SIFT based object tracking

Vijay Harid
Stanford University
vharid@stanford.edu

Abstract – This report provides an algorithm to track an object in a video that has been selected by the user in the first frame. The user may select a particular region/object with a rectangular box in the first frame of the video and the algorithm will track the object through the rest of the video sequence. The algorithm uses a frame-to-frame SIFT based approach with an affine assumption. The algorithm also allows for tracking multiple objects at once. The tracker is not a real-time based algorithm but instead uses the entire video as the input and outputs the same video with boxes around the desired regions.

Introduction

Object tracking is an image processing application with a wide number of applications. Applications may include tracking particular people in a video for security reasons to tracking planetary objects from satellite data for astronomically studies.

Object tracking fundamentally entails estimating the location of a particular region in successive frames in a video sequence. Properly detecting objects can be a particularly challenging task especially since objects can have rather complicated structures and may change in shape, size, location and orientation over subsequent video frames. Various algorithms and schemes have been developed over the past few decades to track objects in a video sequence, and each has their own advantages and drawbacks. Any object tracking algorithm will contain errors which will eventually cause a drift from the object of interest. The better algorithms should be able to minimize this drift such that the tracker is accurate over the time frame of the application.

Prior and Related Work

One of the original and widely known object tracking algorithms is the Kanade-Lucas-Tomasi (KLT) tracker. The KLT algorithm uses an optical flow method to determine the time-evolution of an image over subsequent video frames. The algorithm generally relies on a Harris corner detector to determine keypoints and then tracks the evolution of these keypoints over the rest of the video frame. The KLT tracker can be quite fast when compared to other image trackers; however, can become inaccurate for fast motion or deformations of the original selected object.

Optical flow methods are used quite often for tracking applications. However, optical flow is not the only technique that is utilized for object tracking.

The following table displays the four main categories that are used in image tracking [1]:

Categories	Representative Work
Point detectors	Moravec's detector [Moravec 1979], Harris detector [Harris and Stephens 1988], Scale Invariant Feature Transform [Lowe 2004]. Affine Invariant Point Detector [Mikolajczyk and Schmid 2002].
Segmentation	Mean-shift [Comaniciu and Meer 1999], Graph-cut [Shi and Malik 2000], Active contours [Caselles et al. 1995].
Background Modeling	Mixture of Gaussians[Stauffer and Grimson 2000], Eigenbackground[Oliver et al. 2000], Wall flower [Toyama et al. 1999], Dynamic texture background [Monnet et al. 2003].
Supervised Classifiers	Support Vector Machines [Papageorgiou et al. 1998], Neural Networks [Rowley et al. 1998], Adaptive Boosting [Viola et al. 2003].

As mentioned in the table above, the Scale Invariant Feature Transform (SIFT) is a noteworthy detector often used in image matching applications. The algorithm essentially uses a gradient based descriptor for particular points and matches these descriptors to corresponding points in another image. The matched points can then be used to determine the location of a new object. Such a SIFT based detector is employed in the algorithm that is discussed in this report.

