COMPUTER VISION: IMAGE PROCESSING LAB

VISHAL RAMADOSS AND YESHASVI TVS

Our aim was to track a given object in a video using OpenCV Feature detection and Matching tools . The object to be tracked is the white bulb which is the first frame of video provided . Also we had to indicate the tracking by depicting a rectangle which encloses the bulb on each frame of the video and the rectangle gives the orientation angle of the bulb wrt orientation in first frame .

Figure 1. Object to be tracked

The SURF Algorithm is used and two methods were tested .

1.By matching the object in current frame to object in the first frame of the video

2. By matching the object in the current frame to object in the previous frame of video.

SURF : Speeded Up Robust Features is a performant scale and rotation invariant interest point detector and descriptor.

In a given image, SURF tries to find the interest points – the points where the variance is maximum and then constructs a 64 variable vector around it to extract the features .In this way there will be as many descriptors in an image as there are interest points.

Major stages in SURF[1] :

1. Find keypoints in scale space and precisely locate them in space and scale
    - Search over all scales and locations
    - Uses Laplacian of given image point using Gaussian filter at different scales. Looking at the evolution of filter response for different scale factors , we obtain a curve which reaches a maximum value at some sigma(size of kernel) value .
2. Assign orientation to key points
    - It uses wavelet responses in horizontal and vertical direction for a neighborhood of 6s. The dominant direction of gradient is estimated by calculating the sum of all responses within a sliding orientation window of 60 degrees. The computation is

faster because the wavelet response can be found out using integral images very easily at any scale. This step is basis for rotation invariance .

3. Associate a rotation invariant set of features to the key point at the selected scale.
    - SURF provides a functionality called Upright-SURF or U-SURF which improves speed and robustness upto +-15 deg . The upright Boolean parameter if it is 0, the orientation is calculated and if it is 1 , the orientation is not calculated and it is more faster.
4. Feature detection
    - To detect the features , the Hessian matrix is computed at each pixel that measures the local curvature of a function. The idea is therefore to define corners as imagepoints with high local curvature (high variation in more than one direction). This matrix can be computed using Laplacian Gaussian kernels.
    - The scale invariant features should be detected as local maxima in both the image space and the scale space (as obtained from derivative filters applied at different scales.)
    - The use of sign of Laplacian (trace of Hessian Matrix) for underlying interest point. The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse situation. In the matching stage, we only compare features if they have the same type of contrast (as shown in image below).

SURF adds lot of features to improve speed and it is faster than SIFT. It is good at handling images with blurring and rotation .

Methodology

Case 1 : Object matching in current frame to object in first frame

The steps are given below :

- A set of declarations are made next :
    1. Matrix variables for frame , descriptors , keypoints , matches and homography matrix
    2. Object creation for Surf feature detector , Surf descriptor extractor , Flann based matcher
    3. Vector of keypoints ,matches , object and scene corners.
- Open the video file using Videocapture class and declare all windows for displaying the output
- Get the first frame of video ***frame_bgr*** and turn it into grey scale ***frame_grey*** and clone it in ***frame_grey_1*** for processing.
- Detect the keypoints in the image using detect method. The detector.detect gives us the points of interest in both images.

