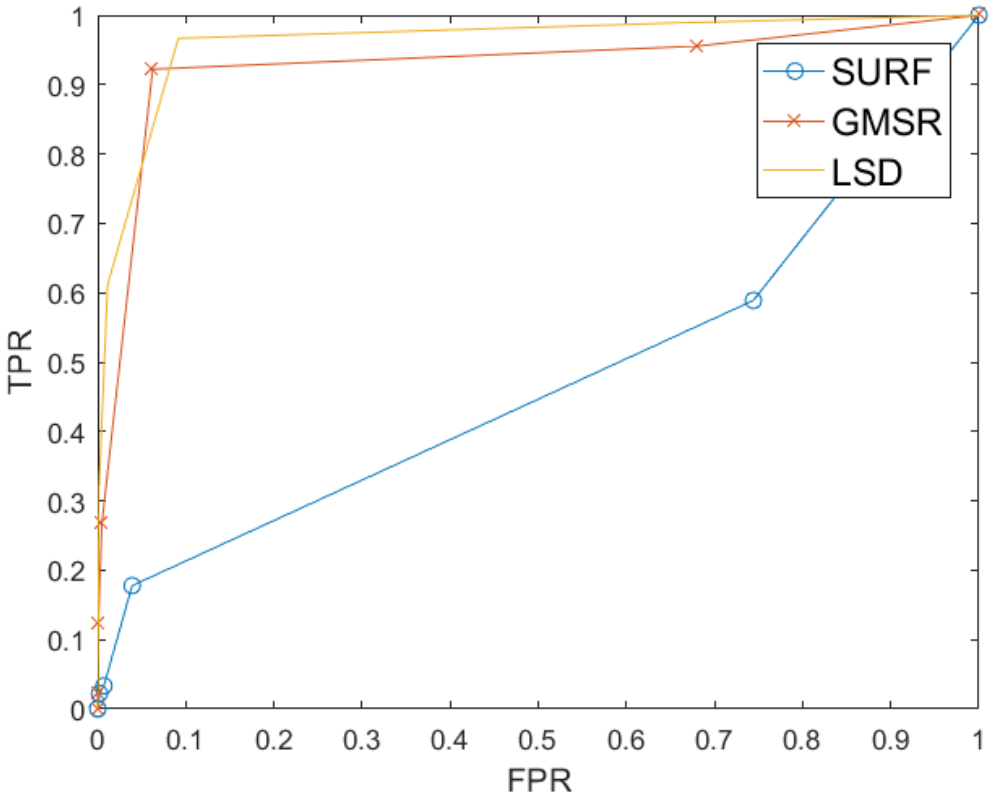


Figure 3.2: Test image ROC curve

The receiver operating characteristics (ROC) curve in figure 3.2 is based on the image in figure 3.1(a). ROC curves have an important function to give the characteristic of the methods. With these curves, it is found at which point the methods are efficient. ROC curve, TPR and FPR values give important information about the results while constructing the experimental setup. The values in this curve are based on the values in table 3.1.

In this study, the threshold value was taken as 0.01. The reason for this value is based on the results of SURF. After the threshold value of 0.01 in many test images in the applied methods, especially in the SURF method, this constant is given a more effective result than other threshold values.

An average result was found for all test images based on the study. This average result was calculated for all methods. All pixel values corresponding to the threshold of 0.01 and found as a result of the methods were taken. Since the areas which found in each test image were different, the cumulative sum of these areas and the cumulative areas of the total areas were calculated. The values corresponding to these fields obtained in the light of equations a and b were calculated. These values were obtained from three different methods in each test image. TPR and FPR values corresponding to the total test images obtained in the light of equations 3.1a and 3.1b were calculated. These values were obtained from three different methods in each test image.

Table 3.2 Test Images Average TPR and FPR Values

Methods	GMSR		LSD		SURF	
Threshold Value	TPR	FPR	TPR	FPR	TPR	FPR
0.01	0.87	0.16	0.91	0.19	0.49	0.23

The values in Table 3.2 are the average values obtained from a total of 100 test images. These average values were calculated for three different methods. For these averages, the ROC curve from the threshold of 0 to 0.06 is shown in figure 3.2.

The average processing time of these methods was calculated in unit seconds. The average GMSR method was 2.6 unit seconds, LSD 592.1 unit seconds and SURF 30.8 unit seconds.

