Stable 2D Feature Tracking for Long Video Sequences

Jong-Seung Park, Jong-Hyun Yoon, and Chungkyue Kim

Department of Computer Science & Engineering, University of Incheon
177 Dohwa-dong, Nam-gu, Incheon, 402-749, Republic of Korea
{jong,jhyoon,ckkim}@incheon.ac.kr

Abstract. In this paper, we propose a 2D feature tracking method that is stable to long video sequences. To improve the stability of long tracking, we use trajectory information about 2D features. We predict the expected feature states and compute a rough estimate of the feature location on the current image frame using the history of previous feature states up to the current frame. A search window is positioned at the estimated location and similarity measures are computed within the search window. Once the feature position is determined from the similarity measures, the current feature states are appended to the history buffer. The outlier rejection stage is also introduced to reduce false matches. Experimental results from real video sequences showed that the proposed method stably tracks point features for long frame sequences.

1 Introduction

Feature tracking is one of the most important issues in the fields of image analysis and computer vision. During the last several decades, video tracking technology has been advanced for the practical uses on the surveillance, analysis and representation of video footage[1]. When an image sequence is given as an input, two sets of feature points are extracted from each pair of two adjacent frames. Then a matching algorithm evaluates feature correspondences between the two sets of feature points. Feature tracking is an important field in various computer vision parts since it provides the fundamental data for the further analysis. They can also be used to estimate the object shape toward the object reconstruction. However, the development of related applications has suffered from the instability of tracking methods. Feature tracking is unstable in nature and false correspondences can be occurred at any unexpected time.

In this paper, we propose a robust feature tracking method for long video sequences. Extraction of good features is also important for the robust feature tracking. Feature points are extracted in stable locations that are invariant with respect to image translation, rotation and scaling[2]. After generating a Gaussian image pyramid, we compute difference of Gaussians and the extrema of the difference of Gaussians are selected as feature locations. If two sets of feature points are extracted, we compute the predicted states of features using the history of the feature points from the first frame up to the immediate previous frame. Then

we predict each feature position in the current frame. The predicted position is used as an initial value for searching a more accurate feature position. When we refine the feature location from the predicted location, we estimate the accurate feature position by minimizing the matching error.

In our method, we allow the camera to move freely in the physical space. However, if the target object moves together with the camera, it is difficult to solve the problem due to the occurrence of tracking occlusion and the motion ambiguity even in the case of a slight camera movement[3]. The moving objects must be detected in advance and should be handled separately. To avoid such an ill-posed problem, we assume that all objects in the physical space are static relative to the camera.

2 Improving the Accuracy of Feature Tracking

For the stable and robust feature tracking, each step of the tracking should be processed accurately. The three main steps are the extraction of stable feature points, the prediction of feature states, and the computation of feature correspondences. In this section, we describe each step in detail explaining how to improve the accuracy of feature tracking.

2.1 Feature extraction

The reliable feature extraction is the first fundamental step for the robust feature tracking. Feature points should be unique and distinguishable from other feature points for stable feature matching. Typically, feature points are chosen from the intensity corners on the image. Extraction of good features directly drives the improvement toward the robust feature tracking. Good features could be tracked well regardless of translation, rotation and scale changes in input images.

Our feature extraction is similar to the method of scale-space extrema[4][5]. In the first stage, we determine the locations of feature points that are computed by the difference of Gaussian function[5]. The scale space is computed using the input image $I(x, y)$ and the scale Gaussian $G(x, y, \sigma)$. The scale space is defined by Eq. (1), which is the convolution of G with I:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \ . \tag{1}$$

In the scale space, the difference of Gaussian (DOG) is used for the computation of keypoint locations[5]. The DOG filter corresponds to $G(x, y, k\sigma) - G(x, y, \sigma)$. The difference of two Gaussian images separated by a factor k are represented by:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \ .$$

The DOG function estimates an approximate value using the scale-normalized Laplacian of Gaussian. If the computed difference is the maximum or the minimum, the location is determined as a feature point location. Such feature points are more stable than feature points from other corner-based feature extraction methods.

