IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 15, Issue 5 (Nov. - Dec. 2013), PP 61-68
www.iosrjournals.org

A Novel approach for Graphical User Interface development and real time Object and Face Tracking using Image Processing and Computer Vision Techniques implemented in MATLAB

Satyajit Mahanta[1], Souvik Ghosal[2], Partha Das[3], Aditi Datta[4], SauravDebnath[5], Sauvik Das Gupta[6]

[12345] *West Bengal University of Technology, Kolkata, West Bengal, India*
[6] *Schoolof Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA*

Abstract:Image Processing is an advancement in the Signal processing domain where the input is an image or video and output can be image, video or in any other forms. With daily advances in Human-Computer interaction, Image Processing, along with computer vision has become an integral part of any intelligent system. They are mainly based on the workings of the visual perception function of the human brain. As such most of the algorithms are designed based around this central concept. The major application of Computer Vision is in the field of intelligent machines and robotic visions, where machines are made intelligent by being able to sense their surrounding visually and optimize themselves accordingly. In case of robots, through computer vision, they are made capable of sensing their surroundings, just like a human.

Index Terms: Image processing,GUI, Object Tracking, Face Tracking, Edge Detection.

I. Introduction

In Image Processing or Computer Vision, the basic idea is to mimic the human visual perception powers and try to find the mathematic models behind them [1] [2] [3]. Having achieved this, the next challenge is to implement these models into viable real world systems and applications. In this paper, we try and implement object detection & tracking along with Face Detection and tracking in real-time.We first start off with detecting and tracking of objects and gradually expand this system to controlling a real world object, here the iRobot Create. Though multiple algorithms have been developed for detecting objects, we have decided to base our approach on the Colour based segmentation. Through this we detect an object and its centroid and over the course of multiple frames, track it, based on its colour. We then further expand and apply the findings to real-world systems (Arduino and iRobot Create) and end up with the ability to control, wirelessly. We then develop the system further to allow us to have complete control of the robot as the user tend to change its position the robot moves simultaneously in that direction.

We then moved onto Computer Vision wherein we implemented the Face Detection and tracking using the highly efficient Viola-Jones [4] [5] Algorithm. For the tracking part, we first detected the face and based on that and the skin tone information of the nose, which is the only part of the face having the least number of background pixels; we implemented a CAMShift [6] algorithm to track the face over time. After the initial findings, we expanded the system to be capable of detecting and tracking multiple faces.

Lastly, we have gathered and implemented all our findings and systems in a unified GUI [7] in order to give the user a graphical user interfacefor ease of accessibility of the advanced functions.

II. GUI for Manipulating Images

GUI (Graphical User Interface) is a very popular Point-and-click control of a software application; it is a well-known design environment which is very effective as the user does not need to know the programing language (in which the software was built) in order to use the software. Thus, GUI is such an interface that can literally be handled by anybody. In this article, we are going to discuss about a GUI which we have created in the MATLAB's Graphics User Interface (GUI) environment and we will also discuss briefly how to make such a GUI in MATLAB environment. This GUI design is all about the editing of pictures. The overview of this GUI is given below but before that the procedure to make a GUI is mentioned below, in a nutshell.

2.1:Making a GUI in MATLAB – In MATLAB, There are vividly four kind of way to make a GUI, namely Basic Graphics, Animation, Handle Graphics Objects and Creation of GUI using Guide. Here we will be demonstrating about the GUI creation by using 'GUIDE' command. This is a command which will fetch us to a window to make a new GUI or to open any existing GUI. Let us make a curt demonstration to make anew GUI.

Below in figure 1 is a chart about the various ways of making GUI in MATLAB:

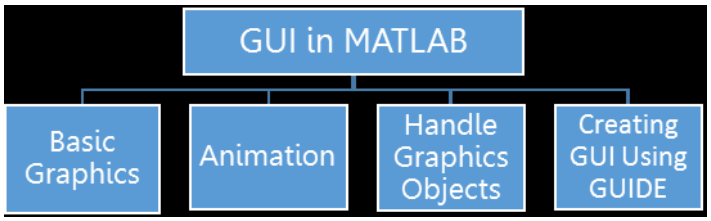

Figure1: Basic ways to implement a GUI

Here, we are going to use the last one i.e. 'Creating GUI using GUIDE' to make a new one. There are some typical steps involved in the process of making a GUI. The diagram shown below demarcatesthose typical steps which are to be followed:

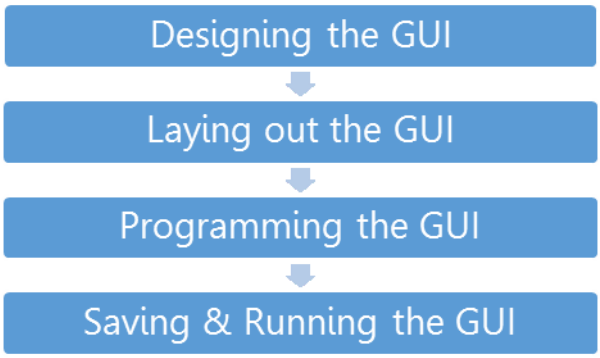

Figure 2: Basic steps to make one GUI by using GUIDE in MATLAB

Now, let us illustrate what these steps really mean. At first, the Step is "Designing the GUI". Here user must have a clear mental picture of what he or she is going to design and accordingly the user must start the design right away. Next comes "Laying out the GUI" [8] [9]. In this step the user will create a layout of his or her preferred design of the GUI in the Layout Editor. Then comes the "Programming the GUI". Now, in this step user must create some logic for their own preferences and accordingly write that logic in the programing language of MATLAB. The last step which is "Saving & Running the GUI". Now, there is one thing that is to be remembered at the time of making a GUI that the design that the user will do in the Layout Editor will be shown as the external interface and there are buttons, sliders and many more things that can be incorporated in the Design. These various buttons are available on the due left corner of the Layout Editor. The user of the GUI will not use any programming language to make any particular button active. Instead, the programming should be under the title of the Button such that when the Button is pressed, it commands MATLAB to run the program which is written under that function.

2.2 :Details - *about the GUI*-We have used several functions and operations in our design. Firstly there are few buttons (technically known as Push Button) available at the top left corner of our design, namely 'Resize' (to resize an image to users desired resolution), Noise, Image adjust, Rotate, Blend etc. The operations that we have incorporated in our design are Morphological Operation, Image Segmentation etc. Here few of them are discussed in the sense of giving an idea about what these functions are and what they do. Firstly we have shown a screen shot of our design below.In this design we have shown the tweaking of the Red, Green and Blue (technically known as RGB) values of a sample image (left), and the result image (right).

Figure 3: The GUI design at the time of real time RGB value tweaking operation on images

Next is the 'Blend' Pushbutton. This is a button by clicking which we can add any two kinds (resolution wise) of picture over one and another, i.e. they get merged up together. First the user has to load one picture to the GUI main frame and this will be assigned as the 1st image, on which a second image will be blended. Then he/she can press this button so that a window will appear to load another image. We have designed this function in such a way that when the image is not equal in dimension with the first one, then firstly the resizing of image will be done according to the resolution of the first image and then the merging operation will be done. Here two distinct images and their blended form are shown below.