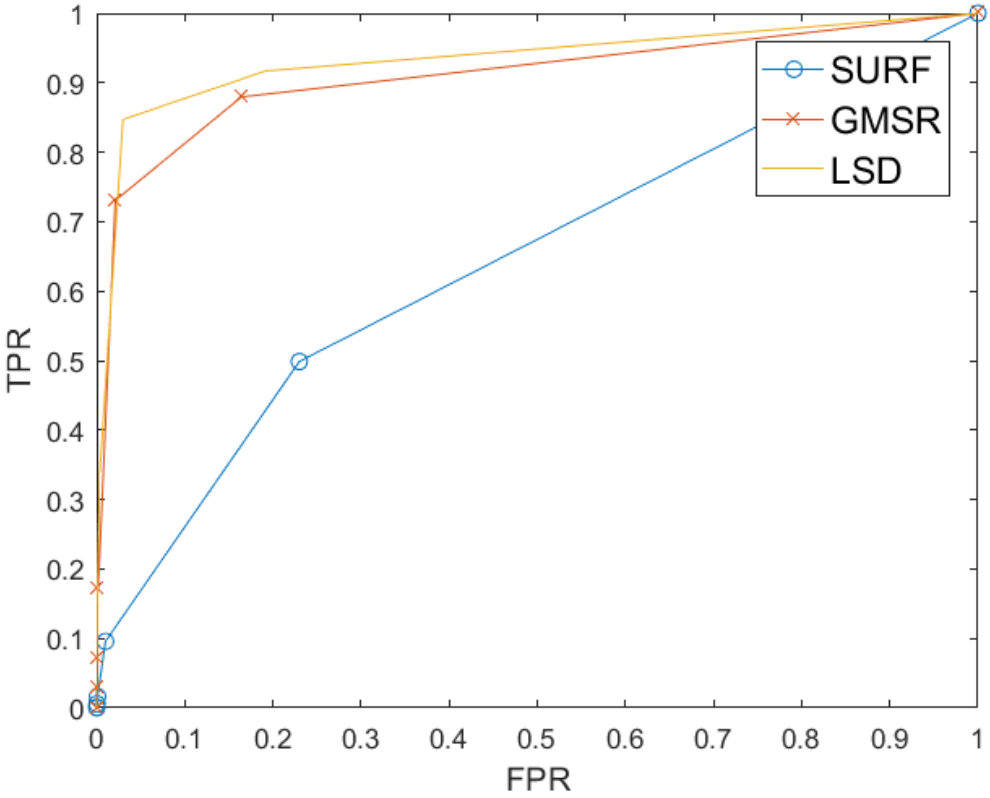


Figure 3.3 Total test images ROC curve

4. CONCLUSIONS

In this thesis, object detection and recognition is made with the help of images taken from the air. The recommended structure for this detection and recognition is ISM. The studies in this field are mentioned in the introduction part of the thesis. Here, the literature review required for this method is made. For this process, the skeleton of this thesis was created with the help of probabilistic framework topics with ISM. In summary, ISM is a method of detecting and recognition an object by pointing to the center of the object after the features of the object have been found. In the second part of the thesis, more detailed information about the method is given, object detection methods are used, feature matching method is explained and voting processing information is given. GMSR, LSD and SURF methods were used for object detection. LSD and GMSR are field-based object detection. In the third part of the thesis, the experiment set is explained and the results are discussed with tables and graphics.

One of the important issues in the study is the creation of a data set. The data set was generated with aerial images with UAV. The training images used in the system were not taken from the test images. In order to generalize the study, the test images were composed of images taken at different times, at different locations and at different distances. Secondly, the system was created via MATLAB. The system was compiled individually for each method.

If the results are examined, LSD and GMSR method at the object detection point is more successful in finding objects in the test images proportionally than SURF method. It was seen that LSD method gives less false acceptance error than GMSR method. In terms of study time, GMSR method was found to be much faster than other methods. When these results are taken into consideration, it is seen that GMSR method is the most ideal time and efficiency for object detection in aerial images.

REFERENCES

[1] Hough, P.V.C., (1959) Machine Scanning of Nuclear Emulsions and Bubble-Chamber Pictures, International Conference on High Energy Accelerators and Instrumentation, 14-19 September, Geneva, Switzerland

[2] Duda, R.O., Hart, P. E., (1972) Use of the Hough Transformation to Detect Lines and Curves in Pictures, Communications of the ACM, 11-15

[3] Bay,H,. Tuytelaars, T., and Van Gool, L., (2008) SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, 346-359

[4] Zhu, Q., Avidan, S., Yeh, M.C., Cheng K.T., (2006) Fast Human Detection Using a Cascade of Histograms of Oriented Gradients, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 17-22 June, New York, NY, USA, USA

[5] Nakajima, C., Pontil, M., Heiselec, B., Poggioc T., (2003) Full-Body Person Recognition System, Pattern Recognition, 1977–2006

[6] Wu, B., Nevatia, R., (2007) Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors, International Journal of Computer Vision, 247 – 266

[7] Druzhkov, P. N., Erukhimov, V. L., Zolotykh, N. Y., Kozinov, Kustikova, E. A., Meerov, I. B., and Polovinkin, A. N., (2011) New Object Detection Features in the OpenCV Library, Pattern Recognition and Image Analysis, 384-386

[8] Geisman, P., Schneider, G.**,** (2008) A Two-Staged Approach to Vision-Based Pedestrian Recognition Using Haar and HOG Features IEEE Intelligent Vehicles Symposium, 4-6 June, Eindhoven, Netherlands

[9] Leibe, B., Leonardis, A., Schiele, B.**,** (2004) Combined Object Categorization and Segmentation with an Implicit Shape Model, ECCV'04 Workshop on Statistical Learning in Computer Vision, 10−14 May, Prague, Czech Republic

[10] Leibe, B., Schiele, B.**,** A. Ettlin, **(**2008) Learning Semantic Object Parts for Object Categorization, Image and Vision Computing, 15-26

[11] Gall, J., Razavi, N., and Gool, L.V.**, (**2011**)** An Introduction to Random Forests for Multi-class Object Detection, 15th International Conference on Theoretical Foundations of Computer Vision: Outdoor and Large-Scale Real-World Scene Analysis, June 26 - July 1, Dagstuhl Castle, Germany

[12] Sırmaçek, B., Ünsalan, C., (2011) A Probabilistic Framework to Detect Buildings in Aerial and Satellite Images, IEEE Transactions on Geoscience and Remote Sensing, 211 - 221

[13] M., İlsever, C., Ünsalan**,** (2013) Building Detection Using HOG Descriptors, RAST'13, 12-14 June, Istanbul, Turkey

[14] Razavi, N., Gall, J., Kohli, P., Gool, L. V.**,** (2012) Latent Hough Transform for Object Detection, 12th European Conference on Computer Vision, 7-13 October, Florence, Italy

[15] Abughalieh, K.M., Sababha, B.H., Rawashdeh, N.A., (2018) A Video-Based Object Detection and Tracking System for Weight Sensitive UAVs, Springer Science and Business Media, 9149-9167

[16] Chen, X., Meng, Q., (2013) Vehicle Detection from UAVs by Using SIFT with Implicit Shape Model, IEEE International Conference on Systems, Man, and Cybernetics, 13-16 October, Manchester, UK

[17] Jüngling, K., Becker, S., Arens, M., (2011) Hierarchical Object Detection and Tracking with an Implicit Shape Model, International Conference on Image Processing, Computer Vision, and Pattern Recognition, IPCV 2011, 18 – 21 July, Las Vegas, Nevada, USA

[18] Seo, C.J., (2016) Vehicle Detection Using Images Taken by Low-Altitude Unmanned Aerial Vehicles (UAVs), Indian Journal of Science and Technology, 1–6

[19] Rematas, K., Leibe, B., (2011) Efficient Object Detection and Segmentation with a Cascaded Hough Forest ISM, IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 6-13 November, Barcelona, Spain

[20] Barinova, O., Lempitsky, V., Kohli, P., (2012) On Detection of Multiple Object Instances Using Hough Transforms, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1773 - 1784

[21] Pan, L., Chu, W.S., Saragih, J. M., Torre, F. De la, Xie, M., (2011) Fast and Robust Circular Object Detection with Probabilistic Pairwise Voting, IEEE Signal Processing Letters, 639 - 642

[22] Razzaghi, P., Samavi, S., (2014) Hierarchical Implicit Shape Modeling, Journal of Visual Communication and Image Representation, 1251-1261

[23] Guo, H., (2011) A Simple Algorithm for Fitting a Gaussian Function, IEEE Signal Processing Magazine, 134 - 137

[24] Hagen, N., Dereniak, E. L., (2009) Gaussian Profile Estimation in Two Dimensions, Applied Optics, 6842-6851