2.2 State modeling for feature tracking

We assume that the target objects are static and the camera is allowed to move freely. Let $\mathbf{x}_n = (x_n, y_n)$ be a feature point in the n'th image. Then the movement from $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$ is described by:

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n, \mathbf{p}_n) \ . \tag{2}$$

In Eq. (2), $\mathbf{p}_n = (p_{n,1}, \ldots, p_{n,m})$ is the vector of state variables that describes the state transition from the previous frame to the current frame. Hence the problem is to estimate values of the state vector $\mathbf{p}_n$. A simple form of the state vector is $\mathbf{p}_n = (p_{n,1}, p_{n,2})$ where $(p_{n,1}, p_{n,2})$ is the translation vector from $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$:

$$x_{n+1} = x_n + p_{n,1}, \ y_{n+1} = y_n + p_{n,2}.$$

Let $\mathbf{p}_n$ be the state vector at the n'th frame. Each state vector is saved to the state history buffer. We use the state vectors saved in the state history buffer to predict the feature point locations. The history buffer contains (n-1) state vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{n-1}$ when processing the n'th frame. To estimate the feature location at the current frame, we utilize the history of state changes from the first frame to the (n-1)'th frame. Using the state history up to the (n-1)'th frame, we initialize the location of the search window to find the correspondence in the n'th frame.

Since the frame rate should be high enough to accommodate to a real-time video stream, we restrict the search area within a small nearby region. We initialize the search window location using the moving velocity of the feature up to the current frame and predict the new feature location using the previous feature position and the current state vector. We approximate the velocity of the state vector by:

$$\mathbf{v}_{n-1} = \mathbf{p}_{n-1} - \mathbf{p}_{n-2} \ .$$

Using the velocity, the state vector at the n'th frame is predicted by:

$$\mathbf{p}_n = \mathbf{p}_{n-1} + \mathbf{v}_{n-1} \ . \tag{3}$$

Then the location of the feature point is predicted by $\mathbf{p}_n$.

Once the predicted location is determined, we put a search window on the location and compute the similarity measures by moving the window around the predicted location. The location of the lowest dissimilarity measure is chosen as the corresponding location. The size of the search window is adaptively determined according to the current residual error.

2.3 Rejection of outliers

We use the RANSAC algorithm[6] to reject outliers among tracked feature points. The RANSAC is a robust algorithm for recognizing wrong tracked features[7]. The basic idea is as following. We make a data model consisting of the minimum number of data from the collected data set. The randomly selected

data set is used to estimate the model coefficients which are computed using a least squares optimization method. After making the model, we apply the model to the unselected data and the evaluated value is compared with the observation. Then we compute the minimum number of data where the difference from the observation is smaller than a predefined threshold. The threshold value is used to determine whether a data is an inlier or an outlier[8]. An inlier is the data whose difference between the computed value and the observed value is smaller than the threshold. This computation is iterated until reaching the predefined maximum number of repeat. We select the final model to have the maximum number of inliers. The feature data is the set of the inlier points in the final model.

2.4 Tracking feature points

In the following, the overall steps of our feature tracking method are described. In the first step, we extract the feature points in the first image frame. The extracted feature points are tracked across the image sequence. If a tracked feature point is revealed as an outlier, it is excluded from the feature set. Feature points failed in matching are also excluded from the set. If the number of feature points in the current set becomes smaller than a predefined threshold on the way of the tracking, new feature points are extracted on the current frame and they are added into the feature set. In the second step, we predict the feature locations in the current frame considering the state vectors in the history buffer. Since the real-time video stream has a sufficiently high frame rate, it is reasonable to assume that the current feature position can be predicted around the previous feature location using the state vector. We predict the feature positions by finding the locations of the smallest error. The predicted feature position is used to initialize the location of the search window. Each predicted feature position is further refined using the measurement data. For the refinement, we compute the similarity of the image patches centered at the feature locations. The final refined location of the feature point corresponds to the location of the maximum similarity. Once all the feature correspondences are computed, we choose and exclude the outliers. Outliers are appeared mainly due to the occlusion. Rejection of outliers is also a necessary step for the improvement of feature tracking accuracy. When the final feature location is determined, we update the state vector and push them into the history buffer.