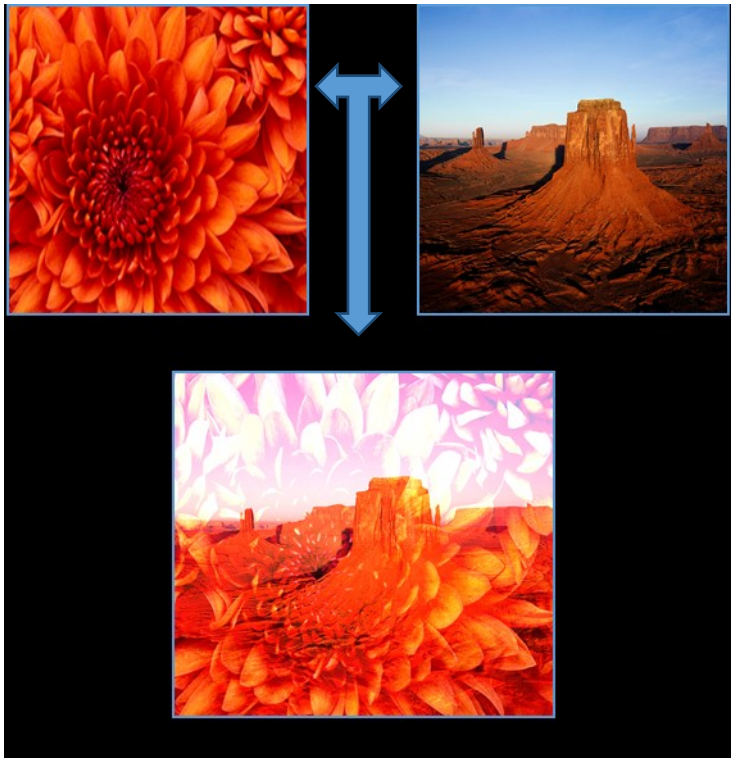

Figure4: The Blend Operation

Another very useful feature of Image Processing that has been integrated in our GUI,is the ability to find edges in a given image [10]. This functionality is widely used in many Computer vision techniques. As such there are multiple algorithms that are used for finding edges. In our GUI, we have included most of the available algorithm. But during our workings, we found that since the algorithms work only with Black & White images, there is a considerable amount of information loss when we need to find edges from a real world or color image, as it is converted to the Black & White image. In order to reduce this information loss, we have implemented a simple new algorithm, based on the original algorithms, to find edges for color images, as shown in the flow diagram (Figure 5). As can be seen from Figure 6: edges of objects that were lost in the general approach are quite visible through the new approach. The weakest edges are denoted by Red, followed by Green & Blue. The most prominent edges are marked with white. These can be used or discarded (Figure 8) as required, by a simple thresh-holding on the component.

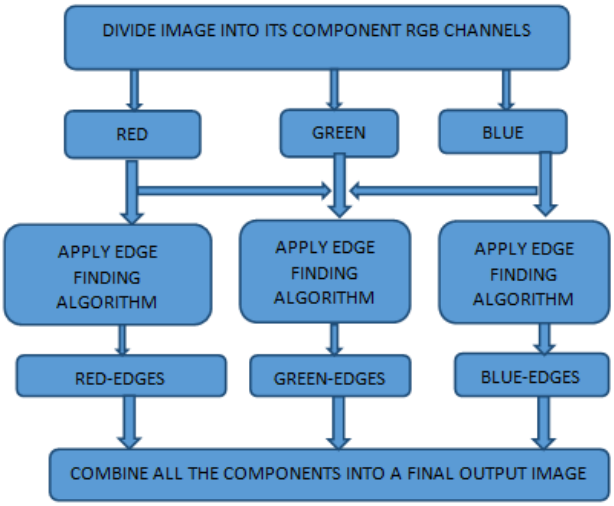

Figure 5: Proposed Algorithm of the new Edge Detection

Figure 6: Original Color Image

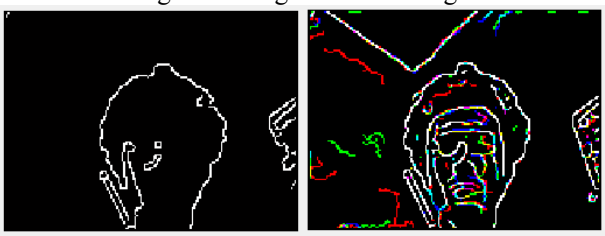

Figure7: (*left*) Result after normal Edge Detection (Canny) (*right*) Result after the new Edge Detection algorithm

Figure 8: Applying thresholding after the new Edge Detection algorithm to discard the weak edges

III. Image Processing

3.1: Object Detection & Tracking- Like any living things, in order to track an object, we first need to identify and detect that object. There are many approaches to this. We base our approach on the colour of the object. For this, we first initialize the camera, the primary input of our processing system, to start grabbing images and return it in the RGB colour space. Once this is done, we then set up a loop that iterate through each frame of the input stream, since a video is essentially a stream of still images. With each of these grabbed frames, we first convert them to grey scale & then we segment the colour channel from it. After this we apply filters to remove the noise & hence minimize errors due to external conditions. Having done this, we find the area of the objects, which are now only the objects with the required colours. Once done, we obtain an array containing descriptions of the centroid and area of all the objects with the desired colour. We then use this array to draw a box around the detected object(s). Since we are finding the centroid for the objects in every frame, the object is tracked along with its motion over the entire duration of the stream.

