

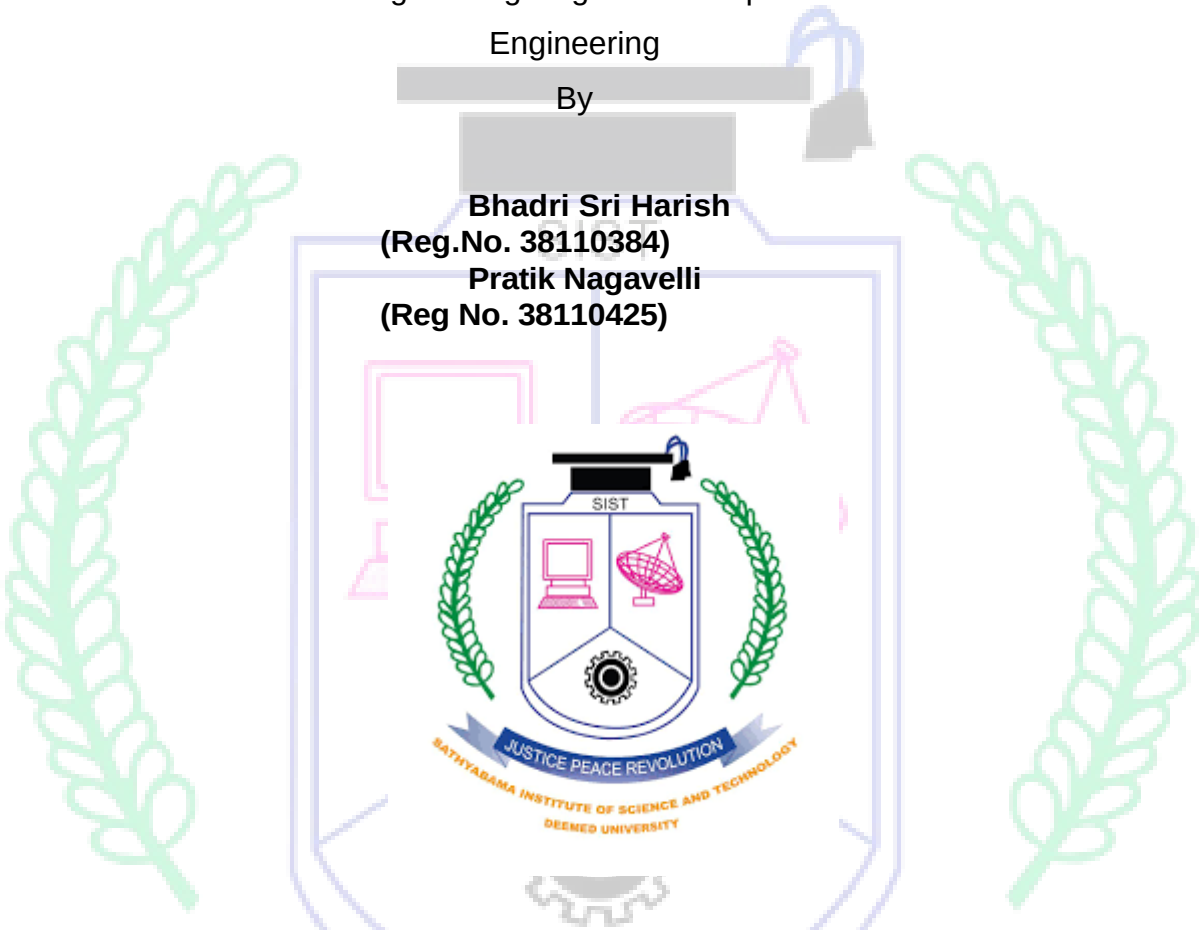
Image Recognition Using SURF Algorithm

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering Degree in Computer Science and

Engineering

By

Bhadri Sri Harish
(Reg.No. 38110384)
Pratik Nagavelli
(Reg No. 38110425)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

MARCH 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Bhadri Sri Harish (38110384) and Pratik Nagavelli (38110425)** who carried out the project entitled "**Image Recognition Using SURF Algorithm**" under my supervision from December 2021 to March 2022.

Internal Guide:
Dr. P.SardarMaran, M.E., PhD

Head of the Department:

Dr.L. Lakshmanan M.E., Ph.D.

Dr.VIGNESHWARI, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

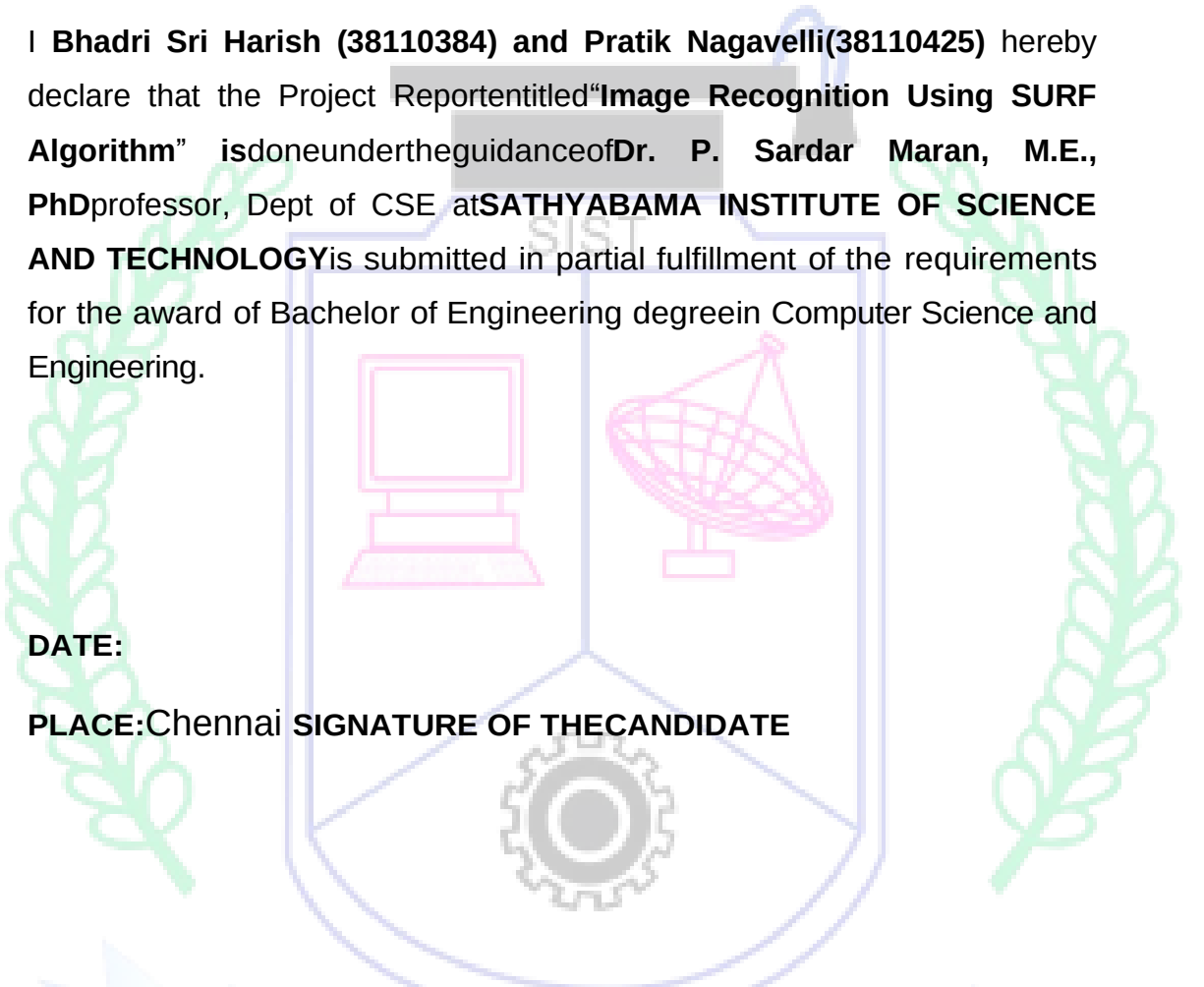
External Examiner

DECLARATION

I **Bhadri Sri Harish (38110384)** and **Pratik Nagavelli(38110425)** hereby declare that the Project Report titled **"Image Recognition Using SURF Algorithm"** is done under the guidance of **Dr. P. Sardar Maran, M.E., PhD** professor, Dept of CSE at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: Chennai SIGNATURE OF THE CANDIDATE



The logo of Sathyabama Institute of Science and Technology is a shield-shaped emblem. It features a computer monitor on the left and a satellite dish on the right, both in pink. At the bottom center is a grey gear. The shield is flanked by green laurel branches. Below the shield is a blue banner with the text 'JUSTICE PEACE REVOLUTION' in white. At the very bottom, the text 'SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY' is written in yellow, with 'DEEMED UNIVERSITY' in smaller yellow text below it.

JUSTICE PEACE REVOLUTION

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEEMED UNIVERSITY

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr.L. Lakshmanan M.E., Ph.D.** and **Dr.S. Vigneshwari M.E., Ph.D.** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P.Sardar Maran,M.E., Ph.D.**, for her valuable guidance,suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEEMED UNIVERSITY
JUSTICE PEACE REVOLUTION

ABSTRACT

In pc vision, Image process is one among the important areas of analysis. Any style of signal process that the input is a picture like images or frames of video. Specifically, feature choice during a very important role in image process. The results of the image process are a picture, or a collection of parameters or characteristics associated with the image. Image-processing techniques which will notice the image on any image or live video applying normal signal-processing techniques thereon. Image process typically refers to digital image process, however analog and optical image process also are attainable.

