Adaptive Window Strategy for High-Speed and Robust KLT Feature Tracker

Nirmala Ramakrishnan[1(✉)], Thambipillai Srikanthan[1], Siew Kei Lam[1], and Gauri Ravindra Tulsulkar[2]

[1] Nanyang Technological University, Singapore, Singapore
rnirms@gmail.com
[2] Manipal Institute of Technology, Manipal, India

Abstract. The Kanade-Lucas-Tomasi tracking (KLT) algorithm is widely used for local tracking of features. As it employs a translation model to find the feature tracks, KLT is not robust in the presence of distortions around the feature resulting in high inaccuracies in the tracks. In this paper we show that the window size in KLT must vary to adapt to the presence of distortions around each feature point in order to increase the number of useful tracks and minimize noisy ones. We propose an adaptive window size strategy for KLT that uses the KLT iterations as an indicator of the quality of the tracks to determine near-optimal window sizes, thereby significantly improving its robustness to distortions. Our evaluations with a well-known tracking dataset show that the proposed adaptive strategy outperforms the conventional fixed-window KLT in terms of robustness. In addition, compared to the well-known affine KLT, our method achieves comparable robustness at an average runtime speedup of 7x.

Keywords: KLT feature tracker · Robust tracker · High-speed tracking

1 Introduction

Feature tracking is an essential step in many computer vision applications, such as global motion estimation, image registration and object tracking, and is used to extract higher level information about camera and/or object motion from the local optical flow vectors of each feature. The Kanade-Lucas-Tomasi (KLT) feature tracking algorithm [1] has been extensively used as it is very effective for small frame-to-frame displacements of features (which is common in video frames), and its low computational complexity enables it to be deployed on resource constrained platforms. In order to handle larger displacements, a multi-resolution approach is presented in [2], which is based on a pyramidal implementation of KLT.

However, KLT assumes that the patch around the feature undergoes only translation motion. Therefore, in the presence of rotation and scaling, KLT is known to suffer from high inaccuracies [3]. This is a severe limitation of KLT

T. Bräunl et al. (Eds.): PSIVT 2015, LNCS 9431, pp. 355–367, 2016.
DOI: 10.1007/978-3-319-29451-3_29

as drastic frame-to-frame rotation is common in many applications, for example videos captured on-board unmanned aerial vehicles that perform bank turns [3].

In this paper, we show that it is necessary for the window size in KLT to adapt to the presence of distortions around the feature points in order to increase the quality of the tracks. We present a novel adaptive window strategy for KLT that does not require explicit error estimate computations. In particular, we show for the first time that the iterations needed to converge within KLT can be used as a crude indicator of the quality of the feature track, thereby providing a simple means to arrive at a suitable window size. The proposed strategy is applied to each level of the pyramid in the pyramidal implementation of KLT to significantly improve the robustness to large displacements and distortions. Our extensive evaluations with simulated as well as annotated tracking datasets show that the proposed strategy significantly outperforms the translation model-based KLT in terms of robustness. In addition, the proposed strategy achieves comparable robustness as the affine model based KLT at 7x run-time speedup.

The paper is organized as follows. Section 2 presents the related work for improving robustness of KLT. Section 3 presents an overview of the KLT algorithm and an analysis of the relationship between the search window size, the motion experienced by the feature and the accuracy of the KLT tracker. Section 4 presents the proposed adaptive window strategy for KLT. In Sect. 5, we show evaluation results with simulated as well as a real annotated tracking dataset [4]. Paper concludes with Sect. 6.

2 Related Work

Feature correspondence algorithms can be classified into two categories: (a) feature matching that uses a rich feature detector-descriptor combination [4], and (b) feature tracking. Feature matching can handle large rotation, scale and view point changes. However frames in video sequences predominantly undergo small translation motion and infrequent rotations/scaling motions between the frames, and therefore feature tracking is more commonly used due to its lower computational complexity.

Several techniques in feature tracking that accommodate the non-translation motions have been proposed and they can be categorized into four groups: (1) fusing the motion data from on-board inertial measurement units (IMUs) [3,5] (2) affine formulation of KLT that replaces the translation model with an affine model [6], (3) computation of error estimates to evaluate the feature tracks [7–9], and (4) employing adaptive window size [10]. The following describes existing work in these groups.

Inertial measurement units (IMUs) can provide the 3D motion of the vehicle and therefore this data can be used to arrive at initial estimates that KLT can then work with and this has been shown to be robust under drastic motions [3,5]. In this paper, we propose a technique that is not dependent on IMUs.

An affine formulation for KLT was proposed in [6] that replaces the translation model for the motion of the feature patch in the search window, with a more

complex affine model - this allows the feature patch to undergo rotation, scaling and skew. However, this is highly compute-intensive and real-time performance is achieved only with GPU implementations [11].