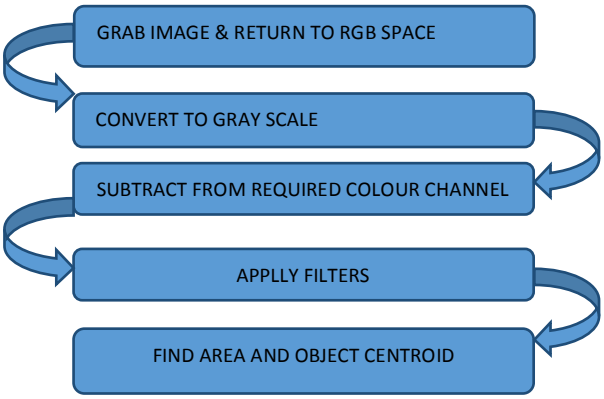

Figure 9: Proposed scheme for Object detection & tracking

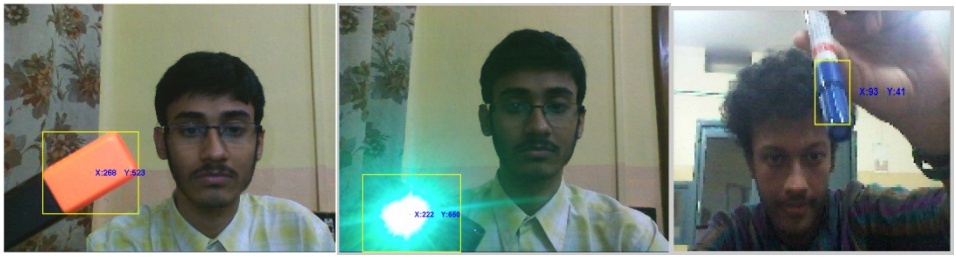

Figure 10: Objects being detected & tracked

3.2: Object Based Control- Once we are able to detect and track an object, we can easily use the data so obtained to control real world systems. For the preliminary stage, we chose a basic Arduino board with an LED. The idea was to make a LED light up when it was in a given range on the X-Y axis and turn off when it went out of the range. We used the differential of the previous and present coordinates of the centroid of the tracked object. By analysing the result from this operation we can detect whether object was moved to the left or right or to the top or bottom. For example if the difference in the X Axis was greater than the Y Axis, then we can surely say that the object was moved in the X-Axis & vice-versa. A value within a small threshold is ignored in order to prevent the system from becoming too sensitive to motion and thus responding to even slight hand jerks.

3.3: Object Based control of iRobot – Once the control algorithm has been tested; we applied the findings to control iRobot over a Bluetooth Access Module (BAM), which gives us a huge range of wireless operability. Just like with the arduino board, we detected the motion using the same algorithm (figure 11) and for movement along Y-Axis, we made it turn clockwise/anti-clockwise, while for X-Axis, we made it go forward and backward, thus detecting 2-D movement of the object. We then further expanded the dimension of the iRobot movement by tracking the area too. If the area increases, then the object is coming closer to the camera, while the decrease signifies the object moving away from the camera in the Z-Axis. With this in mind we made the iRobot back away when the object moved closer and come closer when the object moved away from the camera, thus allowing a 3-D motion control of the iRobot.

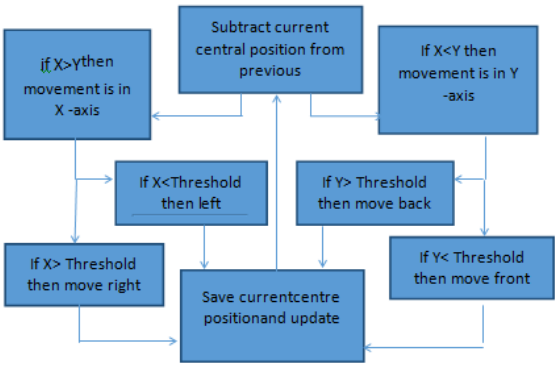

Figure 11: Proposed scheme for controlling iRobot through Object Detection

3.4: Object Motion Mimicking – After controlling the iRobot in three dimensions, we further expanded the capability by making the robot mimic the object's movement, thus freeing it from rigid straight line movements. As such, the robot's movement follows the pattern that is plotted by the movement of the object. The basic idea behind the algorithm is that we can describe each movement in a Cartesian plane with coordinates & when it is translated to the real world, it can be described by the change of angle along with forward, backward or sideways movements. Thus, like the previous algorithm, we detect which direction the object is moving and along with that we also calculate the angle at which the object is moved. This can be easily found by calculating the slope between the two points of the centroid and normalizing it to a value between π to $-\pi$ degrees. Like before we also keep a threshold here to ignore the unwanted jerks of the hands. We then make the iRobot turn the calculated angle along with the direction of the movement, thus mimicking the motion of the object in the real world.