Step 1 (Initialization):
- 1.1 Capture and load the first frame. Set $n \leftarrow 1$.
- 1.2 Extract N feature points from the first frame.

Step 2 (Prediction):
- 2.1 Capture and load the next frame. Set $n \leftarrow n + 1$.
- 2.2 Predict the feature state vector $\mathbf{p}_n^i$ for the i'th feature point using the state history buffer.
- 2.3 Predict the feature location using Eq. (3).

Fig. 1. Input sample frames acquired during camera translation (top), rotation (middle), and scaling (bottom).

2.4 Set the search window w_i based on the predicted feature location for the i'th feature point.

Step 3 (Tracking features):

3.1 Compute the similarity measure for the i'th feature point at w_i.

3.2 Determine the final feature location.

Step 4 (Rejecting outliers):

4.1 Choose outliers using the RANSAC algorithm and exclude them from the feature set.

4.2 Correct the feature state vector $\mathbf{p}_n^i$ using the feature set.

4.3 Update the state vectors and push them into the state history buffer.

Step 5 (Adding new features):

5.1 Check the number of the feature points in the feature set. If the number is less than the given threshold then extract and add new feature points into the feature set.

5.2 Go to Step 2.

3 Experimental Results

We apply our tracking method to several long video sequences. Each sequence contains at least 100 frames. The input video frames are photographed in general indoor and outdoor environments. The camera was moved arbitrary in the physical environment. However, all the captured objects are assumed to be static.

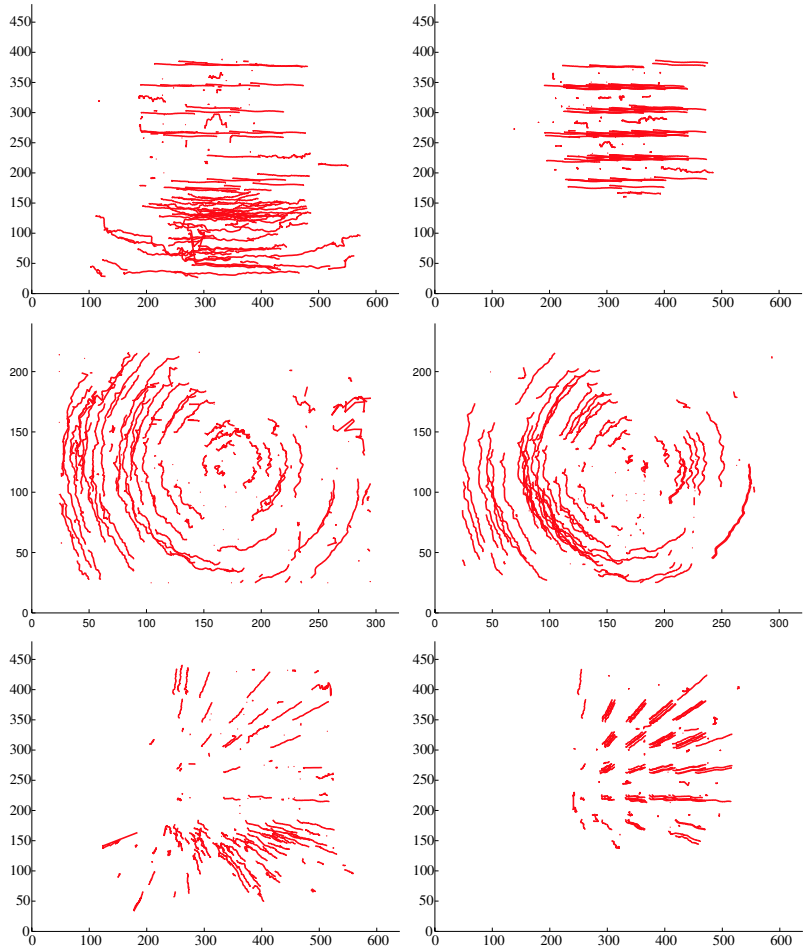

Fig. 2. Trajectories of tracked feature points for the input frames in Fig. 1.