- Draw keypoints detected by SURF detector using draw keypoints function. The parameters are ***frame_grey_1***(source image),***keypoints_1*** (Keypoints from source image),***img_keypoints_1*** (OutputImage), ***Scalar::all(-1)*** (color of Keypoints),flags.
- Caculate the descriptors using compute method. Here we describe those features using extractor.compute and it gives us the way to compare points of interest and comparing descriptors tells us how similar these points are. The parameters are ***frame_grey_1***(image) , ***keypoints_1*** (input collect of keypoints) and ***descriptors_1*** (computed descriptors.)
- Inside the loop,Read the current frame(***frame_bgr***) of the video and turn into grey scale(***frame_grey***) in order to process it. Detect and draw the keypoints andcalculate the descriptors .
- The matcher chosen here is FLANN matcher . The parameters are ***descriptors_1***(query set of descriptors) ,***descriptors***(Train set of descriptors) and ***matches***(matches i.e if a query descriptor is masked out in a mask , no match is added for this descriptor . So, matches size may be smaller than the query descriptors count.)
- Quick calculation of max and min distances between keypoints are made and good matches are found for distances less than 3*min_dist,or a small arbitrary value (0.02) in the event that min_dist is very small .
- The match method calculates actual matches and to visualize the calculated matches drawMatches are used. This function draws matches of keypoints from two images in the output image. Match is a line connecting two keypoints(circles). The parameters are ***frame_grey_1***(first source image), ***keypoints_1***(Keypoints from first source image) , ***frame_grey***(Second source image) , ***keypoints***(Keypoints from second source image) ,***good_matches***(matches from first image to second one) , ***img_matches***(output image) , match color , single point color , matches mask and flags.
- Localize the object and calculate the transformation between the images using the findHomography function using RANSAC that tries many different random subsets of the corresponding point pairs (of four pairs each), estimate the homography matrix using this subset and a simple least-square algorithm, and then compute the quality/goodness of the computed homography which is the number of inliers for RANSAC. The best subset is then used to produce the initial estimate of the homography matrix and mask of inliers/outliers.
- Define the object corners with the given limits
- Calculate the images of the corners of rectangle by transformation using perspectiveTransfrom().
- Draw the lines between corners using line function .
- Show the images (first and current) in a new window named good matches.
- The time of the program is measured in another main program at the bottom of the code which utilizes Boost:time libraries that is initialized at the beginning globally.

It is observed the time taken for this method is ***Time in MilliSeconds 115432 .*** The tracking of object is shown below :

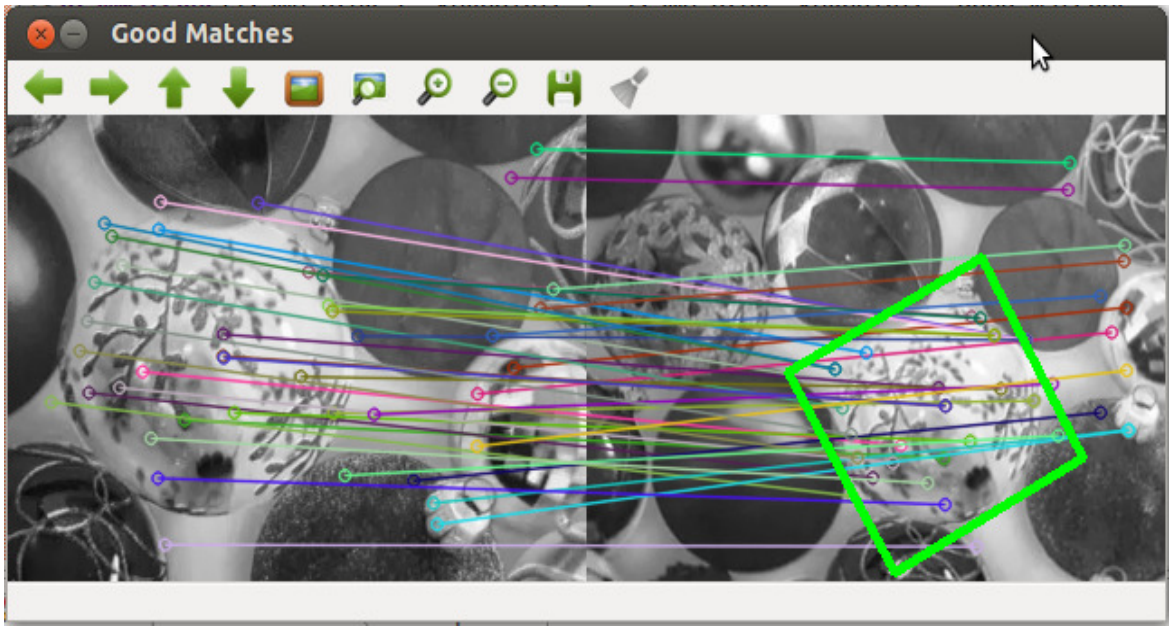

Fig 2. Comparing the current frame with first frame

Case 2 : Object matching in current frame to object in previous frame

Here we are going to consider two cases without geographical filtering of keypoints and with presence of filtering based on geographical location of keypoints.

