Design Flow for Complex Scene Visual Object Tracking and Avoidance System

Thomas Gonnot and Jafar Saniie
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, Illinois, USA

***Abstract*— This paper presents the design flow of efficient and real-time object detection and tracking system in a complex environment. The design relies on color extraction, target recognition using neural networks and feature extraction. Furthermore, the system is designed to identify different objects and also to predict their movements. There are many domains where the detection and tracking of objects can be required. An example is a defense system for perimeter protection where the camera is more cost effective in the case of short range detection and recognition. Another application of the proposed system is the detection of the obstacles in front of a visually impaired person using cameras mounted on the frame of their eyeglasses.**

I. Introduction

Radar systems have always been preferred for tracking objects, either in the air or in the sea, because of their extended range – up to 3000 km [1]. But for short distance tracking, the use of such a system has to be reconsidered. The radar is fast to give the distance of an object in one direction. However, it suffers a slow scanning speed because the unit has to be mounted on a rotating base. This can be problematic for the tracking of an object in a short range where the object is moving fast and the delay between two updates gives a poor tracking accuracy. Another limitation of radar systems is that they need to send electromagnetic bursts to the target which can be deflected, or can yield the position of the radar interrogating system.

The camera as a mean of target detection has many advantages over radar. The first and foremost advantage is that the camera-based system is entirely passive: no burst of electromagnetic waves is used for the detection, and consequently, it is more difficult to counter the detection. It can also be faster to reveal the target position compared to radar: cameras can acquire more than 60 frames per second. Some cameras can even acquire the image in 360°. There are also a number of different camera types, including the standard visual spectrum, the thermal imaging [2], or simply very sensitive cameras enabling the tracking even during night or bad weather conditions. Finally, image processing can be achieved using massive parallel processing for higher detection and tracking performance [3]. Furthermore, it is more difficult to make an object invisible to the camera. At the present time, the existing decoys won't affect the visible spectrum and hence do not counter the detection capability of cameras. Another important aspect of a target tracking system is the cost. Radar systems are expensive units, and not cost effective for ordinary target detection and tracking purposes. However, the cost advantage goes to the cameras, because a network of low resolution cameras has the ability to precisely locate the target [4]. In the case of assisting visually impaired people, the cameras can be more practical when it is integrated into the frame of their eyeglasses.

This paper focuses on the implementation of efficient object recognition (Section II), tracking algorithms (Section III) and object collision avoidance (Section IV). Figure 1 shows a merging of the different algorithms described in this paper for object tracking and avoidance. The real-time implementation of these algorithms is based on the color, the shape, and the features of a moving or static object.

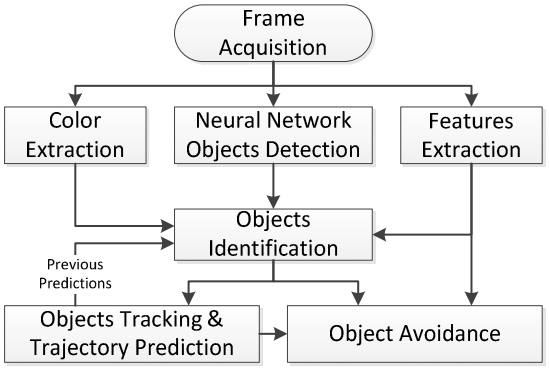

Figure 1. Merging of the different image processing algorithms for visual object tracking and avoidance system.

II. Object Recognition Algorithm

In this study, the image acquisition is accomplished through OpenCV which provides the basic input/output operations, and implements various image analysis algorithms and image display functions [5]. In this study, object detection

and recognition are designed based on the following algorithms:

- The color extraction algorithm
- The neural network based detection algorithm
- The feature extraction and matching algorithm

The color extraction algorithm is performed using the HSV (Hue-Saturation-Value) color space shown in Figure 2. This color space enables to code the color on one plane instead of three planes for RGB (Red-Green-Blue) or YUV (color space in terms of one luma (Y) and two chrominance (UV) components). The HSV colors are less affected by the light intensity or by the shadows. Another advantage of the HSV color space is that the S-plane corresponds to the intensity of the color. By thresholding the S-plane, one can discard the image region such as the walls, the light source, and the background.

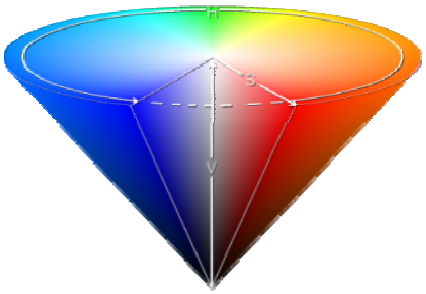

Figure 2. Representation of the HSV space as a cone [6].