[25] Lowe, D. G., (1999) Object Recognition from Local Scale-Invariant Features, Proceedings of the Seventh IEEE International Conference on Computer Vision, 20-27 September, Kerkyra, Greece

[26] Serre T., Kouh M., Cadieu C., Knoblich U., Kreiman G., Poggio T., (2005) A Theory of Object Recognition: Computations and Circuits in the Feedforward Path of the Ventral Stream in Primate Visual Cortex. Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2005-082, Cambridge, MA

[27] Lowe, D.G., (2004) Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, 91-110

[28] Ünsalan, C., (2006) Gradient-Magnitude-Based Support Regions in Structural Land Use Classification, IEEE Geoscience and Remote Sensing Letters, 546 - 550

[29] Sarkar, S., Boyer, K. L., (1991) On Optimal Infinite Impulse Response Edge Detection Filters, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1154 - 1171

[30] Haralick, R.M., (1983) Ridges and Valleys on Digital Images, Computer Vision Graphics and Image Processing, 28-38

[31] Burns, J. B., Hanson, A. R., Riseman, E. M., (1986) Extracting Straight Lines, IEEE Transactions on Pattern Analysis and Machine Intelligence, 425 - 455

[32] Desolneux, A., Moisan, L., Morel, J.M., (2000) Meaningful Alignments, International Journal of Computer Vision, 7–23

[33] Desolneux, A., Moisan, L., Morel, J.M., (2008) From Gestalt Theory to Image Analysis a Probabilistic Approach, Interdisciplinary Applied Mathematics, 1st Edition., Springer Science - Business Media, LLC, New York, USA.

