REAL-TIME OBJECT TRACKING AND LEARNING USING TEMPLATE MATCHING

Subhodeep Roy[1], D. P. Rathod[2]

[1]*Electrical Department, Veermata Jijabai Technological Institute, Mumbai, India*
[2]*Electrical Department, Veermata Jijabai Technological Institute, Mumbai, India*

Abstract

The system that has been proposed here uses a live video stream to enable tracking, learning and detection of real time objects. The object of interest is selected in a cropping action and then it is subsequently tracked with the help of an indicator in the form of a rectangular bounding box. The process mentioned above is a long term process in the sense that the system is perpetually alert regarding the reappearance of the object after a departure from the frame or regarding the deformities in the physical appearance of the object. The proposed system uses the Template Matching algorithm to match the selected object with the 'Region of Interest' in the frame to mark the object's location. If a match is found then the Principle Component Analysis algorithm is used. PN discrimination algorithm has been proposed in this system which uses background subtraction technique to increase the speed of frame processing for object detection. This reduction in frame processing time and the reduction in average localization errors improve the template matching percentage irrespective of scaling of the input image. Thus the proposed system is expected to overcome the drawbacks of existing system which range from loss of information caused by complex shapes, rapid motion, illumination changes, scaling and projection of 3D world on 2D image, etc.

Keywords— *Template matching Tracking, Adaptive Learning, Object Detection*

----------***----------

1. INTRODUCTION

In this information age, video surveillance plays a crucial role, especially with regard to security issues in crowded public places, departmental stores, banks and also in situations such as traffic management, quality control checks in industries, etc. Tracking can be defined as a process of continuously identifying an object of interest in the video. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object.

A tracker estimates the object's motion under the assumption that the object is visible and its motion is limited. A detector performs full scanning of the image to localize all the appearances that have been observer in the past. A detector can reinitialize a tracker and thus minimize the tracking failures. Detection based algorithms estimate the object location in every frame independently. The proposed system is expected to track and learn real time objects from live video streams. The video streams are to be processed at the video's frame rate and the process will run indefinitely thus giving it the term of 'long term tracking'.

2. EXISTING SYSTEMS

Numerous approaches for object tracking have been proposed. These primarily differ from each other based on the way they approach the following questions: Which object representation is suitable for tracking? Which image features should be used? How should the motion, appearance, and shape of the object be modeled?

Almost all tracking algorithms assume that the object motion is smooth with no abrupt changes. It is generally assumed that the object's motion has a constant velocity or constant acceleration based on a priori information. Prior knowledge about the number and the size of objects, or the object appearance and shape, is also required. For example, the CamShift algorithm uses a color feature for real time object tracking. CamShift fails when the video is under rapid motion or when there are changes in the illumination and when there is background distraction. Adaptive Local Search and Kalman Filter systems exist which predict the position of the moving object. Improved CamShift reduces the effect of illumination interference and judges whether the target is lost. SURF algorithm is used with improved CamShift because it is invariant to scale, rotation and translation of the image. Thus, the lost target can be found out using improved CamShift and SURF Algorithm thus improving over the current system.

A popular facial detection and tracking system uses head pose estimation (HPE) and facial expression analysis. The new feature of integrating HPE with smile detection is introduced to support social communication skills learning. The system detects a human face using a boosting algorithm and a set of

Haar-like features, locates the eyes based on Haar-like features, detects the inner eye corners using their intensity probability distribution and edge information, finds the mouth corners using its intensity probability distribution, estimates the nostrils based on their intensity and geometric constraints and tracks the detected facial points using optical flow based tracking. The head pose is estimated from tracked points and a facial feature model using POSIT and RANSAC algorithms.

The system is able to detect tracking failure using constraints derived from a facial feature model and recover from it by searching for one or more features using the feature detection algorithms. Also, the system detects smiling mouth using a cascade of boosted tree classifiers with Haar-like features. Another system is aimed at the real-time tracking problems, an adaptive robust framework for object tracking. It uses the Kalman filter to predict the location of object, and then introduces a CamShift-based adaptive local search method for robust tracking, using the parameter resulted from the Kalman filter. Another system uses Face-TLD, a system for tracking of human faces based on Tracking-Learning-Detection (TLD). Face-TLD combines a priori information about the object class with the information from the video.

3. PROPOSED SYSTEM

The system that is proposed here can be explained by first mentioning the concepts used. We are using the OpenCV library and integrating the dll files in Microsoft Visual Studio to carry out template matching. We are also introducing PN discrimination algorithm to minimize tracking failures.