Algorithm description

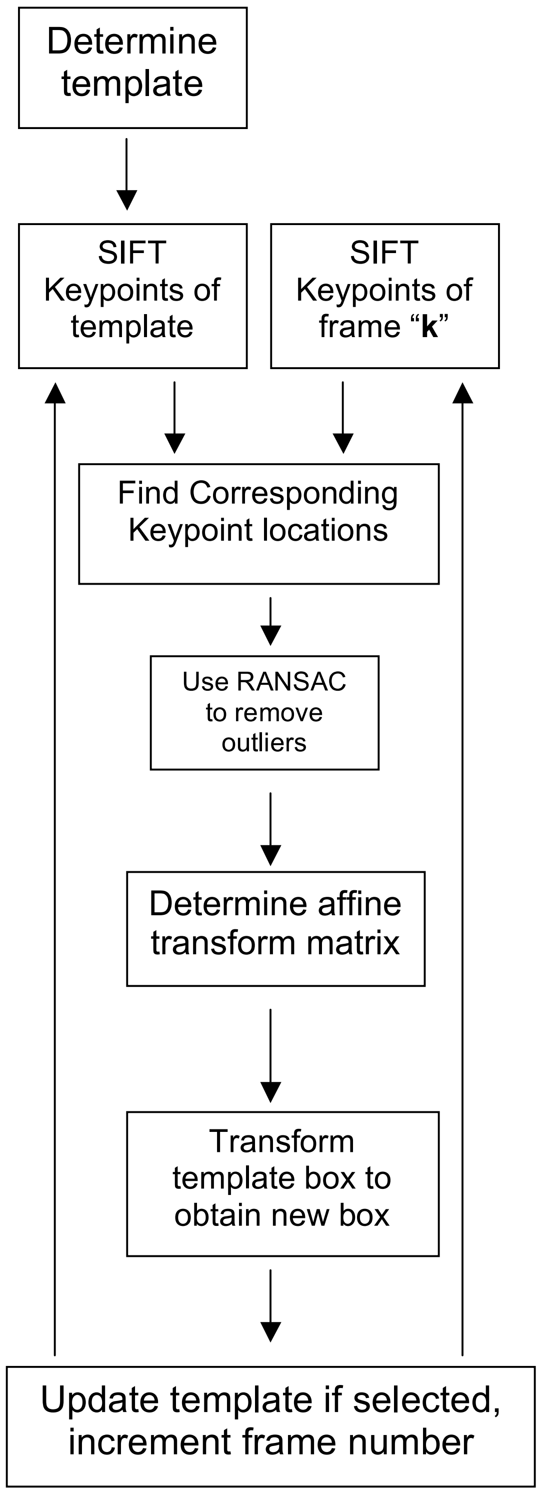
The algorithm first relies on a SIFT descriptor to determine keypoints in the image. The algorithm uses the openly available SIFT algorithm on the University of British Columbia (UBC) Computer Science webpage, courtesy of Professor David Lowe [2].

David Lowe's algorithm identifies keypoints in a given image using a SIFT descriptor and outputs the locations of these keypoints. A quick overview of the algorithm is described below, further detail will be provided after.

The tracker uses this SIFT algorithm to compute the keypoints in the selected image (template) and also computes the keypoints in the next frame of the video sequence. The keypoints in the template is then compared to the keypoints in the next frame to determine the matched corresponding keypoints. If there are N keypoints in the template, there will be K matches in the next frame with N > K always being true. At this point, a 2D affine transformation between the two images is assumed. This is approximately true if the 3D changes between subsequent frames are not too drastic. This would be good enough if the corresponding keypoints are always matched correctly. Since there are almost always outliers in the matching process, a RANSAC algorithm has been developed to remove any outliers that corrupt the affine transformation. Once the RANSAC algorithm is run, it outputs the locations of the best points for the affine transform. With this information, an affine transform matrix can be computed and applied on the original box to determine the new box that surrounds the location of the object in the new frame. Since the affine transform results in a box that is not necessarily rectangular (instead it is a shifted parallelogram), the minimum and maximum x and y values of the parallelogram are determined to make a box that surrounds the object. This leads to further complications that will be discussed later.

In addition, there two options provided to the user before the tracking begins. The first option compares subsequent frames to the original template selected in the first frame and does not update the template. The second option updates the template to the location found in the next frame. This allows for dramatic changes in the selected object over the course of the video. Updating the template, however, results in the box location invariably increasing in size which must be compensated for.

A high level block diagram of the overall procedure is shown below:

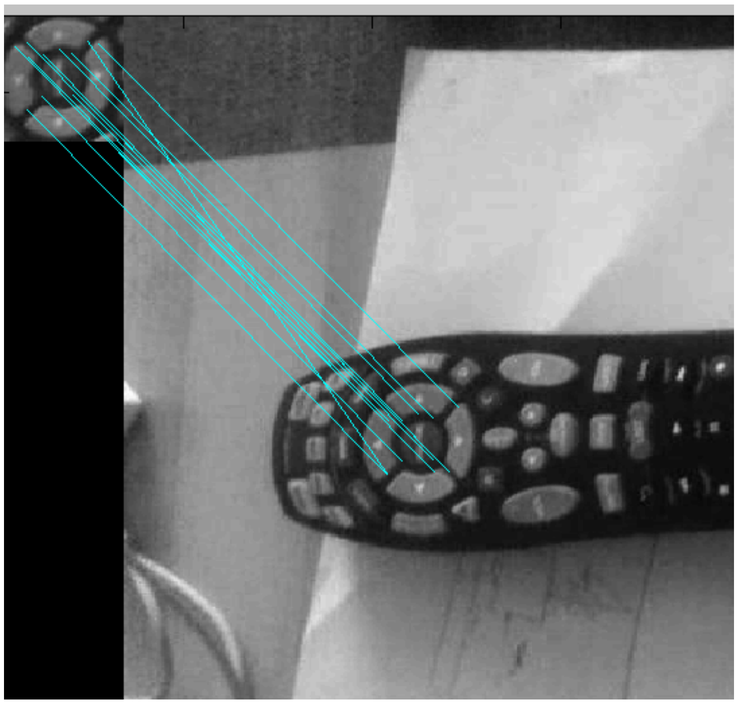

Determine template
SIFT Keypoints of template
SIFT Keypoints of frame "**k**"
Find Corresponding Keypoint locations
Use RANSAC to remove outliers
Determine affine transform matrix
Transform template box to obtain new box
Update template if selected, increment frame number

Object selection:

The tracker first requires the user to select one or more images in a video segment, an example selection is shown below:

As shown in the image above, the selected region is given by a rectangular box.

David Lowe's algorithm determines keypoints in an image using a SIFT descriptor and outputs the locations of the keypoints in a vector. The computation time is almost completely determined by the number of keypoints in the image. The image below shows the corresponding keypoints between a selected template and a subsequent frame without RANSAC:



The above correspondence lines are between the template keypoints and the keypoints in a later frame in the video sequence before RANSAC. Upon close inspection it is evident that not all the correspondence lines are matched correctly and will improperly skew the affine transformation matrix. In order minimize the number of mismatches, an additional approximation is made. It is assumed that the object location does not move more than 50% of its own dimension in any direction over one frame. This approximation removes any mismatches that lie far away from the object and also drastically reduces the computation time.

Affine transform

An affine transformation is assumed between subsequent frames. Mathematically, the affine transformation for N points can be written in the following way:

$$\begin{aligned}
X1 &= a1*x1 + a2*y1 + a3\\
Y1 &= b1*x1 + b2*y1 + b3\\
X2 &= a1*x2 + a2*y2 + a3\\
Y2 &= b1*x2 + b2*y2 + b3\\
X3 &= a1*x3 + a2*y3 + a3\\
Y3 &= b1*x3 + b2*y3 + b3\\
&.\quad .\quad .\quad .\\
&.\quad .\quad .\quad .\\
&.\quad .\quad .\quad .\\
XN &= a1*xN + a2*yN + a3\\
YN &= b1*xN + b2*yN + b3
\end{aligned}$$

Equivalently, in matrix notation:

$$\begin{pmatrix} x1 & y1 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & y1 & x1\\ x2 & y2 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & y2 & x2\\ x3 & y3 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & y3 & x3\\ . & . & . & . & . & .\\ . & . & . & . & . & .\\ . & . & . & . & . & .\\ . & . & . & . & . & .\\ xN & yN & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & yN & xN \end{pmatrix} \begin{pmatrix} a1\\ a2\\ a3\\ b1\\ b2\\ b3 \end{pmatrix} = \begin{pmatrix} X1\\ Y1\\ X2\\ Y2\\ X3\\ Y3\\ .\\ .\\ .\\ .\\ XN\\ YN \end{pmatrix}$$

This matrix equation is of the form Av = b, where the vector 'v' contains the parameters for the affine transform. Here, the uppercase letters correspond to the new points while the lowercase letters correspond to the old coordinate points. Since the matrix A will not be invertible in most cases, a least squares solution for v must be determined. The least squares solution is given by:

$$V_{ls} = (A^T A)^{-1} A^T b$$

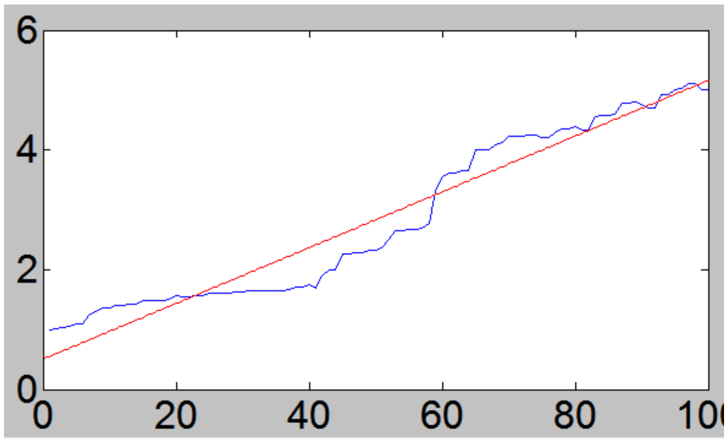
RANSAC:

The affine assumption works quite well for most cases; however, just a few a keypoint mismatches can result in a drastic deformation of the affine transform matrix. For this reason, a RANSAC algorithm is employed to remove such outliers and keep the points that are most likely correct matches.