APPENDIX

APPENDIX – A: MATLAB CODES FOR GMSR METHOD

```
%-----------------------------------------------------------------------------------------
%Train Part

%Multiple images read from file
% clc;clear all;close all;

image_folder = \folder_image;
filenames = dir(fullfile(image_folder, '*.jpg'));
total_images = numel(filenames);

M_region = cell(total_images, 1) ;
M_image  = cell(total_images, 1) ;

for i = 1:total_images
   f= fullfile(image_folder, filenames(i).name);
   our_images = imread(f)  ;
   I_train = rgb2gray(our_images);
   D = padarray(I_train,[20 20],0,'both');
   [rows_test_100,columns_test_100] = size(D);
   cc = bwconncomp(D);
   M_image{i} = D;
   M_region{i} = cc ;
   x_center_train(i) = rows_test_100/2;
   y_center_train(i) = columns_test_100/2;
%     figure;imshow(D);
%     hold on;
end
```

```
X_train = [-20:1:20];
sigma_train = 100;

A_train = length(X_train);
gaussian_derivative_x_train = [];

for n=1:length(X_train)
   element_train = X_train(n);
   new_element_train = ((-2 * element_train)/sigma_train)*exp(-
(element_train^2)/sigma_train);
   gaussian_derivative_x_train = [gaussian_derivative_x_train,new_element_train];
end

gaussian_derivative_y_train = transpose(gaussian_derivative_x_train);



matrix_chy_train  = cell(total_images, 1) ;
matrix_chx_train  = cell(total_images, 1) ;
matrix_convolve   = cell(total_images, 1) ;
matrix_convolve_thresh_binary   = cell(total_images, 1) ;

for i=1:length(M_image)
   image_selection = M_image{i};
   matrix_chy_train{i} = conv2(image_selection,gaussian_derivative_y_train,'same');
   matrix_chx_train{i} = conv2(image_selection,gaussian_derivative_x_train,'same');
   [rows_train,columns_train] = size(matrix_chx_train{i});

   M_train = 0;
   for k=1:rows_train
      for j=1:columns_train
```

```
        value_train = (sqrt((matrix_chx_train{i}(k,j)^2) +
(matrix_chy_train{i}(k,j)^2)));
        M_train(k,j) = value_train;
      end
   end
   M_train_1 = M_train;
   matrix_convolve{i} = M_train_1;
   indices_train = find(M_train>120);
   M_train(indices_train) = 120;

   indices_zero_train = find(M_train<120);
   M_train(indices_zero_train) = 0;

   indices_one_train = find(M_train>0);
   M_train(indices_one_train) = 1;
   matrix_convolve_thresh_binary{i} = M_train;
end

matrix_train_component = cell(total_images, 1) ;
matrix_labeled_train   = cell(total_images, 1) ;
matrix_RGB_label_train = cell(total_images, 1) ;

for i=1:length(M_image)
   matrix_train_component{i} = bwconncomp(matrix_convolve_thresh_binary{i});
   matrix_labeled_train{i}   = labelmatrix(matrix_train_component{i});

   matrix_RGB_label_train{i} = label2rgb(matrix_labeled_train{i}, @copper, 'c',
'shuffle');
end
```

```
matrix_train_variance = cell(total_images, 1);
matrix_train_mean     = cell(total_images, 1);
for i=1:length(matrix_train_component)
   cc_train = matrix_train_component{i};
   variance_last_train = 0;
   mean_last_train = 0;
   for idx_list_train = 1:cc_train.NumObjects
      list_array_train = [];
      for idx_element_train = 1:length(cc_train.PixelIdxList{idx_list_train})
         pixel_access_train = cc_train.PixelIdxList{idx_list_train}(idx_element_train);
         grayscale_pixel_access_train = M_image{i}(pixel_access_train);
         list_array_train(idx_element_train) = [grayscale_pixel_access_train];
      end
      v_train = var(list_array_train);
      m_train = mean(list_array_train);
      variance_last_train(idx_list_train) = [v_train];
      mean_last_train(idx_list_train) = [m_train];
   end
   matrix_train_variance{i} = variance_last_train;
   matrix_train_mean{i}     = mean_last_train;
end

matrix_train_angle    = cell(total_images, 1);
matrix_train_distance = cell(total_images, 1);
matrix_train_x_center = cell(total_images, 1);
matrix_train_y_center = cell(total_images, 1);

for k =1:length(matrix_train_component)
   element_center_train = 0;
   AREA_train           = 0;
   angle_train_total    = 0;
```

```
    distance_train_total = 0;
    x_center_train_total = 0;
    y_center_train_total = 0;
    AREA_train = regionprops(matrix_train_component{k},'Centroid');
    y_center_train_region = y_center_train(k);
    x_center_train_region = x_center_train(k);
    for i = 1:length(AREA_train)
        element_center_train   = (AREA_train(i).Centroid);
        element_center_train_x = round(element_center_train(1));
        element_center_train_y = round(element_center_train(2));

        angle_train    = atan((element_center_train_y -
y_center_train_region)/(element_center_train_x - x_center_train_region));
        distance_train = sqrt(power((element_center_train_y - y_center_train_region),2) +
power((element_center_train_x - x_center_train_region),2));

        angle_train_total(i)    = angle_train;
        distance_train_total(i) = distance_train;
        x_center_train_total(i) = element_center_train_x;
        y_center_train_total(i) = element_center_train_y;
    end
    matrix_train_angle{k}    = angle_train_total;
    matrix_train_distance{k} = distance_train_total;
    matrix_train_x_center{k} = x_center_train_total;
    matrix_train_y_center{k} = y_center_train_total;
end
```

```
%-------------------------------------------------------------------------------------------------
%Test_Part

img_test = imread(\test_image);
I_test = rgb2gray(img_test);

X_test = [-20:1:20];
sigma_test = 100;

A_test = length(X_test);
gaussian_derivative_x_test = [];

for n=1:length(X_test)
   element_test = X_test(n);
   new_element_test = ((-2 * element_test)/sigma_test)*exp(-
(element_test^2)/sigma_test);
   gaussian_derivative_x_test = [gaussian_derivative_x_test,new_element_test];
end

gaussian_derivative_y_test = transpose(gaussian_derivative_x_test);

% ********************************************************
% Y Direction derivative

Chy_test = conv2(I_test,gaussian_derivative_y_test,'same');
% figure ;
% mesh(Chy_test);

% ********************************************************
% X Direction derivative
```

```
Chx_test = conv2(I_test,gaussian_derivative_x_test,'same');
% figure ;
% mesh(Chx_test);

% ************************************************************
% Sum and Magnitude Part

[rows_test,columns_test] = size(Chx_test);

for i=1:rows_test
   for j=1:columns_test
      value_test = (sqrt((Chx_test(i,j)^2) + (Chy_test(i,j)^2)));
      M_test(i,j) = value_test;
   end
end

% figure ;
% mesh(M_test);

% ************************************************************
% Cut and Create area

indices_test = find(M_test>100);
M_test(indices_test) = 100;

% figure ;
% mesh(M_test);

% ************************************************************
% Scale Between 0 and 1
```

```
indices_zero_test = find(M_test<100);
M_test(indices_zero_test) = 0;


indices_one_test = find(M_test>0);
M_test(indices_one_test) = 1;

% figure ;
% mesh(M_test);

%-------------------------------------------------------------------------------------
% Access area and Labelling