A binary image map can be generated by thresholding the Hue and Saturation planes. Then morphological operations are applied to this binary image map to remove the noise. Scanning of the rows and columns determines the position and size of each object in the binary image map. This approach may generate ghost zones where there is no object of interest. As shown in Figure 3, through an iterative process, the false detection of invalid zones can be eliminated. This algorithm is limited to detect only the color of the objects but not their shape and size.

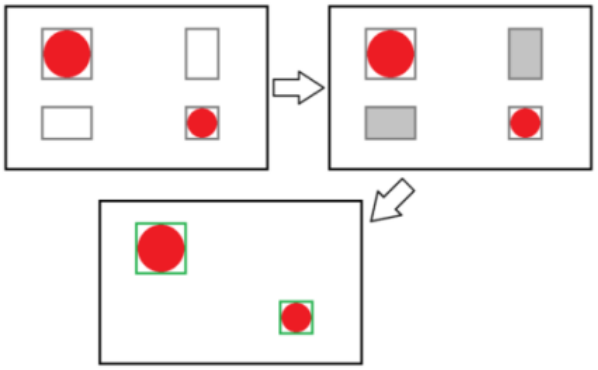

Figure 3. Example of scanning of the binary map.

The second approach for image analysis is based on a neural network to detect and identify the objects. In this method, the image is divided in subparts, with the size defined by the design of the neural network. Each subpart of the image is processed by the neural network and the inner layers of neurons extract the information about the edges and the geometry of the objects. A technique called AdaBoost can be used to reduce the number of image features for the neural network in order to decrease the processing time [7]. The output of the neural network is a single value that is used to provide a 'yes' or 'no' answer for object recognition. Once all the subparts have been processed by the AdaBoost method [7], the image size is then reduced and the process undergoes a new iteration.

The neural network is not very practical for real-time recognition of a new object because of the number of samples required for an acceptable success rate. For example, a neural network trained to recognize a face requires more than hundreds of pictures of faces in different conditions to obtain an accuracy of 75-90% depending on the design of the classifier [8]. In this case, with a smaller set of samples for training, false positives are highly probable. Consequently, another method and criterion needs to be realized to improve the object detection. In this case, the color algorithm can be used to detect the objects assuming that the colors of the objects are known. The flowchart of this algorithm is shown in Figure 4.

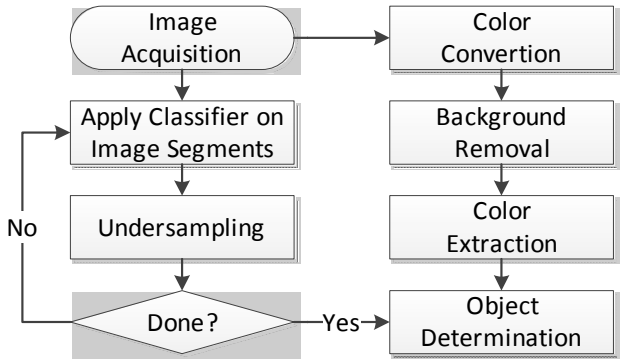

Figure 4. Hybrid neural network classifier and color detection algorithm flowchart.

The final approach is the feature extraction and feature matching pursuit using the SIFT (Scale-Invariant Feature Transform) algorithm [9] [10]. The idea behind the SIFT algorithm is to extract keypoints from an image which are minimally affected by the scale, orientation, sharpness or luminosity. The SIFT algorithm extracts the features of the image around the keypoints, generating vectors describing the image features. These vectors are then matched to another set of vectors in a database of the objects to be recognized. This method has the advantage of being more robust than the neural network and it does not require learning time.

III. Object Tracking Algorithm

After the object detection, the system obtains the position of the object in the image, but nothing about the object's movement. In order to approximate and track the movement of the object, the system must follow the object from frame to frame. Knowing the position of the object in each frame and the time lapse between the frames, it is possible to estimate the movement trajectory of the object.

One simple algorithm is proposed to predict the future position of the object. It consists of using the last three estimated positions and fit a circle on these positions (see points A, B and C) as shown in Figure 5. In this case we assume that the next object position (see point D) should be on the circle that is realized by the previous A, B and C points. We compute the position of the last point on the circle by obtaining the angles between the known points using the center of the circle, and applying the formula: $\gamma = (|\alpha| + |\beta|)/2$. The term α is the angle between the samples n-2 (point A) and n-1 (point B), and β the angle between the samples n-1 (point B) and n (point C). The resulting angle γ represents the angle between the sample n (point C) and the predicted position (point D). Even if not exact, this method yields a relatively accurate prediction for the trajectory. This prediction can be used to estimate the best matching candidate in multiple object tracking. The estimated object which is the closest to the prediction point D is considered to be the best match for the tracking. Knowledge of the trajectories of all the objects including the background allows extracting the speed of the object.