3.5 :Following a map – From all these, we also observed that we can make the iRobot follow a map or pattern in the real world based on a map or pattern drawn as an image. The basic idea is that like the mimicking algorithm, the iRobot will follow a pattern, but instead of the input being from the object's movement, it will be a pre-drawn pattern in a binary image. Another zero initialized matrix (reference) with the same dimension as the image is maintained to track looping. The algorithm is such that, from the starting point, the program will sample its next pixel, which can be in front or to the sides. If the one of the pixels is set as 1 and the corresponding pixel is not set as 1 in the reference, then the iRobot will move in that direction. When the iRobot moves, it updates the corresponding pixel in the reference too. The entire process is repeated until the end of the map is found (when there are no more pixels to move forward to) or the pixel which has already been traversed (using the reference) is met. If the robot encounters a cross-section (overlapping of path), it randomly decides on a direction and updates. The next time it comes, due to the reference being updated, it will only have one way to go. Thus the robot is made to follow a given map/pattern even with loops. The algorithm was further expanded by making the reference increase the weights it deposits on a traversed path, like an ant. This can track the number of loop the robot is required to perform. Also, a scale factor is kept for the map to real world distance scaling. This factor can be tweaked to indicate how much distance of the real world each pixel should designate.

IV. Face Detection

4.1: Viola-Jones Algorithm **-**In 2001, Paul Viola and Michael Jones proposed a framework for an algorithm called the Viola-Jones algorithmthat used a variant of Adaboost, Haar Features and Integral Image to detect faces in any given image. This algorithm can also be used to detect other parts of the face like eyes, nose, upper body, etc. Although this algorithm was motivated by the problem of Face Detection, it is capable of being used as an object detector and hence this algorithm can be trained to detect a wide variety of objects too.

4.2: Methodology - We have used Viola-Jones algorithm forfacedetection to implement a real time face detection system. As illustrated in the flow chart in Figure12, we first initialize the cascade object detector to detect a face. There are various ways to detect a face using the Viola-Jones algorithm. The algorithm can be initialized to detect different parts of the face, like eyes, nose, upper body etc. Since the face is the part which is distinguished mainly by the skin tone, we base our approach on this method and as such we first convert the captured image to the Hue channel, as this gives us the skin tone information and much better detection rate. Once the face is detected, our next objective is to track. Since we are using computer vision, we implemented the Histogram Based tracker, that tracks an object based on the pixel density over time, to track the subject(s)' face. Again, in order to get better and constant tracking results, we first extract the face and detect the nose in it using Viola-Jones Algorithm, all using the Hue channel data. We specifically use the nose instead of the face to track the face because in a human face, the nose is the most prominent part. In addition, the nose has the lowest number of background pixels which has the least variance due to external conditions, like illumination. This can be easily seen in the skin tone information obtained through the Hue channel data. Once the nose is detected, we initialize the tracker with this and extrapolate the tracked information to account for the face. We even further expanded the system to detect multiple faces, using the same principle but looped many times over. The system so developed was integrated in to the GUI that has been developed and discussed in the first part of this paper. The Figure 13 shows the system detecting & tracking multiple faces at once.