The tracker in the system estimates that the object is visible and its motion is limited. We can also enable the tracker to provide weakly labeled training data for a detector and thus improve it during runtime. The detector performs full scanning of the image to localize all the appearances that have been observed in the past. A detector can reinitialize a tracker to minimize the tracking failures. Detection based algorithms estimates the object location in every frame independently.

The detector does not drift and does not fail if the object disappears from the camera. It deals with arbitrarily complex video streams where the tracking failures are frequent. It never degrades the detector if the video does not contain relevant information, and operates in real time.

3.1 PN Discrimination

During the processing of every frame, the detector evaluation is carried out on the basis PN Discrimination. What this essentially implies is that the entire frame is divided into a series of grids and they are labeled either P type or N type depending on their properties. The grids which include the object which is to be tracked calls under P type while the rest of the grids are said to contain the background and they fall under N type.

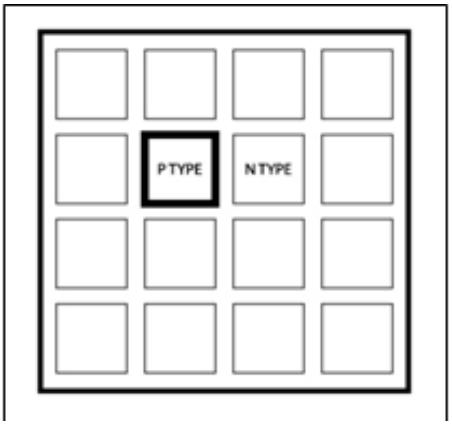

Fig. 1: PN Discrimination

The P type discrimination thus enables the detector to recognize missed detections and thus improves the accuracy of the system. The N type discrimination enables the detector to recognize false detections. A database is formed where these P type and N type images are stored and are constantly evaluated by the detector at the beginning of each frame.

3.2 Template Matching

The basis of detection is template matching. It is a technique for finding areas of an image that match to a template image. Two primary components, namely a source image (**I**) and a template image (**T**) are required. Essentially, the template image is matched line by line with the entire source image using process called as 'sliding'. By sliding, it is meant that the patch is moved one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents how good or bad the match at that location is (or how similar the patch is to that particular area of the source image). For each location of **T** over **I**, the metric is stored in the result matrix (**R**). Each location (X,Y) in **R** contains the match metric.

Fig. 2: Template Matching – Grayscale Result

The image above is the result R of sliding the patch with a metric TM_CCORR_NORMED. The brightest locations indicate the highest matches. As can be seen, the brightest location is the one with the highest value, so that location is considered the match. In practice, we use the function minMaxLoc to locate the highest value (or lower, depending of the type of matching method) in the R matrix.

In the above image, it would be worthwhile to note that the dog's face is white and this white color also matches with the whiteness of the wall's color and the small bowl located to the left lower corner of the image. However, the PN algorithm applied here has prevented false detection in this case. Now, different Template matching techniques are mentioned below:

1. CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

2. CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

3. CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

4. CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

5. CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$

6. CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

3.3 Algorithm

The algorithm is implemented using the OpenCV library. The interface is developed using C# in Microsoft Visual Studio.

Fig. 3: Final User Interface

The above figure shows the final user interface that has been developed. The object to be tracked can be seen within the yellow bounding box whereas the red bounding box denotes the Region of Interest (ROI). The ROI can be said to be a secondary Source Image (I) which is especially demarcated so as to carry out the scanning during template matching. This is because the object has a greater probability of being within the ROI in the next frame which comes up only after 30 milliseconds.

The matching values can be seen within the user interface. The user can also edit the trigger values i.e. the values which determine whether a match is to be registered or not by the detector. It is obvious that the matching is to take place for the P type images as well as for the N type images. Thus trigger values for both has to be entered by the user. The P and N databases, of course, cannot be without bounds as that would give rise to longer processing times which is unacceptable for a real-time process. Thus, the databases are also constantly updated and images which are not required are removed to make way for relevant images.