Another approach is to evaluate the tracking algorithm in an online manner and discard tracks that are of poorer quality, and a good survey on this area of work is found in [7]. Such evaluations associate an error/uncertainty estimate with every track and this allows the applications that use the feature tracks to detect when KLT is unable to deal with the motion. In [8], the time-reversibility constraint of trackers has been applied to compute a forward-backward error for tracks, and this metric is used to discard potentially erroneous tracks. In [9], theoretical error estimates for KLT tracks is presented, however it is reported that the error estimate computations are so high that they cannot be used in real-world applications in an online manner.

The work in [10], which employs adaptive window size is the most similar to ours. The authors used intensity and disparity variations within the search window to compute uncertainty estimates for the displacement found by the tracking algorithm and find an optimal window size that minimizes this uncertainty estimate. However this iterative window sampling is performed for each core KLT operation to estimate the displacement and this adds a huge computation cost to the tracking algorithm. In contrast, our proposed method does not explicitly compute uncertainty estimates and monitors the KLT performance for a given window size as an indicator of successful tracking. The main contributions of this paper are as follows:

1. We analyze the relationship between the search window size, the motion experienced by the feature and the accuracy of the KLT tracker to show the need for adapting the window size of KLT in order to enhance its robustness to distortions.
2. We propose a novel adaptive window strategy that can be applied to pyramidal KLT implementation. We show that the iterations needed to converge within KLT can be used as a crude indicator of the window size being too small, and this can guide the adaptive strategy to land at a suitable window size.
3. Using simulated as well as a real annotated tracking dataset, we demonstrate that the proposed adaptive strategy outperforms the conventional fixed-window KLT in terms of robustness, and achieves significantly lower runtime without sacrificing robustness when compared to the well-known affine KLT.

3 Overview of KLT Algorithm

The goal of the KLT tracking algorithm is to find the displacement $d = [d_x, d_y]^\top$ for a feature x in two images *I(x)* and *J(x)* such that the residual error $\varepsilon(d)$ defined in (1) is minimised:

$$\varepsilon(d) = \sum_{W} [I(x) - J(x+d)]^2 \tag{1}$$

The residual $\varepsilon(d)$ is the intensity difference computed for a region of support defined by the search window W around the feature x. Given a starting position for d(initialised to $d = 0$), (1) solves the least squares optimization problem by iteratively modifying d as $d \leftarrow d + \Delta d$, through a Newton-Raphson method, such that $\varepsilon(d)$ is minimised. (1) assumes a translation model and is sufficient when the motion between the video frames is uniform. In the presence of fast rotation or scaling motions, this translation model fails. In order to account for the tracking challenge of rotation and scaling with KLT, an affine model was proposed in [6]. The residual error is defined as in (2) with the affine model:

$$\varepsilon(d) = \sum_{W} [I(x) - J(Ax + d)]^2 \tag{2}$$

Here, A is the affine transformation matrix given by:

$$A = \begin{bmatrix} 1 + d_{xx} & d_{xy} \\ d_{xy} & 1 + d_{yy} \end{bmatrix} \tag{3}$$

The six parameters $(d_x, d_y, d_{xx}, d_{yy}, d_{xy}, d_{yx})$ allow rotation, anisotropic scaling, skew and translation. However, this robustness comes with a computation cost tradeoff. Therefore, we use the translation model as the basis of our proposed method.

3.1 Effect of Search Window Size for Rotation/Scaling

As shown in Fig. 1(a), given a maximum inter-frame displacement d_{max}, it is advisable to setup the search window size $W = 2 * d_{max} + 1$, such that the displacement of the feature is captured within the KLT search window [2]. Figure 1(b) illustrates the translation model assumed by KLT as in (1) in which all the pixels within the search window W have the same displacement given by (d_x, d_y). This assumption is valid only when the feature itself undergoes translation motion. As shown in Fig. 1(c) when there is rotation, the displacement of each of the pixels in W varies and this violates the underlying assumption of a translation motion model by KLT.

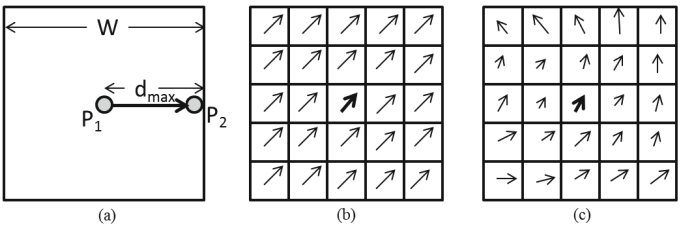

Fig. 1. Search Window Size and Feature Patch Displacement (a) Window size W needs to capture maximum displacement d_{max} (b) Displacements of neighboring pixels for translation motion (c) Displacements of neighboring pixels for rotation motion