The acquisition of image is observed as imaging and most of the formulas focuses on the trailing algorithm to smoothen the video sequence. on the opposite hand, few strategies use the previous accessible data concerning image color, shape, texture then on. the aim of this paper is to research and review the previous approach towards image trailing and image detection exploitation video sequences through the SURF algorithm.



LIST OF ABBREVIATIONS

SR. NO	ABBREVIATION	EXPANSION
1	KNN	K Nearest Neighbor



LIST OF TABLES

SR. NO	NAME OF TABLE	PAGE NUMBER
1	ALGORITHM COMPARISION	31



LIST OF FIGURES

FIGURE. NO.	NAME OF FIGURE	PAGE NUMBER
3.1	Classification vs Regression	10
3.2	System Architecture	12
4.1	Random Forest	13
4.2	KNN	14
5.1	Density Diagram	24
5.2	Confusion Matrix for Random Forest	25
5.3	Confusion Matrix For KNN	26
5.4	Correlation Diagram	27
5.5	Correlation Matrix Plot Diagram	28
5.6	Accuracy Formula	29
5.7	Precision Formula	29
5.8	Recall Formula	30
5.9	F1-score Formula	30

5.10	Algorithm Comparison	31
5.11	Hist Diagram	33
5.12	State Prediction	34
5.13	Crop Prediction	35



TABLE OF CONTENTS

CHAPTER NO. TITLEPAGE NO.

ABSTRACT i

LIST OF ABBREVIATIONS ii

LIST OF TABLES iii

LIST OF FIGURES iv

1 INTRODUCTION1

1.1 OUTLINE

1.2 MODEL IDE

1.3 PROBLEM STATEMENT

1.4 PROPOSED SYSTEM4

1.5 OBJECTIVE

2 LITERATURE SURVEY

3 METHODOLOGY

3.1 INTRODUCTION TO MACHINE LEARNING

3.2 TRAINING THE DATA

3.2.1SUPERVISED LEARNING 9

3.2.2UNSUPERVISED LEARNING 9

3.3 METHODS IN SUPERVISED LEARNING 10

3.3.1CLASSIFICATION 10

3.3.2 REGRESSION 11

3.4SYSTEM ARCHITECTURE 11

4ALGORITHM

13

4.1	RANDOM FOREST	13
4.2	K NEAREST NEIGHBOUR	14
4.3	DATASETS	14
4.4	PACKAGES	15
4.4.1	DATA MANIPULATION PACKAGES	16
4.4.2	MODEL BUILDING PACKAGE	17
4.4.3	DATA VISUALIZATION PACKAGES	18
4.5	SYSTEM REQUIREMENTS	19
4.6	BACKGROUND STUDY	20
4.6.1	OVERVIEW	20
4.6.2	YIELD PREDICTION	20
5	RESULTS AND DISCUSSION	23
5.1	RESULTS	23
5.2	ANALYSIS	33
6	CONCLUSION AND FUTUREWORK	37
6.1	CONCLUSION	37
6.2	FUTURE SCOPE	37
	REFERENCES	38
	APPENDIX	41
A.	SAMPLE CODE	41
B.	SCREEN SHOTS	46
C.	PUBLICATION WITH PLAGIARISM REPORT	50

CHAPTER 1

INTRODUCTION

Image detection has been a well-studied subject for decades since it arises in many practical scenarios of modern marketing and advertising. Image detection aimed to enable computers to detect the image in an image without human intervention.

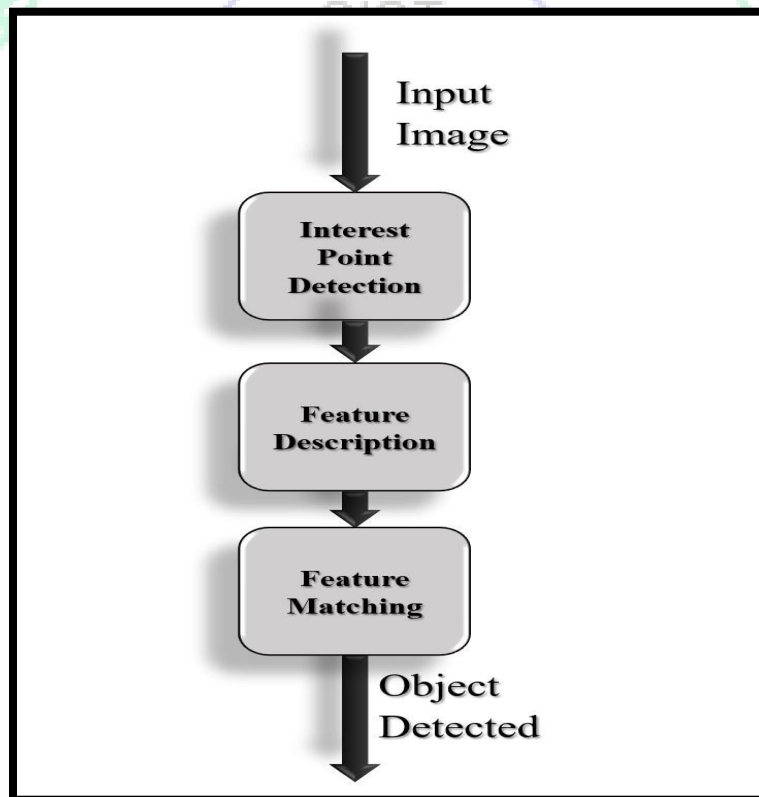


Fig 1.1

Images are key elements for companies and play an essential role in industry and commerce. Different images may have a similar layout but a slight difference in the spatial disposition of the graphics elements, the difference in orientation, size, and shape. The image of different firms exists in the database. The imaged exit in text form, graphic form or in both i.e., hybrid form. Therefore, it is necessary to extract the feature

image of the image as well as the test image so that it can identify the text portion and graphic portion of the image properly.

1.1 PRE-PROCESSING

It includes the binarization of images. Binarization is the process of converting an image from color to black-and-white (called “binary image”). The binary image contains only two-pixel values 0 and 1. This process reduces the number of dimensions we must work with.

1.2 FEATURE EXTRACTION

This process extracts the key-points of all the images. And extract the key-points of the input image with the same technique used to extract the key points from the image. Where the key point is a circular image region with an orientation. It is delineated by a geometrical frame of 4 parameters: the key purpose center coordinates x and y , its scale (the radius of the region), and its orientation.