cc_test = bwconncomp(M_test);
labeled_test = labelmatrix(cc_test);

RGB_label_test = label2rgb(labeled_test, @copper, 'c', 'shuffle');
% figure;
% imshow(RGB_label_test,'InitialMagnification','fit');
% figure ;
% contour(M_test,'ShowText','on');

%-------------------------------------------------------------------------------------
% Calculate Mean and Variance

for idx_list_test = 1:cc_test.NumObjects
   list_array_test = [];
   for idx_element_test = 1:length(cc_test.PixelIdxList{idx_list_test})
      pixel_access_test = cc_test.PixelIdxList{idx_list_test}(idx_element_test);
      grayscale_pixel_access_test = I_test(pixel_access_test);
      list_array_test(idx_element_test) = [grayscale_pixel_access_test];
```

```
    end
    v_test = var(list_array_test);
    m_test = mean(list_array_test);
    variance_last_test(idx_list_test) = v_test;
    mean_last_test(idx_list_test)     = m_test;
end

%-------------------------------------------------------------------------------------------
%Compare and Matching Part

area_compare_array                = [];
match_area_patches_train_component = [];
match_area_patches_train          = [];
match_area_patches_test           = [];

for compare_index_train_component = 1:length(matrix_train_variance)
    variance_compare_train_component  =
matrix_train_variance{compare_index_train_component};
    mean_compare_train_component      =
matrix_train_mean{compare_index_train_component};
    for compare_index_train_element = 1:length(variance_compare_train_component)
        variance_compare_train =
variance_compare_train_component(compare_index_train_element);
        mean_compare_train     =
mean_compare_train_component(compare_index_train_element);
        if (variance_compare_train ~= 0 ) && (mean_compare_train ~= 0 )
          for compare_index_test = 1:length(variance_last_test)
              if (length(cc_test.PixelIdxList{compare_index_test}) < 1000)
                  variance_compare_test = variance_last_test(compare_index_test);
                  mean_compare_test     = mean_last_test(compare_index_test);
                  if (variance_compare_test ~= 0 ) && (mean_compare_test ~= 0 )
                      variance_compare_result = abs(variance_compare_test -
variance_compare_train);
```

```
                mean_compare_result     = abs(mean_compare_test -
mean_compare_train);
                if(variance_compare_result < 1250 && mean_compare_result < 5 )
                   match_area_patches_train_component =
[match_area_patches_train_component,compare_index_train_component];
                   match_area_patches_train          =
[match_area_patches_train,compare_index_train_element];
                   area_compare_array                =
[area_compare_array,compare_index_test];
                   match_area_patches_test           =
[match_area_patches_test,compare_index_test];
                end
             end
          end
       end
    end
end


pixel_list_last = [] ;
for i = 1:length(area_compare_array)
   value = area_compare_array(i);
   for j = 1:length(cc_test.PixelIdxList{value})
      element = cc_test.PixelIdxList{value}(j);
      pixel_list_last = [pixel_list_last,element];
   end
end
zeros_image  = zeros(rows_test,columns_test); %initialize
for i = 1:length(pixel_list_last)
   final = pixel_list_last(i);
   zeros_image(final) = 255;
end
```

```
blending_image =
imfuse(I_test,zeros_image,'falsecolor','Scaling','joint','ColorChannels',[1 2 0]);
figure, imshow(blending_image);

%-------------------------------------------------------------------------------------
%Gaussian Part

AREA_test = regionprops(cc_test,'Centroid');

for i = 1:cc_test.NumObjects
        element_center_test = (AREA_test(i).Centroid);
   element_center_test_x = round(element_center_test(1));
   element_center_test_y = round(element_center_test(2));
   x_center_test(i) = element_center_test_x;
   y_center_test(i) = element_center_test_y;
end

for i = 1:length(match_area_patches_train_component)

   member_train_component = match_area_patches_train_component(i);
   member_train          = match_area_patches_train(i);
   member_test           = match_area_patches_test(i);

   angle_cos_sin_last           =
matrix_train_angle{member_train_component,1}(member_train);
   distance_train_last          =
matrix_train_distance{member_train_component,1}(member_train);
   matrix_train_x_center_element  =
matrix_train_x_center{member_train_component,1}(member_train);
   matrix_train_y_center_element  =
matrix_train_y_center{member_train_component,1}(member_train);

   x_center_train_component = x_center_train(member_train_component);
```

```
    y_center_train_component = y_center_train(member_train_component);


    if (matrix_train_x_center_element > x_center_train_component &&
matrix_train_y_center_element < y_center_train_component)
        x_gaussian_member = x_center_test(member_test) -
abs(distance_train_last*cos(angle_cos_sin_last));
        y_gaussian_member = y_center_test(member_test) +
abs(distance_train_last*sin(angle_cos_sin_last));
    elseif (matrix_train_x_center_element < x_center_train_component &&
matrix_train_y_center_element < y_center_train_component)
        x_gaussian_member = x_center_test(member_test) +
abs(distance_train_last*cos(angle_cos_sin_last));
        y_gaussian_member = y_center_test(member_test) +
abs(distance_train_last*sin(angle_cos_sin_last));
    elseif (matrix_train_x_center_element < x_center_train_component &&
matrix_train_y_center_element > y_center_train_component)
        x_gaussian_member = x_center_test(member_test) +
abs(distance_train_last*cos(angle_cos_sin_last));
        y_gaussian_member = y_center_test(member_test) -
abs(distance_train_last*sin(angle_cos_sin_last));
    else
        x_gaussian_member = x_center_test(member_test) -
abs(distance_train_last*cos(angle_cos_sin_last));
        y_gaussian_member = y_center_test(member_test) -
abs(distance_train_last*sin(angle_cos_sin_last));
    end
    x_gaussian_point(i) = x_gaussian_member;
    y_gaussian_point(i) = y_gaussian_member;
end


[rows_test_100,columns_test_100] = size(I_test);
zeros_image  = zeros(rows_test_100,columns_test_100); %initialize
A = x_gaussian_point;
B = y_gaussian_point;
```

```
for j = 1:length(A)
    point_x = A(j);
    point_y = B(j);
    mu = [point_x point_y];
    Sigma = [10 1; 1 10];
    x1 = -10+point_x:1:point_x+10; x2 = -10+point_y:1:10+point_y;
    [X1,X2] = meshgrid(x1,x2);
    F = mvnpdf([X1(:) X2(:)],mu,Sigma);
    F = reshape(F,length(x2),length(x1));

    for i=1:length(F)
        for k=1:length(F)
            if((int16(point_x - 10 + i) > 0 && int16(point_x - 10 + i) < columns_test_100)
&& (int16(point_y - 10 + k) > 0 && int16(point_y - 10 + k) < rows_test_100))
                zeros_image( int16(point_y - 10 + k),int16(point_x - 10 + i)) =
zeros_image(int16(point_y - 10 + k),int16(point_x - 10 + i)) + F(i,k);
            end
        end
    end
end
```