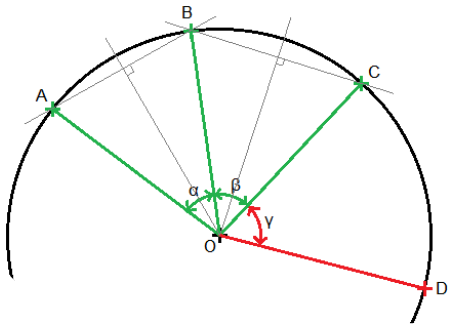

Figure 5. Representation of the trajectory prediction.

IV. Visual Avoidance Algorithm

An important aspect of the proposed system is the detection of the approaching objects in order to avoid collision. Two approaches are feasible for collision avoidance. The first method relies on a depth map estimation using a stereovision system. This method requires a calibration procedure to set the actual distance threshold for collision avoidance. As an alternative, the depth information can be extracted from a single camera video stream. In the case of a single camera system, this becomes more difficult, because the system doesn't know its distance to the object, or the distances between the objects and their actual sizes. In order to avoid object collision, the system must estimate the speed of the object and the direction of its trajectory. To achieve this, the system can use the features extracted from the image and evaluate the optical flow (also known as the motion vectors), of the image feature. Each vector is a combination of a rotation and a translation. The knowledge of the features and their motion allows to separate the translation from the rotation, and to determine a convergence point (called focus of expansion) corresponding to the direction of the object's movement. The faster the features are moving away from the focus of expansion, the closer the object is to the camera position. In order to quantify this, we use the Time-to-Crash [11], or Time-to-Collision equation, $TTC = \Delta_i / \left|\overrightarrow{V^t}\right|$ where Δ_i is the distance of a feature from the focus of expansion, and $\overrightarrow{V^t}$ is the optical flow vector of the feature. When the object's physical size is known by the system, the measurement of the distance from the object can be derived from the physical distance of the features and can be correlated to the motion vectors to predict the collision.

V. Experimental Results

In this study, we present the implementation results for the color detection algorithm, the neural network detection algorithm and the feature extraction and matching algorithm. Figure 6 shows the result after the color extraction. The background is replaced by the black color to show only the resulting extraction of the red objects. In this case, the red ball can clearly be distinguished from the background, and the light intensity differences on the ball don't appear. The left side of Figure 6 also shows the result of the tracking algorithm with the prediction.

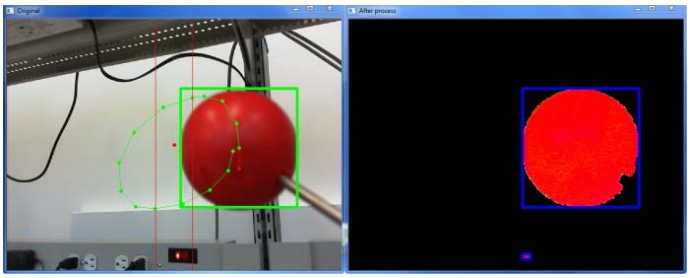

Figure 6. Result of the color extraction and the tracking algorithm.

In the case of the neural network detection algorithm, the neural network is trained to recognize a remote controlled flying helicopter. The detection results of the neural network are then correlated with the colors extracted from the image. The selected object is the one with the highest density of the desired and predetermined color inside the detected area. Figure 7 shows an example of the detection of a remote controlled helicopter. In this case, the neural network is trained with a set of 20 images of the helicopter. The figure shows some false results, indicated by a red rectangle, and the selected result, indicated by a green rectangle. The left side of Figure 7 shows the screen capture of the original picture and right side is the result of the color extraction.

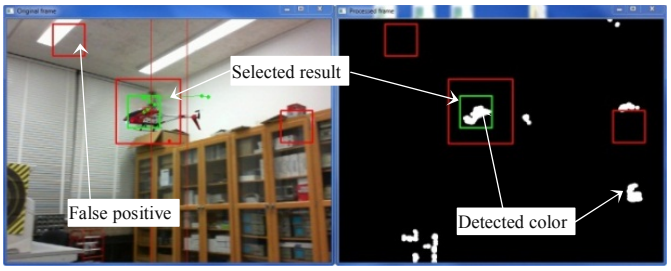

Figure 7. Actual detection of the helicopter.