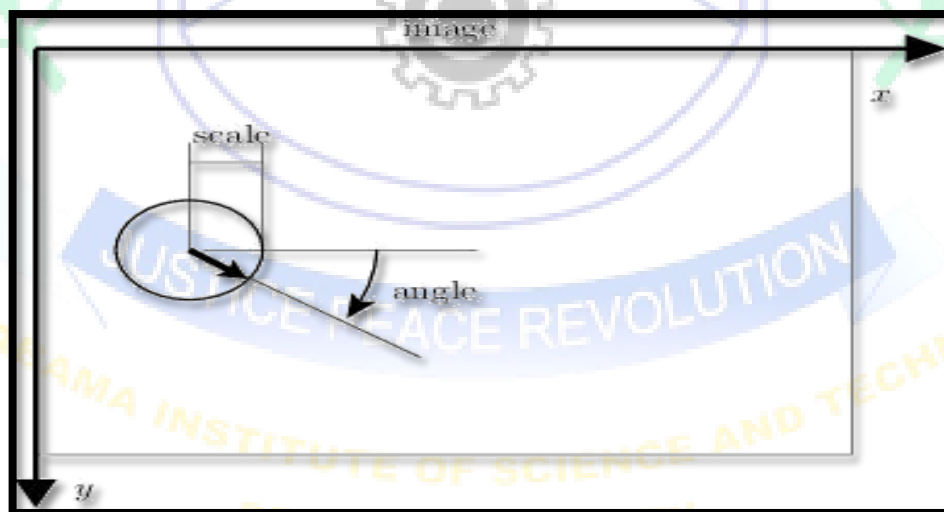
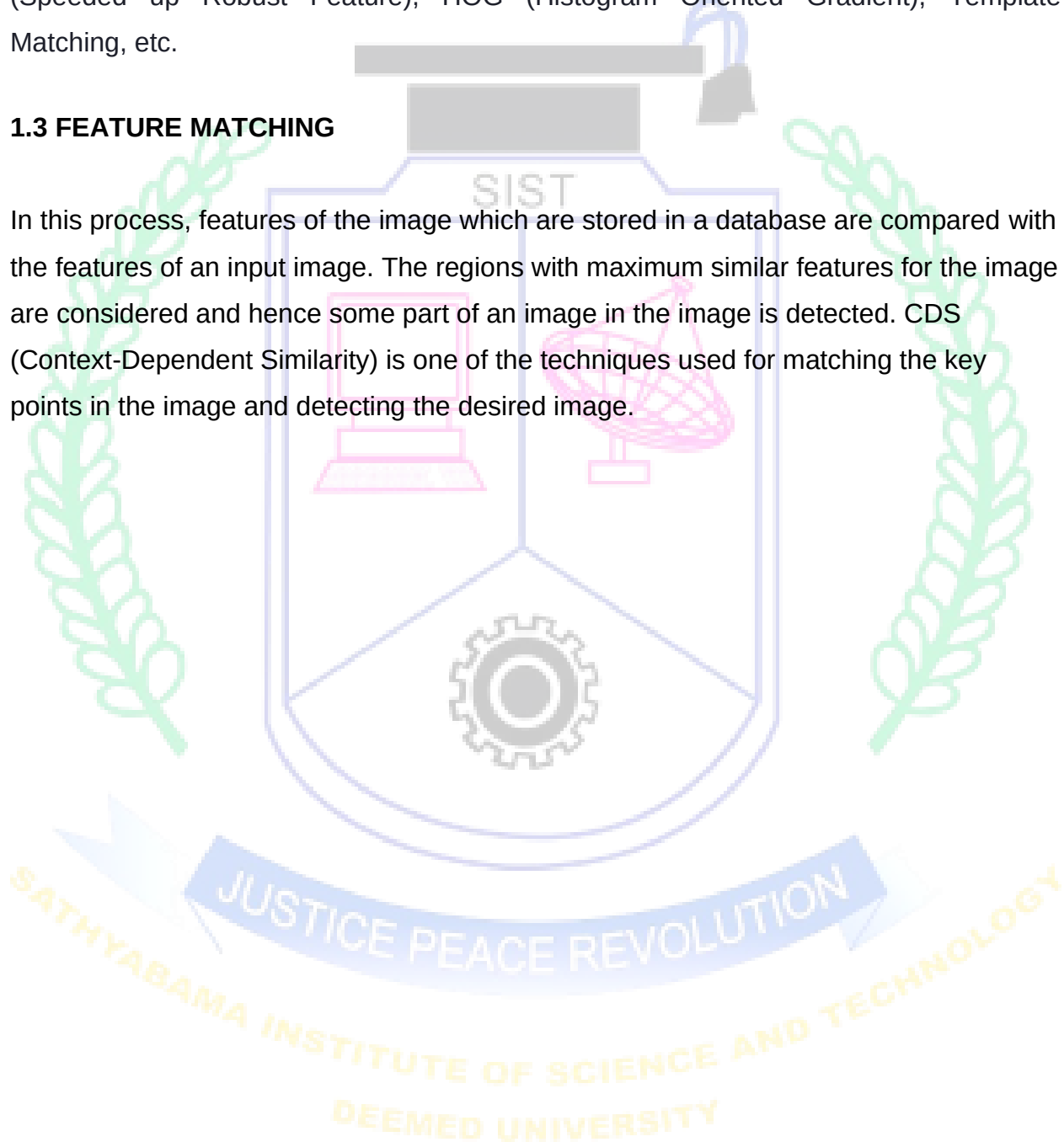


Fig 1.2 Key point in an image

There are many ways to extract features with reference to the goals. Some are based on the edges or the corners, other on the curvature of the shape. The different types of techniques for feature extraction are SIFT (Scale Invariant Feature Transform), SURF (Speeded up Robust Feature), HOG (Histogram Oriented Gradient), Template Matching, etc.

1.3 FEATURE MATCHING

In this process, features of the image which are stored in a database are compared with the features of an input image. The regions with maximum similar features for the image are considered and hence some part of an image in the image is detected. CDS (Context-Dependent Similarity) is one of the techniques used for matching the key points in the image and detecting the desired image.



CHAPTER 2

LITERATURE SURVEY

In the literature survey, we have studied different feature extraction techniques. Images of different formats and styles exist in the database. To recognize the different types of an image of different images features can be used for extracting the feature and testing images. Using some analysis, there are main three techniques using this testing of images are done, which are Speeded up robust feature (SURF), Scale-invariant feature transform (SIFT), and Histogram oriented Gradient (HOG). SURF, SIFT, HOG is used for extracting the shape-based feature vectors of training and testing images. Finally, to check the relationship between the sample images of testing sub-images and training sub-images of the classes the Manhattan Distance classifier is used.

2.1 SCALE INVARIANT FEATURE TRANSFORM (SIFT)

SIFT, Associate in Nursing word form of Scale-invariant feature rework, is one in all the foremost far-famed algorithms to sight and describe the native options in a picture, fictional by David Lowe in 1999. The SIFT is for extracting good invariant options from pictures which will be accustomed perform reliable matching between the various views of a scene or object. This feature descriptor maintains sensible fame for its properties as a result of it's invariants to scaling, translation, rotation, transformation and partly to illumination changes. The goal of those techniques is to convert a picture during a set of vectors, and these vectors have the characteristic to explain enough the image. As David G. Lowe represented SIFT consists of 4 major stages:

- (1) Scale-space peak selection
- (2) Key point localization
- (3) Orientation assignment
- (4) Key point descriptor

First of all, scale-space has to be generated. To generate scale-spaces apply Gaussian over sample image with varying σ value. Then down sample it and find Laplacian of Gaussian. To efficiently locate the key point location in scale space difference of Gaussian.

$D(x,y,s)$ (DOG), is calculated from variations of two close scale separated by a multiplicative factor. A point is decided as a local minimum or maximum concerning neighbourhood points. The orientation and gradient magnitude are computed for each image. An orientation histogram is developed from the gradient orientation of the individual points inside a region around the key point. Then a distinctive descriptor is obtained for each key point. The image gradient and magnitudes and orientations, dependent on the major orientation of the key point are sampled within a 16×16 region. These samples are then assembled into an orientation histogram compiling the contents into 4×4 sub-region. Each histogram has 8 bins covering 360 degrees of orientation. Therefore, having a $4 \times 4 \times 8 = 128$ feature vector is produced for each Key point.

2.2 Histogram Oriented Gradient (Hog)

Histogram of Oriented Gradients (HOG) is motivated on Scale-Invariant Feature Transform (SIFT) descriptors proposed by David G. Lowe. To produce HOG, the cell histograms of each pixel inside the cell cast a weighted vote. In this operation, the histogram channels are calculated across rectangular cells by the calculation of the unsigned gradient. The cells are overlying on half of the area, means that each cell produces more than once to the final feature vector. To measure differences in contrast and illumination, the gradient strengths were regionally normalized, i.e. normalized over each cell. The 9 histograms with 9 bins are then combined to make a 1×81 dimensional feature vector.

2.3 Speeded up Robust Feature (Surf)

SURF, Speeded-Up robust Features, is another image detector and descriptor inspired by SIFT, was first presented by Herbert Bay in the ECCV 2006 conference in Graz, Austria. SURF approximates or even better earlier proposed schemes for robustness, repeatability, and distinctiveness, yet it can be computed and analyzed much faster.

SURF algorithm is used to extract the interest points from the reference image and test image instead of the SIFT algorithm. SURF is faster than SIFT. The main difference of SURF algorithm is it mainly implemented to the DOH (Determinant of Hessian), the extraction of the feature points in the image are achieved through simplification and approximation, compared to the SIFT algorithm, it reduces the complexity of feature point extraction and it also has the better real-time performance.

The major steps of SURF are:

- Fast-Hessian Detector
- Constructing the Scale-Space
- Accurate Interest Point Localization
- Interest Point Descriptor
- Orientation assignment
- Descriptor Components

Much of the performance increase in SURF is due to Integral Image.

The whole image $I(x, y)$ of an image $I(x, y)$ describes the total of all pixels in $I(x, y)$ of a rectangular area formed by $(0, 0)$ and (x, y) .

- Now compute the Hessian matrix, as a function of both scale and space.
- Then measure Hessian determinant applying the approximated Gaussians.
- The function of rotation-invariant interest points and localizing the size within the image will be classified into three steps:

- First, determine the edge value for the responses such all values below the predetermined threshold are removed.
- Then, find a collection of candidate points. to attain this each pixel within the scale-space is checked with its 26 neighbour's, combined the 8 points within the native scale and also the 9 in each of the scales above and below.
- Once interest points are localized each in area and scale.

2.4 EXPERIMENTAL RESULTS

Based on experimental results, it's complete that the accuracy of the SURF formula is beyond SIFT and HOG.

Process	Percentage Accuracy
SIFT	56.66%
HOG	76.66%
SURF	81.66%

Table 2.1 Percentage accuracy based on Feature Matching

The SIFT formula extracts a bigger range of options, however the matter is that the speed for all the process steps. Through it's found that the SIFT has detected a bigger range of options compared to SURF however it's suffered from speed.

The SURF is quick and has sensible performance because the same as SIFT

Algorithm	Detected Feature Points		Matching Features Points	Feature Matching Time
	Image 1	Image 2		
SIFT	892	934	41	1.534s
SURF	281	245	28	0.546s

Table 2.2 Comparison Between SIFT & SURF



CHAPTER 3

METHODOLOGY

3.1 OBJECTIVE

The work to extract the digital information began as soon as the idea of digitizing content that was present in physical mediums such as books, vinyl records gained foothold. The contributing ideas that would lay the foundations came from a variety of fields such as artificial intelligence, computational vision to psychology.

However, was field of computer vision which provided some of the first algorithms for searching features in video, audio, and images. With the growth of the internet Web engines caught on, and started to provide image searches. Efforts were also made for integration of such systems directly into commercial database systems.