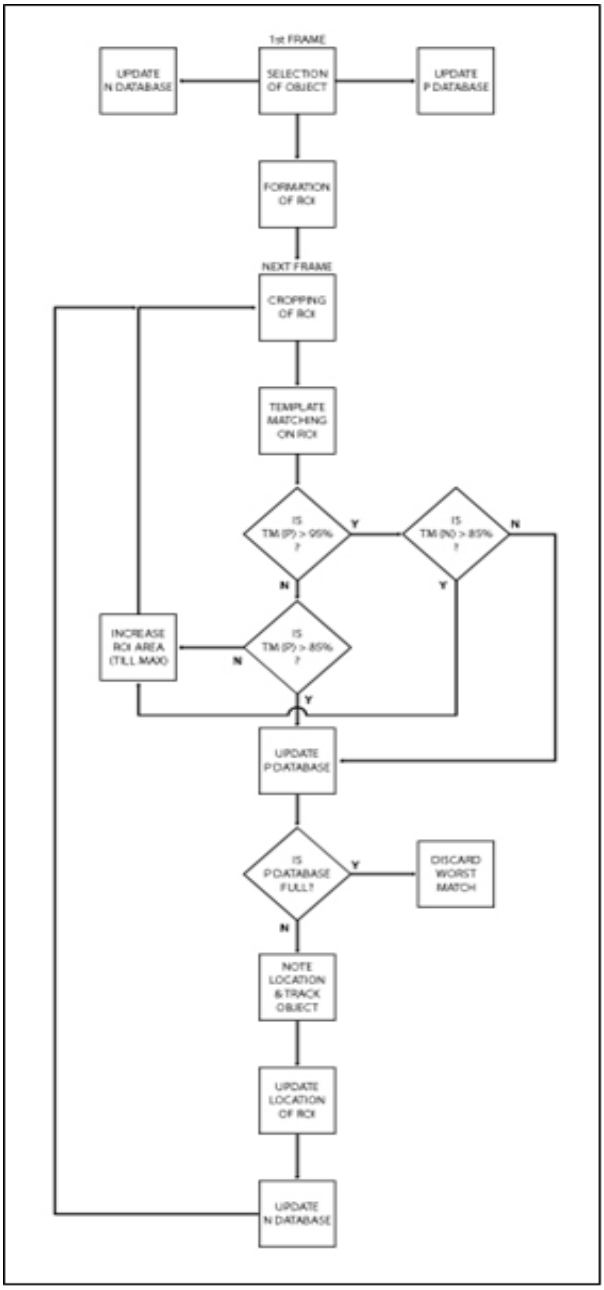

Fig. 4: Algorithm Flow Chart

The following is the proposed algorithm:
1. Initialize all system variables
2. Initialize the camera
3. Fetch the first frame from the camera
4. User selects the object from the first frame
5. P and N databases are updated
6. Create an ROI around the object and crop the ROI
7. Apply template matching in the ROI and obtain the location with the highest intensity as the location of the object
8. If template matching for the P type images is greater than the trigger then check if template matching for N type images is greater than the trigger. If yes then increase ROI and go to step 6 else go to step 10
9. If template matching for P type images is lesser than the trigger then check if it is more than the second trigger. If no then increase ROI and go to step 6 else go to step 10
10. Update P database and discard the image with worst match
11. Track the object and update the positioning of the ROI
12. Update N type database and move to the next frame

4. CONCLUSIONS

The latest breakthroughs suggest that automatic real-time tracking, learning and detection will only improve with time and be even more efficient. An attempt has been made here to eliminate the issues of false detections and loss of detection through the use of PN algorithm. This helped the tracking of the object even during complexities caused by the motion of the object, environmental changes and full or partial object occlusions.

REFERENCES

[1] Zdenek Kalal, Jiri Matas, "Tracking- learning – Detection" IEEE Transactions on pattern analysis and machine intelligence, vol. 34, no. 7, july 2012 1409, 0162- 8828/12 2012 IEEE

[2] Zulfiqar Hasan Khan, "Robust Visual Object Tracking using Multi-mode Anisotropic Mean Shift and Particle Filters" IEEE transactions on circuits and systems for video technology, Vol. 21, No. 1, Jan 2011

[3] Benussi Toni, Juric Darko, "A Robust Hand Detection and Tracking Algorithm with Application to Natural User Interface" May 2012

[4] Xin Chen, Xiang Li, Hefeng Wu, Tashiung Qiu, "Real Time Object Tracking via Cam Shift based Robut Framework" IEEE International Conference on Information Science and Technology, Mar 2012

[5] Zdenek Kalal, Krystian Mikolajczyk, Jiri Matas, "Face TLD – Tracking Learning Detection Applied to Faces" IEEE International Conference on Image Processing , Sep 2010

[6] Zulfiqar Hasan Khan, "Facial Feature Detection and Tracking in a new Multi-modal Technology-Enhanced Learning Environment for Social Communication" IEEE International Conference on Signal and Image Processing Applications, 2009

[7] Open CV documentation