APPENDIX – B: MATLAB CODES FOR SURF METHOD

```
%-------------------------------------------------------------------------------------
%Test Part
image_test = imread(\image_test);
image_test_grayscale = rgb2gray(‘\image_test’);
points_test = detectSURFFeatures(image_test_grayscale);
[f_test,vpts_test] = extractFeatures(image_test_grayscale,points_test);


%-------------------------------------------------------------------------------------
%Train Part

image_folder = ' image_train_folder’;
filenames = dir(fullfile(image_folder, '*.jpg'));
total_images = numel(filenames);

M_region = cell(total_images, 1) ;
M_image  = cell(total_images, 1) ;
k = 1;
for i = 1:total_images
   f= fullfile(image_folder, filenames(i).name);
   our_images = imread(f)  ;
   I_train = rgb2gray(our_images);
   [rows_train,columns_train] = size(I_train);
   points_train = detectSURFFeatures(I_train);

   [f_train,vpts_train] = extractFeatures(I_train,points_train);
   indexPairs = matchFeatures(f_test,f_train) ;
```

```
    if size(indexPairs(:,1)) > 0.5
        index_train{k} = i;
        columns_train_total{k} = columns_train;
        rows_train_total{k} = rows_train;
        indexPairsTotal{k} = indexPairs;
        matchedPoints_test = vpts_test(indexPairs(:,1));
        matchedPoints_train = vpts_train(indexPairs(:,2));
        matchedPoints_test_total_location{k} = matchedPoints_test.Location;
        matchedPoints_train_total_location{k} = matchedPoints_train.Location;


showMatchedFeatures(image_test_grayscale,I_train,matchedPoints_test,matchedPoints
_train);
        legend('matched points 1','matched points 2');
        k = k +1 ;
%         pause(0.75);
    end
end
angle_train_total    = cell(length(index_train), 1);
distance_train_total = cell(length(index_train), 1);

for i = 1:length(index_train)

    index_train_element = index_train{i};
    element_train_center_x = rows_train_total{i};
    element_train_center_y = columns_train_total{i}
    angel_train_array = [];
    distance_train_array = [];
    angle_train = 0;
    distance_train = 0;
    for k = 1:size(matchedPoints_train_total_location{i})
```

```
        matchedPoints_train_total_location_element_x =
matchedPoints_train_total_location{i}(k,1);

        matchedPoints_train_total_location_element_y =
matchedPoints_train_total_location{i}(k,2);


        angle_train    = atan((matchedPoints_train_total_location_element_y -
element_train_center_y)/(matchedPoints_train_total_location_element_y -
element_train_center_x));

        distance_train = sqrt(power((matchedPoints_train_total_location_element_y -
element_train_center_y),2) + power((matchedPoints_train_total_location_element_y -
element_train_center_x),2));


        angel_train_array(k) = angle_train;

        distance_train_array(k) = distance_train;

    end


    angle_train_total{i}    = angel_train_array;

    distance_train_total{i} = distance_train_array;

end

for i = 1:length(columns_train_total)


    columns_train_total_element = columns_train_total{i};

    rows_train_total_element    = rows_train_total{i};


    x_gaussian_point = [];

    y_gaussian_point = [];

    k = 1;
```

```
%-------------------------------------------------------------------------------------

%Compare and Matching Part


   for k = 1:length(distance_train_total{i})


       matchedPoints_train_total_location_element_x =
matchedPoints_train_total_location{i}(k,1);

       matchedPoints_train_total_location_element_y =
matchedPoints_train_total_location{i}(k,2);


       matchedPoints_test_total_location_element_x =
matchedPoints_test_total_location{i}(k,1);

       matchedPoints_test_total_location_element_y =
matchedPoints_test_total_location{i}(k,2);


       distance_train_total_element = distance_train_total{i}(k);

       angle_train_total_element    = angle_train_total{i}(k);


       if (matchedPoints_train_total_location_element_x > rows_train_total_element &&
matchedPoints_train_total_location_element_y < columns_train_total_element)

           x_gaussian_member = matchedPoints_test_total_location_element_x -
abs(distance_train_total_element*cos(angle_train_total_element));

           y_gaussian_member = matchedPoints_test_total_location_element_y +
abs(distance_train_total_element*sin(angle_train_total_element));

       elseif (matchedPoints_train_total_location_element_x < rows_train_total_element
&& matchedPoints_train_total_location_element_y < columns_train_total_element)

           x_gaussian_member = matchedPoints_test_total_location_element_x +
abs(distance_train_total_element*cos(angle_train_total_element));

           y_gaussian_member = matchedPoints_test_total_location_element_y +
abs(distance_train_total_element*sin(angle_train_total_element));

       elseif (matchedPoints_train_total_location_element_x < rows_train_total_element
&& matchedPoints_train_total_location_element_y > columns_train_total_element)

           x_gaussian_member = matchedPoints_test_total_location_element_x +
abs(distance_train_total_element*cos(angle_train_total_element));
```

```
        y_gaussian_member = matchedPoints_test_total_location_element_y -
abs(distance_train_total_element*sin(angle_train_total_element));
    elseif (matchedPoints_train_total_location_element_x > rows_train_total_element
&& matchedPoints_train_total_location_element_y > columns_train_total_element)
        x_gaussian_member = matchedPoints_test_total_location_element_x -
abs(distance_train_total_element*cos(angle_train_total_element));
        y_gaussian_member = matchedPoints_test_total_location_element_y -
abs(distance_train_total_element*sin(angle_train_total_element));
    end


    x_gaussian_point(k) = x_gaussian_member;
    y_gaussian_point(k) = y_gaussian_member;
  end
  x_gaussian_point_total{i} = x_gaussian_point;
  y_gaussian_point_total{i} = y_gaussian_point;
end


x_gaussian_point_total_last = [];
y_gaussian_point_total_last = [];



for i=1:length(x_gaussian_point_total)
  x_gaussian_point_total_last =
[x_gaussian_point_total_last,x_gaussian_point_total{i}];
  y_gaussian_point_total_last =
[y_gaussian_point_total_last,y_gaussian_point_total{i}];
end
[rows_test_100,columns_test_100] = size(image_test_grayscale);
zeros_image  = zeros(rows_test_100,columns_test_100); %initialize
A = x_gaussian_point_total_last;
B = y_gaussian_point_total_last;
```

```
%-------------------------------------------------------------------------------------
%Gaussian Part


for j = 1:length(A)
   point_x = A(j);
   point_y = B(j);
   mu = [point_x point_y];
   Sigma = [20 10; 10 20];
   x1 = -10+point_x:1:point_x+10; x2 = -10+point_y:1:10+point_y;
   [X1,X2] = meshgrid(x1,x2);
   F = mvnpdf([X1(:) X2(:)],mu,Sigma);
   F = reshape(F,length(x2),length(x1));


   for i=1:length(F)
      for k=1:length(F)
         if((int16(point_x - 10 + i) > 0 && int16(point_x - 10 + i) < columns_test_100)
&& (int16(point_y - 10 + k) > 0 && int16(point_y - 10 + k) < rows_test_100))
            zeros_image( int16(point_y - 10 + k),int16(point_x - 10 + i)) =
zeros_image(int16(point_y - 10 + k),int16(point_x - 10 + i)) + F(i,k);
         end
      end
   end
end
figure;
% hs=surf(U,V,Z);
hs=surf(zeros_image);
% grid off
% axis off
view(0,90);
set(hs,'LineStyle','none');
xlabel('X');
```

```
ylabel('Y');
zlabel('Z');
set(gca,'XAxisLocation','top','YAxisLocation','left','ydir','reverse');
% colorbar;
% axis([0 rows_test_100 0 columns_test_100])
hs.EdgeColor='red';
```