Fig. 1 shows three sets of sample images taken from three different static scenes with a moving camera. The three sequences contain camera motions of translation, rotation, and scaling, respectively. We initially extract 300 feature points and track them over 100 frames. Fig. 2 shows the tracking results. The feature extraction method used in our method is robust for the rotation and scaling as shown in the tracking results in Fig. 2. We compared the accuracy of our method and the KLT tracking method. The left figure shows the result of the KLT tracking method and the right figure shows the result of our method. Comparing the two results, our method shows a better result mainly due to the outlier rejection. Fig. 3 shows the tracking error measured along the 100 frames for the translation sequence shown in Fig. 2. Because the camera was moved in a

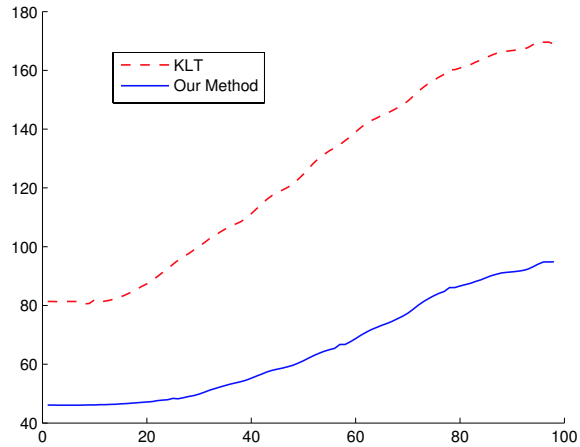

Fig. 3. Errors of the tracked feature points for the translation sequence.

parallel direction with respect to the target object, the vertical image coordinates of all feature points are nearly the same. The errors were computed by summing all differences of each y-coordinate value and the average y-coordinate value. As shown in the error graph, our method gives better results than the KLT tracker.

4 Conclusion

This paper presented a novel feature tracking method that could track point features robustly even on long image sequences. Our approach is different from previous works in the sense that the proposed method effectively utilizes the history of the feature movements. We extract feature points that are invariant to 2D transformations. Using the feature state history, we predict the next feature positions and refine the locations by computing similarity measures. For each frame, outliers are detected and they are excluded from the feature set. Newly appeared features are instantly added to the feature set to persist tracking on a long sequence.

Our tracking method might be useful for the implementation of various augmented reality applications and 3D reconstruction applications. As long as reliable feature trajectories are provided, the accurate estimation of the 3D structure is also possible. Hence the proposed method is also useful to improve the accuracy of the model-based 3D object tracking.

However, good tracking results are not always guaranteed due to irregularity and diversity of the real environment. The feature point set could possibly contain unexpected outliers and, hence, more accurate outlier rejection scheme is required to reduce the possibility of false matching. Our future research includes improving the robustness of feature tracking to ensure there is no outlier in the

feature set. Our future work also includes inferring the camera poses and the structures of 3D objects from the tracked feature points.

Acknowledgements. This work was supported by grant No. RTI05-03-01 from the Regional Technology Innovation Program of the Ministry of Knowledge Economy.

References

1. R. Hartley and A. Zisserman, "Mutiple view geometry in computer vision," *Cambridge University Press*, 2000.
2. J. Shi and C. Tomasi, "Good features to track," *in IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, 1994, pp. 593-600.
3. T. Ziner and C. Gräßl and H. Niemann, "Efficient feature tracking for long video sequences," *in DAGM*, Vol. 3175, 2004, pp. 326-333.
4. D. Lowe, "Object recognition from local scale-invariant features," *in International Conference on Computer Vision*, 1999, pp. 1150-1157.
5. D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2003.
6. T. Tommasini and A. Fusiello and V. Roberto and E. Trucco, "Robust feature tracking," *in Proceeding of the Joint Workshop of AI*IA and IAPR-IC*, 1998, pp. 93-98.
7. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *in ACM*, Vol. 24, 1981, pp. 381-395.
8. J. Matas and O. Chum, "Randomized RANSAC with sequential probability ratio test," *in IEEE International Conference on Computer Vision*, Vol. 2, 2005, pp. 1727-1732.