It was realized by scientists during the course of developing media information systems that there was a widening semantic gap between the low level features that were used in computations by scientists and high level features that general users used their daily language when searching for images of interest. One of the earliest image-based search engine to address the semantic gap was the Imagescape and it could provide searches on over 10 million images on topics like sky, trees, river etc.

This paper covers only images though feature techniques applied on image can be applied on videos and vice versa. Even though a video is inherently composed of images, an image collection presents unique challenge because unlike a video where subject matter is unlikely to change suddenly from frame to frame and the object is a bit easier to find due to dominant presence in consecutive frames of the video, images in a collection may differ widely in the scope of matter they cover.

Early attempts were mainly focused towards human face recognition but steadily the search diversified towards detecting objects in general. The difficulty of detecting objects led to research and advancements in proposal of novel and better similarity features like color features based on NF, RGB and m color space. Other techniques have included texture matching on the basis of histogram and use of computational geometry to match shapes.

Learning algorithms became an important part of image retrieval systems as it was realized that pattern recognition between underlying relationships of features would yield better results than was possible by simple matching of features. They allow the computer system to build a semantic understanding of the image collection and reduce the effects of noise introduced due to real world clutter in the image contents as well as ordering of images in vast collections.

The earliest learning systems were based on neural networks, also used are components-based searches, statistics-based methods have also shown great promise. Some image retrieval systems also integrate human feedback in training the systems so there is a more human centered relevance to search results.

The search for better features continues to capture more relevant information as well as in terms of computation scalability in regards to application use in a server application or running on the latest Smart phones.

Through learning from historical relationships and trends in the data, these analytical models enable researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights."

3.2 SYSTEM ARCHITECTURE

The figure shows flow diagram for Image detection. It consists of steps: pre-processing feature extraction, and finally image detection.

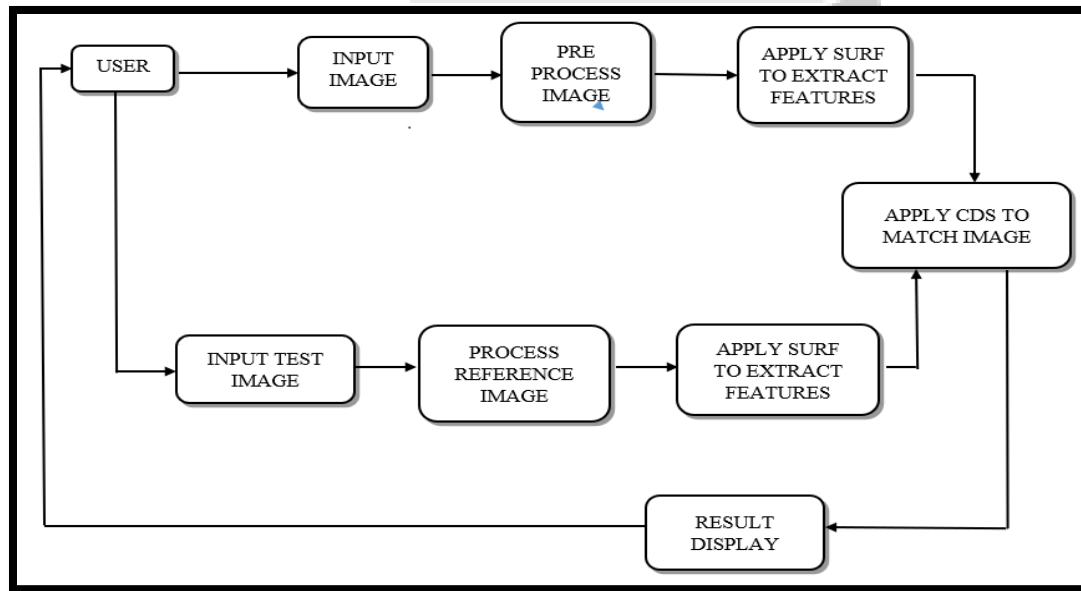


Fig 3.1 Flow diagram for Image detection

The user inputs the look at image and reference image that's needed to be born-again into a binary image. once binarization, the present pictures are the photographs to that SURF is applied for extracting the options. Here, we have a tendency to a victimization the SURF formula as a result of from a literature survey it's found that the SURF algorithm is best than different techniques.

After extracting the options CDS (Context-Dependent Similarity) is applied for matching the key points of the photographs. during this approach, the take a look at image are going to be detected in a picture.

CHAPTER 4

ALGORITHMS

4.1CDS IMAGE DETECTION ALGORITHM

Context-Dependent Similarity

Let $TX = \{x_1, \dots, x_n\}$, $TY = \{y_1, \dots, y_m\}$ be respectively the list of interest points taken from a reference image and a test image (the value of n, m may vary with TX, TY).

Input: {reference image: I_x Test image: I_y , CDS parameter: $\epsilon, \alpha, \tau, \beta, M_a, M_r$ }

Processes: Extract SURF from I_x, I_y and let $T_x = \{x_1, x_n\}$,

$T_y = \{y_1, y_n\}$ be respectively the list of interest points taken from both images;

Step 1:

For $j=1$ to n

Find context matching for x_j where it is a key point of the reference image.

End for

For $j=1$ to n

Find context matching for y_j where it is a key point of the test image.

End For

Step 2:

Set $p=1$ to $\max=30$

Step 3:

For $j=1$ to n

For $i=1$ to m Compute CDS matrix

Increment t i.e., does $p++$;

End for

End for

Repeat step 3 until $p > \max$ or convergence.

Step 4:

For $j=1$ to n do

For $i=1$ to m do

Compute $Ky_i|x_j$

Match between x_j and x_i is declared only if

$Ky_i|x_j \geq \sum_{ms=i} Ky_s|x_j$

Step 5:

If number of matches in $T_y > \tau|T_x|$

Then image matched i.e. detected

otherwise

Image not detected.



Fig 4.1 Flowchart of SURF algorithm

Output: A Boolean value determining whether the source image in I_x is identified in I_y .

In our project, the SURF algorithm will be used for extracting features and CDS for matching features as these techniques are efficient comparatively.

Sample output:

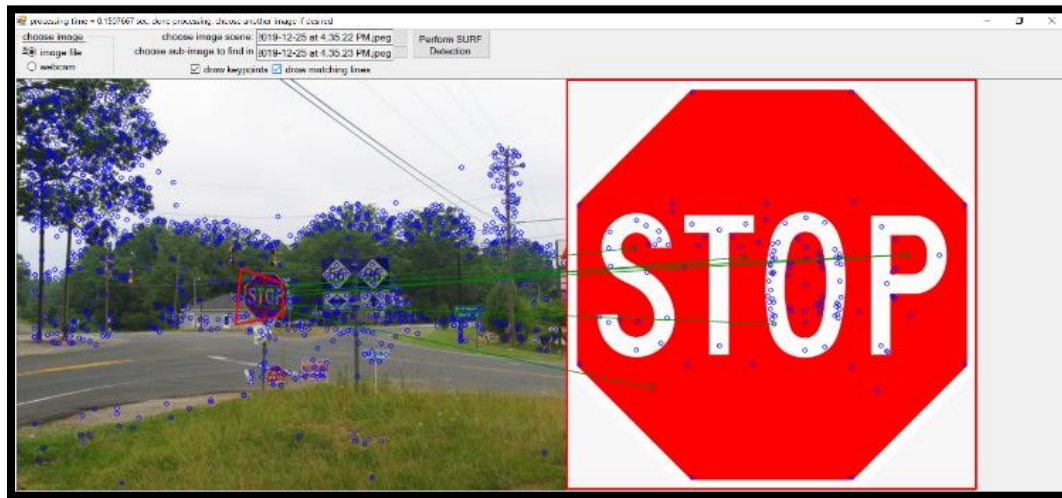


Fig 4.2 Sample Output

4.2 SOFTWARE REQUIREMENT

The software requirements in our project are Open CV and Visual Studio 2013.

4.2.1 INTRODUCTION TO OPEN CV

An Open CV (Open-Source Computer Vision) is an Image Processing library designed by Intel and controlled by Willow Garage. The Open CV is a library of machine learning software as well as an open-source computer vision. It is implemented to stimulate the use of machine knowledge in commercial products and to provide standard infrastructure for computer vision applications.

The library of an Open CV has more than 2500 optimized algorithms, which consist of a broad set of both standard and state-of-the-art computer vision and machine learning algorithms.

These all algorithms are used to recognize and identify faces, analyze human actions in videos, classify objects, track moving objects, track camera movements, produce 3D point clouds from stereo cameras, extract 3D models of objects, join images collectively to produce a high-resolution image of an entire picture, find related images from an image database, remove red eyes from images obtained using flash, follow eye movements, recognize spectacle and establish tags to overlay it with augmented fact, etc. The library is used broadly in companies, governmental bodies, and research groups.

Open CV is written purely in C++. It produces C++, C, Java, Python and MATLAB interfaces and supports Windows, Android, Linux, and Mac OS

4.2.2 INTRODUCTION TO MICROSOFT VISUAL STUDIO

Create the characteristic with the most helpful in class Integrated Development Environment (IDE) from Microsoft. It is used to produce computer programs for web sites, as well as Microsoft Windows, web services, and web applications. Windows Store, Microsoft Silverlight, Windows API, and Windows Forms all are managed Microsoft software development platforms. It can produce both primary codes and obtained code.