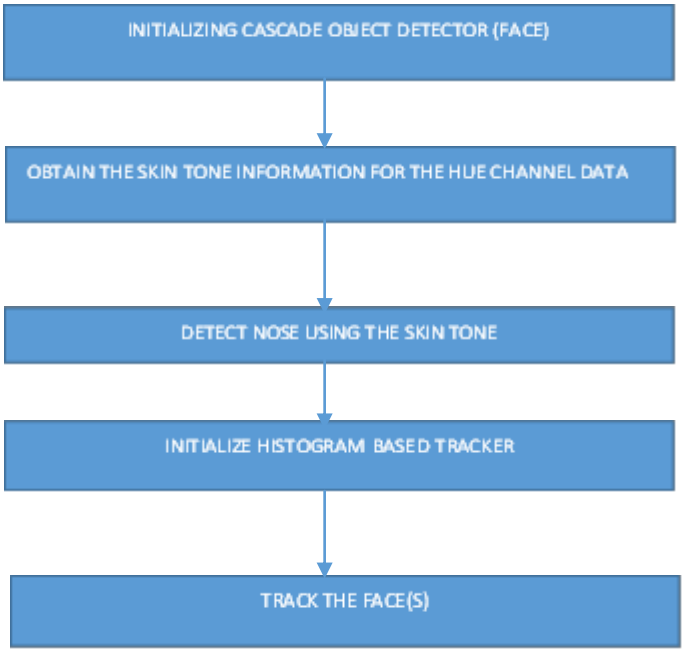

Figure 12: Proposed method for Face Detection & Tracking

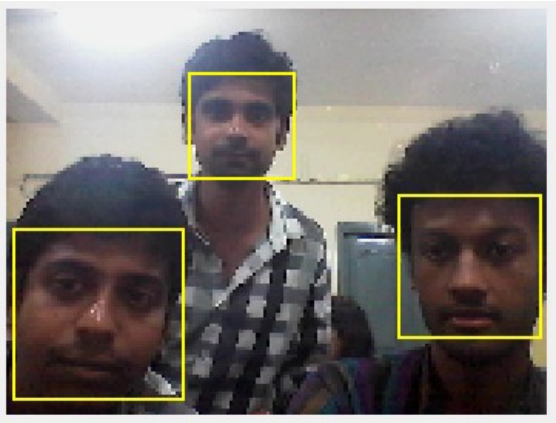

Figure 13: Multiple faces being detected and tracked simultaneously

V. Discussion & Future Prospects

In the paper, we have seen and implemented many functions in the GUI which are widely used in the field of image processing and we also tried to make some improvements in some of the functions. However, all of the functions could not be shown in the one screen shot as there are other sub-steps involved in this process. Image Processing opens up a wide variety of Human-Computer interaction that can only be limited by the human imagination. The works that we have shown here can easily be expanded to various real world situations. For example, the iRobot function of following the map can easily be embedded and made into a robot that can perform tasks like a peon in a disaster stricken area where normal transportations are hindered. The topography of the area can be easily found out and mapped using conventional method and then using this approach an automated delivery or even rescue system can be setup quickly, by chalking out a map based on the topography, for the robot to follow. Again the motion mimicking part can be integrated here to make the entire process real time and thus help guide people within a, for example, collapsed building/area, by guiding the victims by drawing the escape path and making a guide robot follow it. This is but just one way of implementing the work we have done here in the real world. The algorithms, though, that have been implemented still does have room for improvements. For example, in the basic object detection model, instead of scanning the entire image for the next position of the tracked image, we could set up a probabilistic boundary in which to search for the next position. If the object is not found there, we scan the entire image. This can result in an even better, faster and efficient detection and tracking algorithm.

Along the same line, we have Face Detection and tracking. Needless to say, this has an ever growing demand in the field of security. But, as more powerful hardware are getting miniaturized face detection and tracking are also creeping into our daily lives. For example, smart TVs and electronics can be more efficient by automatically tracking and thus regulating themselves for efficient management of energy. They can for example turn themselves off or go into a sleep mode when they can't see any person in their field of service. Also, advanced gesture recognition systems can be developed using facial features detection. One example, out of many such possibilities, is that a user might control robots or other electronics with just the movement of their heads, which the electronics can track. It is only a matter of time, before we will have electronics that will actually give us personalized services based on who we are.

Acknowledgements

We would like to thank ESL, www.eschoollearning.net, Kolkata for giving us a platform to work on this project.

References

[1] Richard Szeliski,Computer Vision algorithms and applications, (2010)
[2] R. Fisher , K Dawson-Howe, A. Fitzgibbon, C.Robertson, E. Trucco, John Wiley, *Dictionary of Computer Vision and Image Processing*, 2005
[3] F. Tanner, B. Colder, C. Pullen, D. Heagy, C. Oertel, & P. Sallee, *Overhead Imagery Research Data Set (OIRDS) – an annotated data library and tools to aid in the development of computer vision algorithms*, June 2009
[4] Liming Wang, Jianbosh Shi, Gang Song, I-fan Shen, Object Detection Combining recognition and Segmentation, Fudan University, Shanghai, PRC, 200433, obj_det_liming_accv07.pdf
[5] Viola, Jones: Robust Real-time Object Detection, IJCV 2001,http://en.wikipedia.org/wiki/Caltech_101#cite_note-Viola_Jones-
[6] D. Comaniciu, V. Ramesh and P. Meer, Real-time tracking of non-rigid objects using mean shift. In Proc. Conf. Comp. Vision Pattern Rec., pages II: 142–149, Hilton Head, SC, and June 2000
[7] Scott T. Smith, MATLAB Advanced GUI Development, Dog Ear Publication, published (2006)
[8] Yair Moshe, Signal and Image Processing Laboratory, GUI with MATLAB, Published May 2004
[9] Rafael C. Gonzalez, Richard E. Woods and Steven L, Eddins, Digital Image Processing using MATLAB, (2004)
[10] Scott E Umbaugh, Digital image processing and analysis: human and computer vision applications with CVIPTools, segmentation and edge like detection, CRC press, November 19, 2010