Since here we compare each frame to the next in the video rather than to the first frame , the motion of the object from one frame to next can be assumed to be a small translation . As a consequence , when declaring the detector of class SurfFeatureDetector , it is possible to use U-SURF. i.e UPRIGHT =True . In this case , the rotational invariance is not necessary and this speeds up the calculations .

```
SurfFeatureDetector detector(minHessian, 4, 2, 0, true);
```

Parameters:

hessianThreshold – Threshold for hessian keypoint detector used in SURF.
nOctaves – Number of pyramid octaves the keypoint detector will use.
nOctaveLayers – Number of octave layers within each octave.
extended – Extended descriptor flag (true - use extended 128-element descriptors; false - use 64-element descriptors).
upright – Up-right or rotated features flag (true - do not compute orientation of features; false - compute orientation).

The code is similar to the previous one except we need to update the previous frame with current one as we need to compare current frame with previous frame . i.e.

- We open and read the video file***(frame_bgr)*** and declare the necessary mat variables , SURF objects and vectors . The read frame is converted to grayscale and cloned for processing ***(frame_grey_p).*** The detect and compute methods are used to perform keypoint detection and descriptor calculation and the image is displayed in ***img_keypoints*** where the detected keypoints are drawn***.***
- Similarly , inside the loop the new frame is read from the video (***frame_bgr***) and converted to grayscale (***frame_grey_c***) . We detect keypoints for current image and compute the descriptors and use FLANN matcher to get matches . Draw matches are used to visualize the matches, homography and perspective transform is calculated and lines are drawn between corners.
- To do a comparison between previous and current frame , we need to update the previous frame and respective parameters also needs to updated .

So we add :

```
frame_grey_p = frame_grey_c.clone();    //Update the previous frame as current frame
obj_corners = scene_corners;            //Update object corners
keypoints_p = keypoints_c;              //Update previous keypoints
descriptors_p = descriptors_c.clone(); //Update previous descriptors
p++;                                    //Increment counter
```

At the end of while loop after the processing is done , so each time the previous frame gets updated with the current finished one and similarly object corners ,keypoints and descriptors .

It is observed the time taken for this method(without geographical filtering) is ***Time in MilliSeconds 107055 .*** The tracking of object is shown below :

Notations used : _p = previous frame _c= current frame

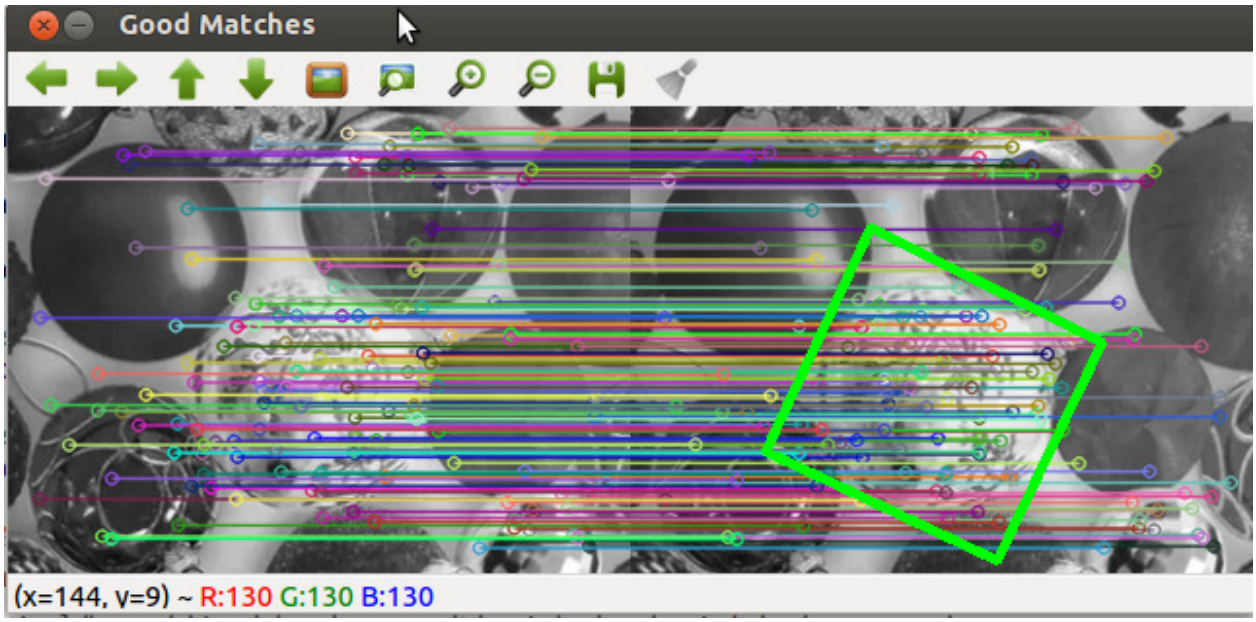

Fig 3. Comparing the current frame with previous frame without filtering

With geographical filtering of keypoints :