Visual Studio carries various programming languages and provides the code debugger and editor to verify nearly any. C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# these all are Built in languages.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 RESULTS

From this data, we can see that model 1 resulted in the lowest value of accurate classifications. Using SURF and colour histogram, there is a clear trend of increasing in classification accuracy when model 2 and model 3 were used, which both behave quite similarly. On average, model 2 and model 3 were shown to have the optimal performance for SURF and colour histograms techniques. Hence, it could conceivably be hypothesised that for these techniques, matching query images with either image clusters or composite models increases the accuracy of the classification by about 21.5%. Regarding spatial colour histograms, model 2 has shown optimal performance compared to model 1 and model 3. In our project, we proposed a new classification approach that combines colour features along with geometry features using SURF algorithm, colour histograms and spatial colour histograms. Since model 2 and model 3 perform quite similarly using SURF and colour histograms, model 3 has been selected as it makes classification faster (comparing to a single image is faster than comparing to 4 images)

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

We have a new Image and localization approach referred to as reference likeness reliant of a new classroom was founded on the strength of the suggested process in many aspects. First equality in addition to information about the spatial configuration that design as well as Visual characteristics, the second is command of our energy context and leverage answer by regularization ability, encompassing various aspects of the third partial occlusion tolerance, both with some flexibility in his eye to detect duplicate Image as well as the Image suitable for children, and the fourth equal theoretical equality, such as composition a roundedness check likelihood of the idea of the existence of a quotation in the Image displays the success under equivalent and explore high. In addition, we achieved scalability and other rigid and nonrigid Image changes.

6.2 FUTURE SCOPE

There are currently no regulations in the United States expressly covering the biometric data of a person. Facial recognition devices are already being tested or implemented for airport protection, and it is reported that their faceprint has now been produced by more than half the United States populace. Information may be collected and processed by a facial recognition program, and a person does not even recognize it. Then, a hacker might reach the details, and the knowledge of a person would propagate without even realizing it. Government entities or marketers may use this data to monitor individuals too. Worse still, a false positive may include a person for a crime they are not.

Hundreds of companies have embraced face recognition. Integrating and installing is reasonably straightforward, but it has also provided users a feeling of utilizing a system that is more sophisticated and safer than passwords or PINs, thereby increasing user experience. Nonetheless, plenty is often unclear on the road to implementing what

many deem the ideal biometric approach, causing several relatively severe blunders along the way

REFERENCES

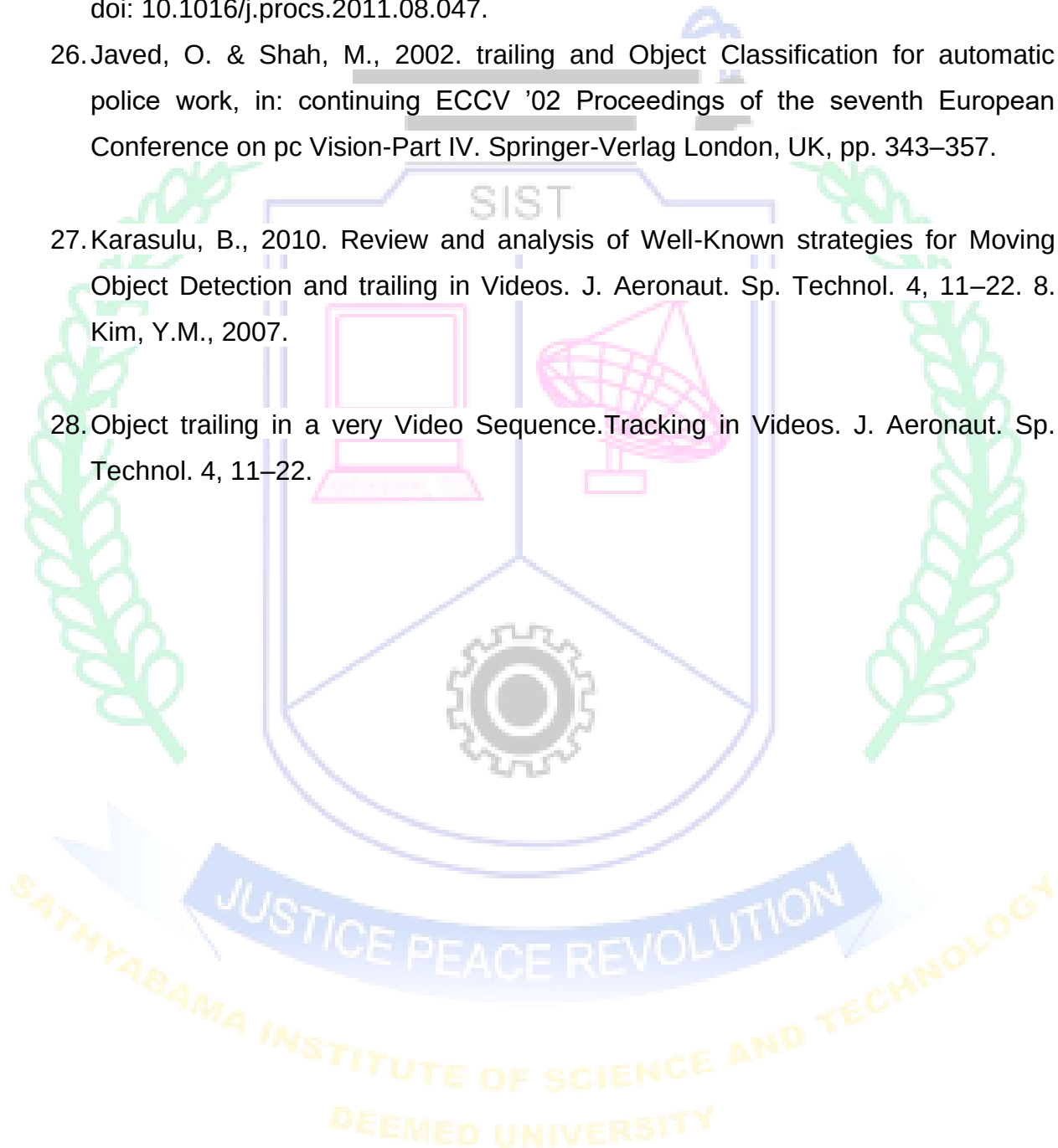
1. "Image Recognition Technique Victimisation Sift Descriptor, Surf Descriptor and Hog Descriptor", Chinmoy Biswas, Jaydeep Mukherjee, International Journal of laptop Applications (0975 – 8887) Volume 117 – No. 22, May 2015.
2. "Image Detection and Recognition from the pictures yet as Videos", Meera SambhajiSawalkar, MrunalineePatole, International Journal of Science and analysis (IJSR) ISSN (Online): 2319-7064 Impact issue (2012): three.358.
3. "Techniques for seeing in pictures and Multi-Object Detection", Khushboo Khurana, Reetu Awasthi, International Journal of Advanced analysis in laptop Engineering & Technology (IJARCET) Volume a pair of, Issue 4, April 2013.
4. "Image Matching And Recognition System victimisation Surf", Remya Ramachandran, Andrews Jose, International Journal of analysis in laptop and Communication Technology, Vol 3, Issue 9, Gregorian calendar month – 2014.
5. "A Comparison of SIFT and SURF", P M Panchal, S R Panchal, S K Shah, International Journal of Innovative analysis in laptop and Communication Engineering Vol. 1, Issue 2, April 2013.
6. "Object recognition from native scale-invariant features", D. G. Lowe, in Proc. of the International Conference on laptop Vision, Corfu, 1999.
7. "Surf: accelerated sturdy features", H. Bay, T. Tuytelaars, and L. J. V. Gool, in ECCV (1) (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of Lecture Notes in computing, pp. 404_417, Springer, 2006.

8. "Digital Image Processing" book, By Rafael C Gonzalez, page no. 29.
9. accolade Firschein and Thomas M Strat, RADIUS: Image Understanding for mental imagery Intelligence. Morgan Kaufmann, 1997.
10. Andres Huertas and Ramakant Nevatia, "Detecting changes in aerial views of synthetic structures," in Proc. of International Conference on laptop Vision, 1998.
11. Aaron J Heller, Yvan G Leclerc, and Quang-Tuan Luong, "A framework for sturdy three-D amendment detection," in Proceedings for International conference on Remote Sensing, SPIE., 2001.
12. Chris Stauffer and W E Grimson, "Adaptive background mixture models for real time pursuit," in Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)., vol. 2, 1999, pp. 246-252.
13. K. Sumithra, S. Buvana, R. Somasundaram, "A Survey on varied sorts of Image process Technique", International Journal of Engineering analysis & Technology (IJERT) Vol. 4 Issue 03, March-2015.
14. Richard O Duda, Peter E Hart, and David G wading bird, Pattern Classification.: Wiley-Interscience, 2001.
15. MeghaSoni, Anand Khare, Asst. Prof. Saurabh Jain, "a survey of digital image process and its problem", International Journal of Scientific and analysis Publications, Volume 4, Issue 2, Feb 2014 one ISSN 2250-3153.
16. Aggarwal, A., Biswas, S., Singh, S., Sural, S. & Majumdar, A.K., 2006. Object trailing victimisation Background Subtraction and Motion Estimation in MPEG Videos, in seventh Asian Conference on pc Vision. Springer-Verlag Berlin

Heidelberg, pp. 121-130. Doi:10.1007/11612704_13.