For the feature extraction, the SIFT algorithm has been integrated to a program where the user is invited to select the object to track. The selected image is then analyzed and its features are stored to match the object in the next frames. An example of detection is shown

Figure 8, where we can see the features, matches and the detected position of the object inside the image.

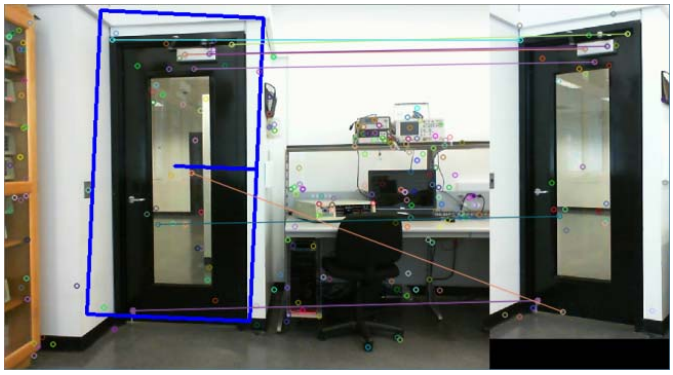

Figure 8. Example of the detection of a door in real-time. The door picture on the right side is the template.

In this case, the algorithm detects a certain number of matches, and if there are more than two matches, it determines the orientation and scale of the objects, and even discards incorrect matches when they are too different from the other matches.

The performances of the three algorithms have been evaluated using a computer equipped with a Core i7-2760QM @ 2.4GHz and with a NVIDIA GeForce 550M CUDA enabled. OpenCV version 2.4.2 was used with the Intel TBB (Threading Building Blocks) and CUDA enabled. The processing times and frame rates are listed below:

TABLE I. PROCESSING TIMES OF THE ALGORITHMS

	Color detection	***Neural network***	***Features extraction***
Average time/ frame per seconds	17.36ms/57.59fps	24.94ms/40.09fps	76.74ms/13.03fps
Minimum time/ Maximum fps	8ms/125fps	22ms/45.45fps	50ms/20fps
Maximum time/ Minimum fps	56ms/17.86fps	43ms/23.26fps	104ms/9.61fps
Standard deviation of processing time	7.54ms	3.77ms	4.86ms

As we can see in Table I, the system can execute the three algorithms in real-time, the color detection being the fastest with an average of nearly 60 frames processed per second, against the neural network algorithm, that allows about 40 frames per second.

VI. CONCLUSION

This paper presents a feasibility study for tracking an object using a camera, image processing, target detection, and recognition in real-time. The results show that tracking an object with a camera in a complex scene in terms of background, light or complexity of the object is possible with a great degree of efficiency. The algorithms presented, because of their efficiency to remove the background scene, or to extract the colors, can be used as part of a more complex system design, combining all of them in a single system for object tracking and collision avoidance.

REFERENCES

[1] S. Bren, "PROJECT JINDALEE: FROM BARE BONES TO OPERATIONAL OTHR," *The Record of the IEEE 2000 International Radar Conference,* pp. 825-830, 2000.

[2] D. Zhou, M. Dillon and E. Kwon, "Tracking-Based Deer Vehicle Collision Detection Using Thermal Imaging," *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO),* pp. 688-693, 2009.

[3] N. Zhang, Y.-s. Chen and J.-l. Wang, "Image Parallel Processing Based on GPU," *2nd International Conference on Advanced Computer Control,* pp. 367-370, 2010.

[4] K. Kim and L. S. Davis, "Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering," *Proc. Ninth European Conf. Computer Vision,* 2006.

[5] Intel, "OpenCV," 2012. [Online]. Available: http://opencv.willowgarage.com/.

[6] iLab, "HSV and H2SV Color Space," 2008. [Online]. Available: http://ilab.usc.edu/wiki/index.php/HSV_And_H2SV_Color_Space.

[7] P. Viola and M. Jones, "Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade," *International Journal of Computer Vision,* pp. 1311-1318, 2001.

[8] P. Parveen and B. Thuraisingham, "Face Recognition Using Various Classifiers: Artificial Neural Network, Linear Discriminant and Principal Component Analysis," 2006.

[9] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* pp. 91-110, 2004.

[10] T. Lindeberg, "Scale Invariant Feature Transform," 2012. [Online]. Available: http://www.scholarpedia.org/article/SIFT.

[11] N. Ancona and T. Poggio, "Optical flow from 1D correlation: Application to a simple time-to-crash detector," *Fourth International Conference on Computer Vision,* pp. 209-214, 1993.