The RANSAC method works in the following way. The algorithm first picks three random points from the keypoints that have been determined by the SIFT algorithm. Since neighboring points are more likely to be similar, the three points are picked to be at least 10 pixels apart. Only three points are picked at first since this is the minimum number of points required to uniquely determine all the parameters of an affine transform matrix. An affine transform matrix is then computed with these three points and is applied to the rest of the keypoints in the template. As each keypoint is transformed by the affine matrix, it results in a new point that is compared to the original corresponding location of the template keypoint. If the new point is within 2 pixels of the corresponding location of the template keypoint, the template keypoint is added to a "consensus set". Once a consensus set is obtained, a new affine transform matrix is determined with the consensus set points. This affine transform matrix is then applied to the consensus set points and the Euclidian distance from the calculated points to the actual corresponding keypoint locations are calculated. From this, an average Euclidian distance can be determined which is called the "error". The entire process is then repeated 300 times and the consensus set with the least error is chosen as the best consensus set. A final affine transformation matrix is then determined from the best consensus set.

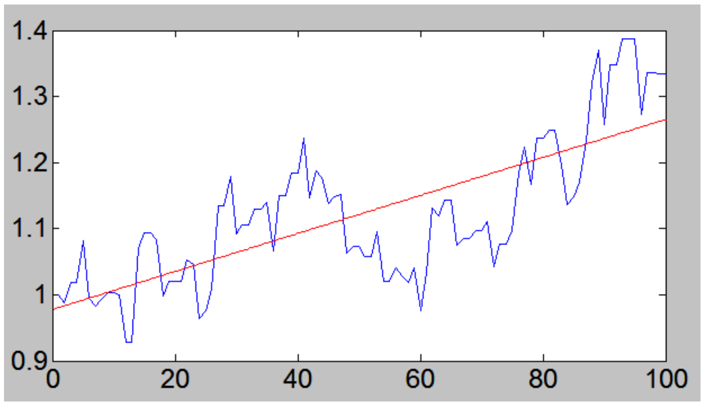
Once the RANSAC algorithm computes the most appropriate affine transform matrix, this matrix is applied to the original box that was used to select the object of interest. The result is a new box (parallelogram) that surrounds the new location of the object of interest. Since the new box is generally a parallelogram, a rectangular box is determined by finding the minimum x and y coordinates of the parallelogram and creates a rectangle of size (xmax-xmin) x (ymax-ymin) around the parallelogram. This procedure tends to be an issue since the rectangular box is always bigger than the original parallelogram which will lead to the area of the rectangular box increasing in size as the object evolves over the course of the video sequence.

The following plot shows the area of the box of a purely translational image normalized to the area of the original box.



The above plot is derived from the first 100 frames a video sequence. The red line in the plot is a linear fit to the data. It is apparent that the area of the box grows approximately linearly with each subsequent frame. The area in the above plot increases at a rate of approximately 3% per frame. This corresponds to approximately 1.5% increase in each dimension per frame.

In order to compensate for this "enlargement drift", each dimension of the box is reduced by 5% every six frames. This reduces the amount of enlargement significantly; however, the problem is not completely solved. Properly correcting for this drift would require more complicated methods. The figure below shows the normalized box area after enlargement compensation:



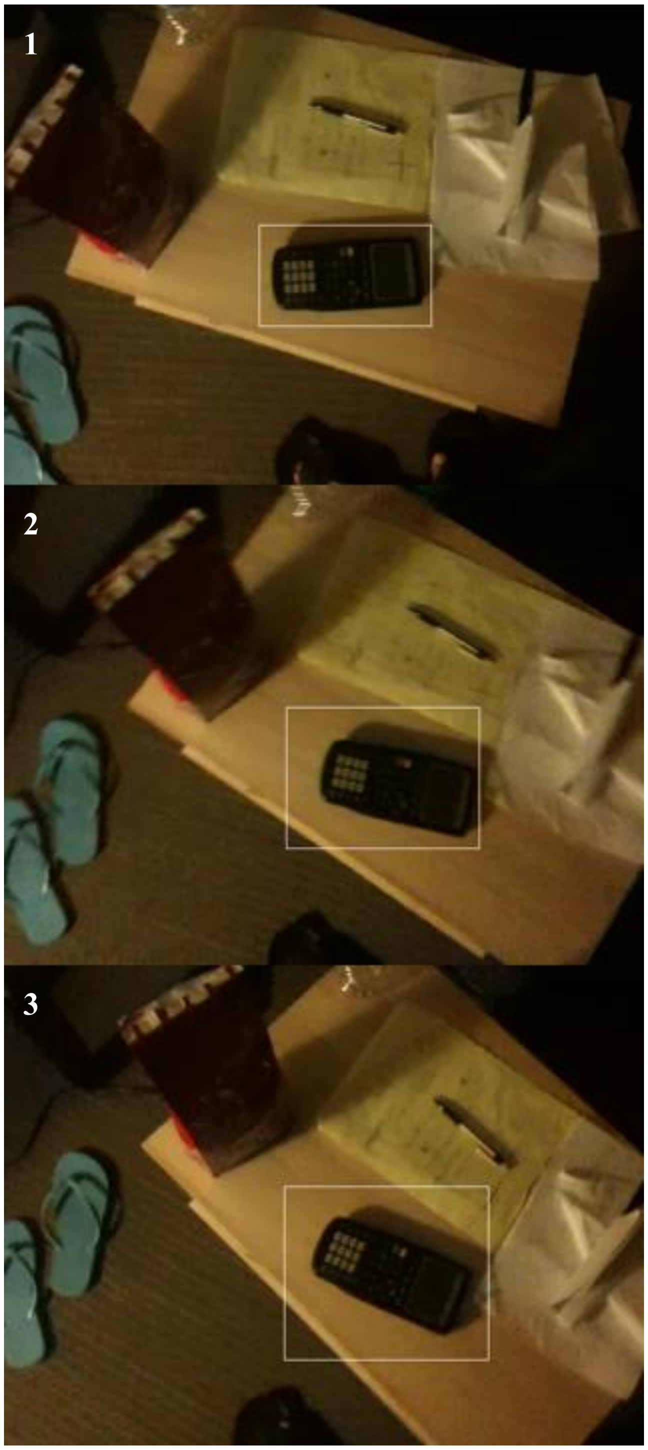
It is apparent by comparing the two box area plots that enlargement compensation does decrease the box area on average.

Updating the template causes this enlargement drift that tends to accumulate as the video evolves. For this reason, the user is given the option of not updating the template (small change approximation) or updating the template. Not updating the template tends to provide more accurate results especially when 3D changes are minor.

Experimental results

Since the tracker presented in this report is a SIFT based tracker, the computation time tends to be rather slow. The algorithm takes approximately .5 to 2 seconds per frame depending on the size of the selected region. The following figure includes four random consecutive images in a 180 frame video after a region is tracked. The tracking mode for this video sequence does not utilize template updating and compares successive frames only to the original selected template.

It is apparent in the image below that the tracker successively tracks the select region throughout the video sequence. However, the example figure shown is from a video sequence that does not include drastic changes to the original template. Extremely quick motion or large deformations to the template can result in errors in the object tracking algorithm.



Conclusion

This proposed project involved tracking an object after it is selected on the first frame of a video sequence. The method used in this project involved using a SIFT descriptor combined with RANSAC and an affine transformation assumption. The results tend to be rather accurate for motions and deformations that are not too drastic. Updating the template when tracking leads to an “enlargement drift” that needs to be compensated for but still leads to errors in tracking. Thus, comparing subsequent frames to the original template alone tends to provide the most accurate tracking results.

References

[1] Y.Alper, J.Omar, and S.Mubarak. “*Object Tracking: A Survey*” ACM Computing Surveys, vol. 38, no. 4, Article 13, December 2006.

[2] L.David. "*Demo Software: SIFT Keypoint Detector*," [online] 2005, http://www.cs.ubc.ca/~lowe/keypoints/ (Accessed: 20 May 2010).