Our aim here is to filter out other keypoints and retain the keypoints that is pertaining to the object alone . The basic idea here is to calculate the center and radius of the white bulb and filter based on the geographical location i.e if the distance between the center and keypoints is less than or equal to radius then we push back those keypoints into a vector which is of our interest . This in turn filters out other keypoints that are outside the white bulb . This is achieved by following part of code :

```
center=(obj_corners[0]+obj_corners[1]+obj_corners[2]+obj_corners[3])*0.25;//since the
bounding box is given , we define center as mean of given 4 corners
r=round(norm(center-obj_corners[0]));// distance between center and one corner
```

The filtering is achieved by following part of code :

```
// Geographical filtering of keypoints detected
 for( unsigned i=0 ; i<keypoints_p.size() ; i++ )
 {
 if(norm(center-keypoints_p[i].pt)<=r)
 keypoints_p_f.push_back(keypoints_p[i]);
 }
```

We update the previous image frame ,keypoints and descriptors with the current values and we update the object corners with scene corners . Also we recalculate the center and radius of the circle for geographical filtering after updating the value of object corners with scene corners for next iteration.

Notations used : _p = previous frame _c= current frame _p_f = previous frame with filtering
_c_f= current frame with filtering

It is observed the time taken for this method(with geographical filtering) is ***Time in MilliSeconds 82588 .*** The tracking of object is shown below :

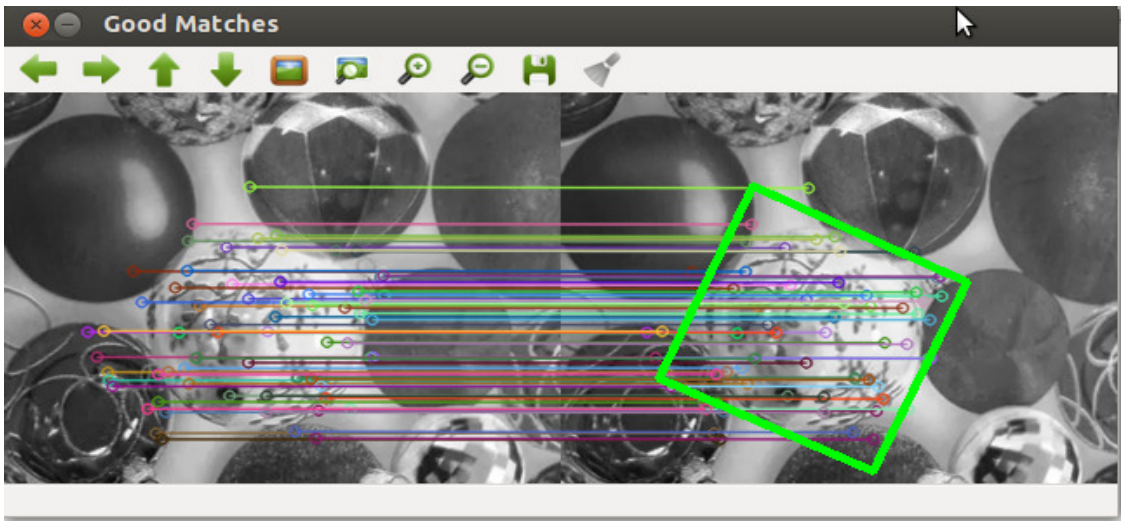

Fig 4. Comparing the current frame with previous frame with filtering of keypoints based on geographical location

Analysis and Conclusion

With the experimental results , we conclude that comparing current frame with previous frame with filtering(Method 2 with filtering) is the efficient method for solving object tracking as the computation time is least also , we are considering only those key points which inside ROI (Region of interest). In Fig 3. We changed the upright parameter to be true as the motion of the object from one frame to next can be assumed to be a small translation and In this case , the rotational invariance is not necessary and this speeds up the calculations .

References

[1] “Introduction to SURF”

http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#