APPENDIX – C: MATLAB CODES FOR LSD METHOD

```
%-------------------------------------------------------------------------------------
%  LSD algorithm with MATLAB
%Test Part
img_test = imread('./images/test_100.jpg');
img_test_grayscale = rgb2gray(img_test);
zeros_image = zeros(size(img_test_grayscale, 1), size(img_test_grayscale, 2));
Fig = imshow(zeros_image);
%% get the start_points and end_points of each straight line use LSD.
% note: input parameter is the path of image, use '/' as file separator.
lines = lsd('./images/test_100.jpg');
% %% plot the lines.
hold on;
set(gca,'XAxisLocation','top','YAxisLocation','left','ydir','reverse');

for i = 1:size(lines, 2)
   plot(lines(1:2, i), lines(3:4, i),'linewidth', lines(5, i),'Color', [1, 1, 1]);
end
hold off;

zeros_image_test = zeros(size(img_test_grayscale, 1), size(img_test_grayscale, 2));
Lsd = imshow(zeros_image_test);
hold on;
for i = 1:size(lines, 2)
   plot(lines(1:2, i), lines(3:4, i),'linewidth', 2,'Color', [1, 1, 1]);
   F = getframe ;
   % save the image:
   save_file_name = strcat(\somewhere_folder_path');
   imwrite(F.cdata, \somewhere_folder_path')
   img_test_lsd_patch = imread('\somewhere_folder_path');
```

```
    img_test_lsd_patch_grayscale = rgb2gray(img_test_lsd_patch);

    [y, x] = find( img_test_lsd_patch_grayscale > 225);
    line_x_pixel{i}= x;
    line_y_pixel{i}= y;

    plot(lines(1:2, i), lines(3:4, i),'linewidth', 2,'Color', [0, 0, 0]);
end

%Mean and Variance Values Part

for i_1 = 1:length(line_x_pixel)
    line_x_pixel_element = line_x_pixel{1,i_1};
    line_y_pixel_element = line_y_pixel{1,i_1};
    img_test_grayscale_lsd_value = [];
    for j = 1:length(line_x_pixel_element)
        line_x_pixel_element_value = line_x_pixel_element(j);
        line_y_pixel_element_value = line_y_pixel_element(j);
        test_lsd_value =
img_test_grayscale(line_y_pixel_element_value,line_x_pixel_element_value);
        img_test_grayscale_lsd_value(j) = test_lsd_value;
    end
    img_test_grayscale_lsd_array{i_1} = img_test_grayscale_lsd_value;
    variance_test_total(i_1) = var(img_test_grayscale_lsd_value);
    mean_test_total(i_1) = mean(img_test_grayscale_lsd_value);
    lines_test_gravity_x_total{i_1} =
line_x_pixel{1,i_1}(round(COG(line_x_pixel{1,i_1})));
    lines_test_gravity_y_total{i_1} =
line_y_pixel{1,i_1}(round(COG(line_y_pixel{1,i_1})));
end
```

```
%-------------------------------------------------------------------------------------
%Train Part
image_folder = './train_images';
filenames = dir(fullfile(image_folder, '*.jpg'));
total_images = numel(filenames);
for i = 1:total_images
   f= fullfile(image_folder, filenames(i).name);
   our_images = imread(f)  ;
   I_train = rgb2gray(our_images);
   lines_train = lsd(f);
   zeros_image = zeros(size(I_train, 1), size(I_train, 2));
   figure,imshow(zeros_image);
   hold on;
   for j = 1:size(lines_train, 2)
      plot(lines_train(1:2, j), lines_train(3:4, j), 'linewidth', 2, 'Color', [1, 1, 1]);
      F = getframe ;
      % save the image:
      save_file_name = strcat('\somewhere_folder_path');
      imwrite(F.cdata, \somewhere_folder_path')
      img_train_lsd_patch = imread('\somewhere_folder_path');
      img_train_lsd_patch_grayscale = rgb2gray(img_train_lsd_patch);

      [y, x] = find( img_train_lsd_patch_grayscale > 225);
      %     disp(y);
      line_train_x_pixel{j}= x;
      line_train_y_pixel{j}= y;
      plot(lines_train(1:2, j), lines_train(3:4, j),'linewidth', 2,'Color', [0, 0, 0]);
   end
   lines_train_total{i} = lines_train;
   lines_train_x_pixel_total{i} = line_train_x_pixel;
   lines_train_y_pixel_total{i} = line_train_y_pixel;
```

```
   grayscale_train_images_total{i} = I_train;
end



for i = 1:length(lines_train_x_pixel_total)
   grayscale_image_element = grayscale_train_images_total{i};
   [cols_train_1,rows_train_1] = size(grayscale_image_element);
   lines_train_x_pixel_total_element = lines_train_x_pixel_total{1,i};
   lines_train_y_pixel_total_element = lines_train_y_pixel_total{1,i};
   for j = 1:length(lines_train_x_pixel_total_element)
      lines_train_x_pixel_total_element_2 = lines_train_x_pixel_total_element{1,j};
      lines_train_y_pixel_total_element_2 = lines_train_y_pixel_total_element{1,j};
      lines_train_lsd_value = 0;
      for k = 1:length(lines_train_x_pixel_total_element_2)
         lines_train_x_pixel_total_element_3 = lines_train_x_pixel_total_element_2(k);
         lines_train_y_pixel_total_element_3 = lines_train_y_pixel_total_element_2(k);
         lsd_values =
double(grayscale_image_element(lines_train_y_pixel_total_element_3,lines_train_x_pi
xel_total_element_3));
         lines_train_lsd_value(k) = lsd_values;
      end
      images_lines_train_lsd_value{j} = lines_train_lsd_value;
      lines_train_gravity_x{j} =
lines_train_x_pixel_total_element_2(round(COG(lines_train_x_pixel_total_element_2)
));
      lines_train_gravity_y{j} =
lines_train_y_pixel_total_element_2(round(COG(lines_train_y_pixel_total_element_2)
));
      distance_train{j}       = sqrt(power((lines_train_gravity_x{j} - rows_train_1),2) +
power((lines_train_gravity_y{j} - cols_train_1),2));
      angle_train{j}          = atan((lines_train_gravity_y{j} -
cols_train_1)/(lines_train_gravity_x{j} - rows_train_1));
      variance_lines{j}       = var(lines_train_lsd_value);
      mean_lines{j}           = mean(lines_train_lsd_value);
```

```
    end
    lines_train_gravity_x_total{i} = lines_train_gravity_x;
    lines_train_gravity_y_total{i} = lines_train_gravity_y;
    variance_train_total{i}        = variance_lines;
    mean_train_total{i}            = mean_lines;
    distance_train_total{i}        = distance_train;
    angle_train_total{i}           = angle_train;
end

%-------------------------------------------------------------------------------------
%Compare and Matching Part

area_compare_array                = [];
match_area_patches_train_component  = [];
match_area_patches_train            = [];
match_area_patches_test             = [];
for compare_index_train_component = 1:length(variance_train_total)
    variance_compare_train_component  =
variance_train_total{compare_index_train_component};
    mean_compare_train_component      =
mean_train_total{compare_index_train_component};
    for compare_index_train_element = 1:length(variance_compare_train_component)
        variance_compare_train_element =
variance_compare_train_component{compare_index_train_element};
        mean_compare_train_element     =
mean_compare_train_component{compare_index_train_element};
        for compare_index_test = 1:length(variance_test_total)
            if(length(line_x_pixel{1,compare_index_test}) < 50)
                variance_compare_test = variance_test_total(compare_index_test);
                mean_compare_test     = mean_test_total(compare_index_test);
                variance_compare_result = abs(variance_compare_test -
variance_compare_train_element);
```

```
            mean_compare_result     = abs(mean_compare_test -
mean_compare_train_element);
            if(variance_compare_result < 100 && mean_compare_result < 5 )
                match_area_patches_train_component =
[match_area_patches_train_component,compare_index_train_component];
                match_area_patches_train          =
[match_area_patches_train,compare_index_train_element];
                area_compare_array                =
[area_compare_array,compare_index_test];
                match_area_patches_test           =
[match_area_patches_test,compare_index_test];


            end
          end
       end
   end
end
figure,imshow(img_test_grayscale);
hold on;
set(gca,'XAxisLocation','top','YAxisLocation','left','ydir','reverse');
for i = 1:length(match_area_patches_test)
    element = match_area_patches_test(i);
%    hold on;
   plot(lines(1:2, element), lines(3:4, element),'linewidth', lines(5, element),'Color', [1,
1, 1]);
%    pause(20);
End
```

```
%-------------------------------------------------------------------------------------
%Gaussian Part
for i = 1:length(match_area_patches_train_component)
   member_train_component = match_area_patches_train_component(i);
   member_train           = match_area_patches_train(i);
   member_test            = match_area_patches_test(i);
   angle_cos_sin_last           =
angle_train_total{1,member_train_component}{1,member_train};
   distance_train_last          =
distance_train_total{1,member_train_component}{1,member_train};
   matrix_train_x_center_element  =
lines_train_gravity_x_total{1,member_train_component}{1,member_train};
   matrix_train_y_center_element  =
lines_train_gravity_y_total{1,member_train_component}{1,member_train};
   grayscale_image_element =
grayscale_train_images_total{member_train_component};
   [y_center_train_component,x_center_train_component] =
size(grayscale_image_element);
   if (matrix_train_x_center_element > x_center_train_component &&
matrix_train_y_center_element < y_center_train_component)
       x_gaussian_member = lines_test_gravity_x_total{member_test} -
abs(distance_train_last*cos(angle_cos_sin_last));
       y_gaussian_member = lines_test_gravity_y_total{member_test} +
abs(distance_train_last*sin(angle_cos_sin_last));
   elseif (matrix_train_x_center_element < x_center_train_component &&
matrix_train_y_center_element < y_center_train_component)
       x_gaussian_member = lines_test_gravity_x_total{member_test} +
abs(distance_train_last*cos(angle_cos_sin_last));
       y_gaussian_member = lines_test_gravity_y_total{member_test} +
abs(distance_train_last*sin(angle_cos_sin_last));
   elseif (matrix_train_x_center_element < x_center_train_component &&
matrix_train_y_center_element > y_center_train_component)
       x_gaussian_member = lines_test_gravity_x_total{member_test} +
abs(distance_train_last*cos(angle_cos_sin_last));
       y_gaussian_member = lines_test_gravity_y_total{member_test} -
abs(distance_train_last*sin(angle_cos_sin_last));
```

```
    else
        x_gaussian_member = lines_test_gravity_x_total{member_test} -
abs(distance_train_last*cos(angle_cos_sin_last));
        y_gaussian_member = lines_test_gravity_y_total{member_test} -
abs(distance_train_last*sin(angle_cos_sin_last));
    end
    x_gaussian_point(i) = x_gaussian_member;
    y_gaussian_point(i) = y_gaussian_member;
end
[rows_test_100,columns_test_100] = size(img_test_grayscale);
zeros_image  = zeros(rows_test_100,columns_test_100); %initialize
A = x_gaussian_point;
B = y_gaussian_point;
for j = 1:length(A)
    point_x = A(j);
    point_y = B(j);
    mu = [point_x point_y];
    Sigma = [10 1; 1 10];
    x1 = -10+point_x:1:point_x+10; x2 = -10+point_y:1:10+point_y;
    [X1,X2] = meshgrid(x1,x2);
    F = mvnpdf([X1(:) X2(:)],mu,Sigma);
    F = reshape(F,length(x2),length(x1));
    for i=1:length(F)
        for k=1:length(F)
            if((int16(point_x - 10 + i) > 0 && int16(point_x - 10 + i) < columns_test_100)
&& (int16(point_y - 10 + k) > 0 && int16(point_y - 10 + k) < rows_test_100))
                zeros_image( int16(point_y - 10 + k),int16(point_x - 10 + i)) =
zeros_image(int16(point_y - 10 + k),int16(point_x - 10 + i)) + F(i,k);
            end
        end
    end
end
```