17. Aldhaheeri, A.R. & Edirisinghe, E.A., 2014. Detection and Classification of a Moving Object in an exceedingly Video Stream, in: Proc. of the Intl. Conf. on Advances in Computing and data Technology. Institute of analysis Engineers and Doctors, Kingdom of Saudi Arabia, pp. 105–111. doi:10.3850/ 978-981-07-8859-9_23.
18. Ali, S.S. & Zafar, M.F., 2009. a strong adaptational methodology for detection and trailing of moving objects, in: 2009 International Conference on rising Technologies. IEEE, pp. 262–266. doi:10.1109/ICET.2009.5353164
19. Amandeep, Goyal, M., 2015. Review: Moving Object Detection Techniques. Int. J. Comput. Sci. Mob. Comput. 4, 345 – 349.
20. Athanesious, J. & Suresh, P., 2012. Systematic Survey on Object trailing strategies in Video. Int. J. Adv. Res. Comput. Eng. Technol. 1, 242–247.
21. Balasubramanian, A., Kamate, S., & Yilmazer, N., 2014. Utilization of strong video process techniques to assist economical object detection and trailing. Procedia Comput. Sci. 36, 579–586. doi: 10.1016/j.procs.2014.09.057.
22. Chakravarthy, S., Aved, A., Shirvani, S., Annappa, M., & Blasch, E., 2015. Adapting Stream process Framework for Video Analysis. Procedia Comput. Sci. 51, 2648–2657. doi: 10.1016/j.procs.2015.05.372.
23. Chate, M., Amudha, S., & Gohokar, V., 2012. Object Detection and trailing in Video Sequences. ACEEE Int. J. Signal Image method. 3.
24. Feris, R.S., 2000. Detection and trailing of countenance in Video Sequences, in: Lecture Notes in computing. Springer-Verlag Berlin Heidelberg, pp. 197–206.

25. Foytik, J., Sankaran, P. & Asari, V., 2011. trailing and recognizing multiple faces victimization Kalman filter and ModularPCA. Procedia Comput. Sci. 256–261. doi: 10.1016/j.procs.2011.08.047.
26. Javed, O. & Shah, M., 2002. trailing and Object Classification for automatic police work, in: continuing ECCV '02 Proceedings of the seventh European Conference on pc Vision-Part IV. Springer-Verlag London, UK, pp. 343–357.
27. Karasulu, B., 2010. Review and analysis of Well-Known strategies for Moving Object Detection and trailing in Videos. J. Aeronaut. Sp. Technol. 4, 11–22. 8. Kim, Y.M., 2007.
28. Object trailing in a very Video Sequence. Tracking in Videos. J. Aeronaut. Sp. Technol. 4, 11–22.



APPENDIX

A. CODE

Option Strict

Imports System.Drawing

Imports System.Diagnostics

Imports Emgu.CV

Imports Emgu.CV.CvEnum

Imports Emgu.CV.Features2D

Imports Emgu.CV.Structure

Imports Emgu.CV.UI

Imports Emgu.CV.Util

Public Class frmSURF

' member variables

Dim blnFirstTimeInResizeEvent As Boolean = True

'used to throw out first time in form resize event, see resize event comments for details

Dim intOrigFormWidth As Integer

'vars used to save original

Dim intOrigFormHeight As Integer

'form and image box sizes,

Dim intOrigImageBoxWidth As Integer

'used to resize image box

Dim intOrigImageBoxHeight As Integer

'when form is resized

Dim capWebcam As Capture

'Capture object for webcam

Dim blnWebcamCapturingInProgress As Boolean = False

'variable to keep track of if SURF function has been added to application's list of tasks

Dim imgSceneColor As Image(Of Bgr, Byte) = Nothing

'original image scene, in color

Dim imgToFindColor As Image(Of Bgr, Byte) = Nothing

'original image to find, in color

Dim imgCopyOfImageToFindWithBorder As Image(Of Bgr, Byte) = Nothing

'use as a copy of image to

find, so we can draw a border on this image without altering original image to find

Dim blnImageSceneLoaded As Boolean = False

'flag to track if a scene image has been loaded successfully

Dim blnImageToFindLoaded As Boolean = False

'flag to track if an image to find has been loaded successfully

'resulting image of image scene and image to find concatenated together,

Dim imgResult As Image(Of Bgr, Byte) = Nothing

'with

border drawn around found image, and key points and matching lines also drawn if chosen

Dim bgrKeyPointsColor As Bgr = New Bgr(Color.Blue)

'color to draw key

points on result image

```
Dim bgrMatchingLinesColor As Bgr = New Bgr(Color.Green)  
lines on result image
```

'color to draw matching

```
Dim bgrFoundImageColor As Bgr = New Bgr(Color.Red)  
draw around found image in scene portion of result image
```

'color of box to

```
Dim stopwatch As Stopwatch = New Stopwatch()  
'stopwatch to track processing time
```

```
' constructor  
Sub New()  
    InitializeComponent()  
EndSub
```

'this call is required by the

```
intOrigFormWidth = Me.Width  
intOrigFormHeight = Me.Height  
intOrigImageBoxWidth = ibResult.Width  
intOrigImageBoxHeight = ibResult.Height
```

EndSub

```
Private Sub frmSURF_Resize( sender As System.Object, e As System.EventArgs) Handles MyBase.Resize
```

'This If Else statement is necessary to throw out the first time the frmSURF_Resize event is called.
'For some reason, in VB.NET the Resize event is called once before the constructor, then the constructor is called,

'then the Resize event is called each time the form is resized. The first time the Resize event is called
'(i.e. before the constructor is called) the coordinates of the components on the form all read zero,
'therefore we have to throw out this first call, then the constructor will run and get the correct initial
'component location data, then every time after that we can let the Resize event run as expected
If (blnFirstTimeInResizeEvent = True) Then

'if first time in resize event

blnFirstTimeInResizeEvent = False

'update flag variable

Else

'after first time in resize event

ibResult.Width = Me.Width - (intOrigFormWidth - intOrigImageBoxWidth)

'resize image box to form

ibResult.Height = Me.Height - (intOrigImageBoxHeight - intOrigImageBoxHeight)

EndIf

EndSub

```
Private Sub rdoImageFile_CheckedChanged( sender As System.Object, e As System.EventArgs)
```

```
Handles rdoImageFile.CheckedChanged
```

```
If (rdoImageFile.Checked = True) Then
```

'if image file was just chosen

```
If (blnWebcamCapturingInProgress = True) Then
```

'if webcam capturing is in

process (i.e. if webcam radio button was previously chosen)

```
RemoveHandler Application.Idle,
```

```
New EventHandler(AddressOf Me.PerformSURFDetectionAndUpdateGUI)  
application idle
```

'remove event from

blnWebcamCapturingInProcess = False

'update flag variable

EndIf

imgSceneColor = Nothing

'reset class level variables

imgToFindColor = Nothing

imgCopyOfImageToFindWithBorder = Nothing

imgResult = Nothing

blnImageSceneLoaded = False

blnImageToFindLoaded = False

txtImageScene.Text = ""

'reset form

txtImageToFind.Text = ""

ibResult.Image = Nothing

button"

Me.Text = "Instructions: use ""..."" buttons to choose both image files, then press Perform SURF

'update title bar text

btnPerformSURFOrGetImageToTrack.Text = "Perform SURF Detection"

'update button text

ibResult.Image = Nothing

'make image box blank

lblImageScene.Visible = True

'show controls that are used for still images

lblImageToFind.Visible = True

txtImageScene.Visible = True

txtImageToFind.Visible = True

btnImageScene.Visible = True

btnImageToFind.Visible = True

EndIf

EndSub

PrivateSubrdoWebcam_CheckedChanged(sender As System.Object, e As System.EventArgs)

HandlesrdoWebcam.CheckedChanged

If(rdoWebcam.Checked = True) Then

'if webcam was just chosen

imgSceneColor = Nothing

'reset class level variables

imgToFindColor = Nothing

imgCopyOfImageToFindWithBorder = Nothing

```

imgResult = Nothing

blnImageSceneLoaded = False

blnImageToFindLoaded = False

```

```

txtImageScene.Text = ""
'reset form

txtImageToFind.Text = ""

ibResult.Image = Nothing

Try
    capWebcam = New Capture()
    'to instantiate Capture object for webcam
Catch ex As Exception
    'if not successful
    Me.Text = ex.Message
    'show error message on title bar
    Return
EndTry
'and bail

Me.Text = "Instructions: hold image to track up to camera, then press ""get image to track"" "
'update title bar text
btnPerformSURFOrGetImageToTrack.Text = "get image to track"
'update button text

imgToFindColor = Nothing

AddHandler Application.Idle,
New EventHandler(AddressOf Me.PerformSURFDetectionAndUpdateGUI)
'add SURF function to
application's list of tasks

blnWebcamCapturingInProgress = True

flag variable

lblImageScene.Visible = False
'hide controls that are not used with webcam
lblImageToFind.Visible = False

txtImageScene.Visible = False

txtImageToFind.Visible = False

btnImageScene.Visible = False

btnImageToFind.Visible = False

EndIf

EndSub

```

```

.....

```

```

Private Sub btnImageScene_Click( sender As System.Object, e As System.EventArgs) Handles btnImageScene.Click
    Dim dialogResult As DialogResult = ofdImageScene.ShowDialog()
    'bring up image scene file dialog box

    If (dialogResult = Windows.Forms.DialogResult.OK Or dialogResult = Windows.Forms.DialogResult.Yes) Then
        'if OK or Yes was chosen
        txtImageScene.Text = ofdImageScene.FileName
        'write

        file name to text box
    Else
        Return
    End If

    Try
        imgSceneColor = New Image(Of Bgr, Byte)(txtImageScene.Text)
        'try to load scene image
    Catch ex As Exception
        'if not successful
        Me.Text = ex.Message
        'show error message on title bar
        Return
    End Try
    'and bail

    blnImageSceneLoaded = True
    'if we get here, scene image was loaded successfully, update member variable

    If (blnImageToFindLoaded = False) Then
        'if to find image has not
        'been loaded yet
        ibResult.Image = imgSceneColor
        'show
        image scene we just loaded on image box
    Else
        'if to find image has already been
        loaded
        ibResult.Image = imgSceneColor.ConcatHorizontal(imgCopyOfImageToFindWithBorder)
        'concatenate image to find with border and scene image, show result on image box
    End If
End Sub

Private Sub btnImageToFind_Click( sender As System.Object, e As System.EventArgs) Handles btnImageToFind.Click
    Dim dialogResult As DialogResult = ofdImageToFind.ShowDialog()
    'bring up image to find dialog box

```

```

If(dialogResult = Windows.Forms.DialogResult.OK Or dialogResult = Windows.Forms.DialogResult.Yes) Then
    'if OK or Yes was chosen
    txtImageToFind.Text = ofdImageToFind.FileName

    'write
    file name to text box
    Else
        'if cancel was chosen
        Return

    'bail
    EndIf

Try
    imgToFindColor = New Image(Of Bgr, Byte)(txtImageToFind.Text)
    'try to load to find image
    Catch ex As Exception
        'if not successful
        Me.Text = ex.Message
        'show error message on title bar
        Return
    EndTry

    'and bail

    blnImageToFindLoaded = True
    'if we get
    here, to find image loaded successfully, update member variable

    imgCopyOfImageToFindWithBorder = imgToFindColor.Copy()
    'make a copy of
    the image to find, so we can draw on the copy, therefore not changing the original image to find

    'draw 2 pixel wide border around the copy of the image to find,
    use same color as box for found image
    imgCopyOfImageToFindWithBorder.Draw(New Rectangle(1, 1, imgCopyOfImageToFindWithBorder.Width - 3,
    imgCopyOfImageToFindWithBorder.Height - 3), bgrFoundImageColor, 2)

    If(blnImageSceneLoaded = True) Then
        'if scene image is already loaded
        ibResult.Image = imgSceneColor.ConcatHorizontal(imgCopyOfImageToFindWithBorder)
        'concatenate to find image with border to scene image, show result on image box
    Else
        'if scene image has not already been loaded
        ibResult.Image = imgCopyOfImageToFindWithBorder
    EndIf

```

```

        'show to find image border we just loaded on image box
    EndIf
EndSub

.....

PrivateSubtxtImageScene_TextChanged( senderAsSystem.Object, e AsSystem.EventArgs)
HandlestxtImageScene.TextChanged
    txtImageScene.SelectionStart = txtImageScene.Text.Length
    'move
    caret to end of text box so file name is visible rather than file extension
EndSub

.....

PrivateSubtxtImageToFind_TextChanged( senderAsSystem.Object, e AsSystem.EventArgs)
HandlestxtImageToFind.TextChanged
    txtImageToFind.SelectionStart = txtImageToFind.Text.Length
    'move caret to end
    of text box so file name is visible rather than file extension
EndSub

.....

PrivateSubbtnPerformSURForGetImageToTrack_Click( senderAsSystem.Object, e AsSystem.EventArgs)
HandlesbtnPerformSURForGetImageToTrack.Click

    If(rdoImageFile.Checked = True) Then

        'if using still images
        If(txtImageToFind.Text<>String.EmptyAndtxtImageScene.Text<>String.Empty) Then
            'check that something is entered in each image file text box
            PerformSURFDetectionAndUpdateGUI(NewObject(), NewEventArgs())
            'call Perform SURF
        Else
            'else
            Me.Text = "choose image files first, then choose Perform SURF Detection button"
            'remind user to choose both image files before choosing Perform SURF
        EndIf
    ElseIf(rdoWebcam.Checked = True) Then

        'if using the webcam
        imgToFindColor = imgSceneColor.Resize(320, 240, INTER.CV_INTER_CUBIC, True)
        'use most recent image from webcam, which will be in imgSceneColor, then shrink and save as new image to
        track
        Me.Text = "Instructions: to update image to track, hold image up to camera, then press ""update image
        to track""""
        'update title bar text
        btnPerformSURForGetImageToTrack.Text = "update image to track"
        'update button text

    Else
        'should never get here
    EndIf
EndSub

.....

PrivateSubckDrawKeyPoints_CheckedChanged( senderAsSystem.Object, e AsSystem.EventArgs)
HandlesckDrawKeyPoints.CheckedChanged
    If(ckDrawKeyPoints.Checked = False) Then

```

```

'if draw key points was just unchecked
    ckDrawMatchingLines.Checked = False
    'unchecked draw matching lines check box
    ckDrawMatchingLines.Enabled = False
    'and disable draw matching lines check box
Elseif(ckDrawKeyPoints.Checked = True) then
'else if draw key points was just checked
    ckDrawMatchingLines.Enabled = True
    're-enable draw matching lines check box
EndIf

If(rdoImageFile.Checked = True) Then
    'if using
    image from file
    btnPerformSURFOrGetImageToTrack_Click(NewObject(), NewEventArgs())
    'call SURF button click event to update image for draw key points check box change
EndIf
EndSub

.....

PrivateSubckDrawMatchingLines_CheckedChanged( sender AsSystem.Object, e AsSystem.EventArgs)
HandlesckDrawMatchingLines.CheckedChanged
    If(rdoImageFile.Checked = True) Then
        'if using image from file
        btnPerformSURFOrGetImageToTrack_Click(NewObject(), NewEventArgs())
        'call SURF button click event to update image for draw matching lines check box change
    EndIf
EndSub

.....

SubPerformSURFDetectionAndUpdateGUI(sender AsObject, arg AsEventArgs)
    If(rdoImageFile.Checked = True) Then
        'if
        getting still image from file
        If(blnImageSceneLoaded = FalseOrblnImageToFindLoaded =
FalseOrimgSceneColorIsNothingOrimgToFindColorIsNothing) Then
            'if either flag variable indicates we do
            not have image or either is null
            Me.Text = "either or both images are not loaded or null, please choose both images before
            performing SURF"
            'show
            error message on title bar
            Return
        EndIf
        'and bail
        Me.Text = "processing, please wait . . ."
        'show processing message on title bar
        Application.DoEvents()
        'call DoEvents() to update form
        stopwatch.Restart()
        'and start the
        stopwatch
        Elseif(rdoWebcam.Checked = True) Then
            'else if
            getting image from webcam
            Try

```

```

        'try
        imgSceneColor = capWebcam.QueryFrame()
        'to get next frame
    Catch ex As Exception
        'if not successful
        Me.Text = ex.Message
        'show error message on title
    Return
    'and bail
EndTry
If(imgSceneColorIsNothing) Then
    'if next frame was not read successfully from webcam into scene image
    Me.Text = "error, imgSceneColor Is Nothing"
    'show error message
    Return
    'and bail
EndIf
If(imgToFindColorIsNothing) Then
    'if we do not have an image to track yet,
    ibResult.Image = imgSceneColor
    'show scene image on image box
    Return
    'and bail, note that this is not necessarily an error , the user may have not chosen an image to track yet
EndIf
EndIf
'if we get here, we know both color
images are good, we can begin the SURF detection stuff ...
Dim surfDetector As SURFDetector = New SURFDetector(500, False)
'initiate SURF object, params are threshold and extended flag
Dim imgSceneGray As Image(Of Gray, Byte) = Nothing
'grayscale image scene
Dim imgToFindGray As Image(Of Gray, Byte) = Nothing
'grayscale image to find
Dim vkpSceneKeyPoints As VectorOfKeyPoint
'vector of key points in scene image
Dim vkpToFindKeyPoints As VectorOfKeyPoint
'vector of key points in to find image
Dim mtxSceneDescriptors As Matrix(Of Single)
'matrix of descriptors to be queried for nearest neighbors
Dim mtxToFindDescriptors As Matrix(Of Single)
'matrix of descriptors for to find image
Dim mtxMatchIndices As Matrix(Of Integer)
'matrix of descriptor indices, will result from training descriptors (call to
KnnMatch())
Dim mtxDistance As Matrix(Of Single)
'matrix of distance values, will result from training descriptors
(call to KnnMatch())

```

`DimmtxMaskAsMatrix(OfByte)`

'both input and output to function

`VoteForUniqueness()`, indicates which row is valid for the matches

`DimbruteForceMatcherAsBruteForceMatcher(OfSingle)`

'for each descriptor in the

first set, this matcher finds the closest descriptor in the second set by trying each one