CURRICULUM VITAE

Name Surname : İbrahim Eren POSTACI

Birth of Place and Time: ŞANLIURFA 04.02.1995

Language : English

Email : ierenpostaci@gmail.com

EDUCATION

Marmara University Electrical and Electronics Engineering (M.Sc.)

- Expecting to graduate in January 2020
- Research area: Image Processing, Machine Learning, 3-D Image Processing, Computer Vision, Medical Imaging
- Current GPA: 3.21/4.00
- Thesis subject: Object Detection and Recognition with Unmanned Aerial Vehicle (Advisor: Prof. Dr. Cem Ünsalan)

Gebze Technical University Electronics Engineering (B.Sc.)

- Graduated with 3.00/4.00 GPA in June 2017.
- Thesis subject: Remote Controlled Circuit Design For Intelligent Home System, Advisor: Dr. Önder Şuvak)

WORK EXPERIENCE

- Working as R&D engineer in Strategic Innovative Initiatives for software and hardware parts. I am interested in Computer Vision projects, IoT applications and Front-End process.
- Worked as intern in Türk Telekom 3rd Regional Directorate in İstanbul. I have gained experience with the technology involved in the communication sensor, such as the work of the bit data centre and the transfer office. (2016)
- Worked as intern in TUBITAK National Metrology Institute Impedance Laboratory in Kocaeli.Here we measure that stable and sensitive manner, the component of resistance, capacitance and inductance, which we call impedance. (2016)

- Worked as intern in TUBITAK National Metrology Institute Impedance Laboratory in Kocaeli.Here we measure that stable and sensitive manner, the component of resistance, capacitance and inductance, which we call impedance. (2016)
- Worked as intern in Vito Electric in İstanbul. I worked in electric design, wall washer structures, test sections of energy saving bulbs. (2014)
- Worked as intern in Sami Automation in İstanbul. I have experienced for control circuit design and software, remote operations and power electronics.

SOFTWARE

- Python 2.7 / 3.2 (OpenCv)
- MATLAB
- C/C++
- Micro Code Studio
- VHDL
- Proteus, LT Spice, NI Multisim
- NI LabView
- Cisco Packet Teaser

COURSES AND CERTIFICATES

- C & SYSTEM PROGRAMMERS ASSOCIATION
  - C (February 2019)

Marmara Üniversitesi
Fen Bilimleri Enstitüsü
Yönetim Kurulu'nun 15.01.2020
Tarihli Tutanak Örneğidir

KARAR NO: 2020/02-36:

Enstitümüz Elektrik-Elektronik Mühendisliği (İngilizce) Anabilim Dalı 525017017 numaralı Yüksek Lisans öğrencisi İbrahim Eren POSTACI'nın "Object Detection and Recognition with Unmanned Aerial Vehicle" başlıklı tezinin Tez Savunma Sınav Jürisi'nin aşağıda belirtildiği şekilde kabulüne Yönetim Kurulumuzca oybirliği ile karar verildi.

Prof. Dr. Cem ÜNSALAN	(Danışman)
Doç. Dr. Engin MAŞAZADE	(Üye)
Doç. Dr. Serkan TOPALOĞLU	(Üye, Yeditepe Üniv.)
Dr. Öğr. Üyesi Salih BAYAR	(Yedek Üye)
Prof. Dr. Duygun Erol BARKANA	(Yedek Üye, Yeditepe Üniv.)

ASLININ AYNIDIR

Enstitü Sekreteri
Sevim BEKTAŞ