`DimhomographyMatrixAsHomographyMatrix = Nothing`

'used for

`ProjectPoints()` function to set location of found image in scene image

`DimintKNumNearestNeighborsAsInteger = 2`

'k, number of nearest neighbors to search for

`DimdblUniquenessThresholdAsDouble = 0.8`

'the distance difference ratio for a match to be considered unique

`DimintNumNonZeroElementsAsInteger`

'used as return

value for number of non-zero elements both in matrix mask, and also from call to `GetHomographyMatrixFromMatchedFeatures()`

`VoteForSizeAndOrientation()`

`DimdblScaleIncrementAsDouble = 1.5`

'determines the difference in scale for neighboring bins

`DimintRotationBinsAsInteger = 20`

'number

of bins for rotation out of 360 deg, for example if set to 20, each bin covers 18 deg ($20 * 18 = 360$)

`DimdblRansacReprojectionThresholdAsDouble = 2.0`

'for use with

`GetHomographyMatrixFromMatchedFeatures()`, the maximum allowed reprojection error to treat a point pair as an inlier

`DimrectImageToFindAs Rectangle`

'rectangle encompassing the entire image to find

`DimptfPointsFAsPointF()`

'4 points defining box around location of found image in scene, in float type (in VB, "Single" => float)

`DimptPointsAsPoint()`

'4 points defining box around location of found image in scene, in integer type

`imgSceneGray = imgSceneColor.Convert(Of Gray, Byte)()`

'convert scene color image to grayscale

`imgToFindGray = imgToFindColor.Convert(Of Gray, Byte)()`

'convert

to find color image to grayscale

`vkpSceneKeyPoints = surfDetector.DetectKeyPointsRaw(imgSceneGray, Nothing)`

'detect the key points in the

scene, 2nd param is a mask, use Nothing if not needed

`mtxSceneDescriptors = surfDetector.ComputeDescriptorsRaw(imgSceneGray, Nothing, vkpSceneKeyPoints)`

'compute scene descriptor, params are scene image, mask, and image scene key points

`vkpToFindKeyPoints = surfDetector.DetectKeyPointsRaw(imgToFindGray, Nothing)`

'detect the key points in the

image to find, 2nd param is a mask, use Nothing if not needed

`mtxToFindDescriptors = surfDetector.ComputeDescriptorsRaw(imgToFindGray, Nothing,`

`vkpToFindKeyPoints)` 'compute to find descriptor, params are to find image, mask, and image to find key points

`bruteForceMatcher = NewBruteForceMatcher(OfSingle)(DistanceType.L2)`

'instantiate brute force matcher object with L2, squared Euclidean distance

`bruteForceMatcher.Add(mtxToFindDescriptors)`

'add matrix for to find

descriptors to brute force matcher

```
mtxMatchIndices = NewMatrix(Of Integer)(mtxSceneDescriptors.Rows, intKNumNearestNeighbors)
'instantiate the indices matrix, params are rows and columns
mtxDistance = NewMatrix(Of Single)(mtxSceneDescriptors.Rows, intKNumNearestNeighbors)
'instantiate the distance matrix, params are rows and columns

bruteForceMatcher.KnnMatch(mtxSceneDescriptors, mtxMatchIndices, mtxDistance,
intKNumNearestNeighbors, Nothing) 'find the k-nearest match, last param is a mask (use Nothing if not needed)

mtxMask = NewMatrix(Of Byte)(mtxDistance.Rows, 1)
'instantiate matrix mask
mtxMask.SetValue(255)
'set value

of all elements in mask matrix

Features2DToolbox.VoteForUniqueness(mtxDistance, dblUniquenessThreshold, mtxMask) 'filter the
matched features, such that if a match is not unique, it is rejected

intNumNonZeroElements = CvInvoke.cvCountNonZero(mtxMask)
'get number of non-zero elements in mask matrix
If(intNumNonZeroElements >= 4) Then
'if at least 4,
'eliminate the
matched features whose scale and rotation do not agree with the majority's scale and rotation
intNumNonZeroElements = Features2DToolbox.VoteForSizeAndOrientation(vkpToFindKeyPoints,
vkpSceneKeyPoints, mtxMatchIndices, mtxMask, dblScaleIncrement, intRotationBins)

If(intNumNonZeroElements >= 4) Then 'if still at least 4
non-zero elements,
'get the
homography matrix using RANSAC (Random Sample Consensus)
homographyMatrix =
Features2DToolbox.GetHomographyMatrixFromMatchedFeatures(vkpToFindKeyPoints, vkpSceneKeyPoints,
mtxMatchIndices, mtxMask, dblRansacReprojectionThreshold)
EndIf

EndIf

imgCopyOfImageToFindWithBorder = imgToFindColor.Copy() 'make a copy of
the image to find, so we can draw on the copy, therefore no changing the original image to find

'draw a 2 pixel wide border around the copy of the image to find,
use same color as box for found image
imgCopyOfImageToFindWithBorder.Draw(New Rectangle(1, 1, imgCopyOfImageToFindWithBorder.Width - 3,
imgCopyOfImageToFindWithBorder.Height - 3), bgrFoundImageColor, 2)

'next we will draw the scene image and image to find together in
the result image

'3 conditionals are necessary, depending on which check boxes are
checked (draw key points and / or draw matching lines)
If(ckDrawKeyPoints.Checked = True And ckDrawMatchingLines.Checked = True) Then
'if drawing both key points & matching lines on result image,
```

'useDrawMatches function, combines scene image and copy of image to find into result, then draws key points and matching lines all in one step

```
imgResult = Features2DToolbox.DrawMatches(imgCopyOfImageToFindWithBorder, _
```

```
vkpToFindKeyPoints, _
```

```
imgSceneColor, _
```

```
vkpSceneKeyPoints, _
```

```
mtxMatchIndices, _
```

```
bgrMatchingLinesColor, _
```

```
bgrKeyPointsColor, _
```

```
mtxMask, _
```

```
Features2DToolbox.KeypointDrawType.DEFAULT)
```

```
Elseif(ckDrawKeyPoints.Checked = TrueAndckDrawMatchingLines.Checked = False) Then
```

```
'if drawing key points but not matching lines on result image,
```

```
'draw scene with key points on result image,
```

```
imgResult = Features2DToolbox.DrawKeypoints(imgSceneColor, vkpSceneKeyPoints,  
bgrKeyPointsColor, Features2DToolbox.KeypointDrawType.DEFAULT)
```

```
'then draw key points on copy of image to find,
```

```
imgCopyOfImageToFindWithBorder =
```

```
Features2DToolbox.DrawKeypoints(imgCopyOfImageToFindWithBorder, vkpToFindKeyPoints, bgrKeyPointsColor,  
Features2DToolbox.KeypointDrawType.DEFAULT)
```

```
imgResult = imgResult.ConcateHorizontal(imgCopyOfImageToFindWithBorder)
```

```
'then concatenate copy of image to find onto result image
```

```
Elseif(ckDrawKeyPoints.Checked = FalseAndckDrawMatchingLines.Checked = False) Then
```

```
'if not drawing key points or matching lines,
```

```
imgResult = imgSceneColor
```

```
'assign scene image to result image
```

```
imgResult = imgResult.ConcateHorizontal(imgCopyOfImageToFindWithBorder)
```

```
'then concatenate copy of image to find onto result image
```

```
Else
```

```
'should never get here
```

```
EndIf
```

If(homographyMatrixIsNotNothing) Then 'check to make sure homographyMatrix is not null, this is necessary b/c in a moment we will call a homographyMatrix function

'in this next portion we draw a border on the scene portion of the result image, around where the found object is located

```
rectImageToFind.X = 0
```

```

rectImageToFind.Y = 0
rectImageToFind.Width = imgToFindGray.Width
rectImageToFind.Height = imgToFindGray.Height

'next, instantiate an array of PointF's corresponding to the rectangle
ptfPointsF = New PointF() { New PointF(rectImageToFind.Left, rectImageToFind.Top), _
    New PointF(rectImageToFind.Right, rectImageToFind.Top), _
    New PointF(rectImageToFind.Right, rectImageToFind.Bottom), _
    New PointF(rectImageToFind.Left, rectImageToFind.Bottom) }

'ProjectPoints() will set ptfPointsF (pass by reference) to be the
homographyMatrix.ProjectPoints(ptfPointsF)
scene portion of the result image, where the box surrounds the image we are looking for

'convert from type PointF() to type Point(), this
is necessary at this step b/c ProjectPoints() takes type PointF() but DrawPolyline takes type Point()
ptPoints = New Point() { Point.Round(ptfPointsF(0)), _
    Point.Round(ptfPointsF(1)), _
    Point.Round(ptfPointsF(2)), _
    Point.Round(ptfPointsF(3)) }

imgResult.DrawPolyline(ptPoints, True, bgrFoundImageColor, 2)
border around found image in scene portion of result image
EndIf

detection stuff, now we update the form . . .
ibResult.Image = imgResult
'assign resulting image to image box on form

If(rdoImageFile.Checked = True) Then
image from file,
stopwatch.Stop()
'stop the stopwatch and update the form title bar with the time
Me.Text = "processing time = " + stopwatch.Elapsed.TotalSeconds.ToString() + " sec, done processing,
choose another image if desired"
EndIf
EndSub
EndClass

```

'to start with, set the rectangle to be
'the full size of the image to find

'location of a box in the

'draw

'end of the SURF

'if using

B.SCREENSHOTS