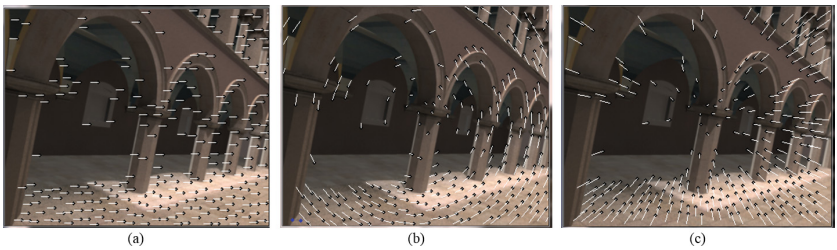

Fig. 2. Local feature motions for various global motion (a) translation (b) rotation (c) scaling

When the feature undergoes translation motion alone, having a larger window size is always preferable because any noise in the intensity patterns within the window is averaged out. However, when the frame undergoes rotation or scaling then the pixels within the patch have varying displacement. As we move further away from the center pixel, these variations increase – therefore, relying on the farther away pixels results in higher error in the KLT results. In such cases, typically the center pixel undergoes a small displacement and therefore, this can still be captured with a small window size without including potentially erroneous estimates from far away pixels.

Hence, for images undergoing rotation and scaling, the window size needs to be large enough to capture the displacement of the feature but small enough to not invite potentially erroneous estimates from pixels far from the center of the feature. Similar to the conclusions in [10], there exists an optimum window size that results in the most accurate track for KLT for a given feature patch and inter-frame motion.

When the frame undergoes a rotation or scaling motion, as shown in Fig. 2(b) and (c), then the local displacement and distortion experienced by different features depends on their location in the frame. Therefore, an optimum window size needs to be determined for each individual feature in the presence of global rotation and/or scaling.

3.2 Implications of Fixed Search Window Size with Pyramidal KLT

The KLT tracker employs linear approximation in its core step which works only if the displacement Δd is very small. In order to overcome this limitation, a multi-resolution pyramidal approach was proposed in [2] to deal with larger inter-frame displacements. KLT starts at the highest level of the image pyramids for both frames and the displacement found at this level is up-sampled and used as the initial estimate for the next lower level in the pyramid. This allows the use of a reasonable window size at the highest level to capture a large displacement – for example, with a pyramid of 3 levels, a displacement of 32 pixels is reduced to $32/2^3 = 4$ pixels at the highest level $L3$, and can be captured with a window size of $W = 9$. Once this displacement has been reasonably captured by a higher

level and is passed on to the lower level, this lower level needs to only capture any error in this initial estimate.

For example, if the KLT result at *L3* is 3.75 and the actual displacement is 4 pixels, then we have an error of 0.25 pixels which translates to 0.5 pixels when up-sampled for *L2*. At *L2*, using the same window size of $W = 9$ may be too large for this displacement of 0.5 pixels in the presence of distortions. Therefore, when the displacement itself has to be captured, the window size needs to be large enough. But once it has been captured, and we only need to "refine" this estimate with higher resolution in the lower levels of the pyramid, then a smaller window size is more appropriate to capture the errors in the initial estimates.

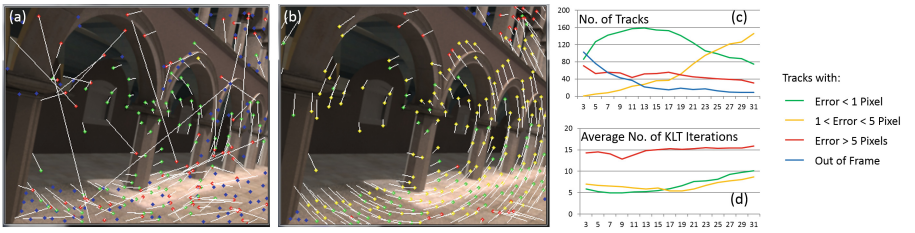

Fig. 3. KLT tracks with in-plane rotation of 10° (a) W = (3,3), (b) W = (31,31). For a range of window sizes (x-axis) (c) Total no. of tracks categorised based on tracking error (d) Average no. of iterations taken by KLT to converge for each category of tracks (Color figure online)

In conclusion, KLT is susceptible to errors with a fixed window size in the presence of rotation and scaling. Given a feature with an associated motion, there exists an optimal window size that is large enough to capture the displacement of the feature but small enough to not be affected by the distortion that the neighborhood of the feature undergoes. Based on our analysis in Sects. 3.1 and 3.2, we can conclude:

- In the presence of rotation and scaling, a fixed window size cannot be applied to all features in the frame. It needs to adapt to the motion experienced by each individual feature.
- Within the pyramidal implementation, the displacement that needs to be captured at the highest level is different from the other lower levels, and therefore the window size needs to adapt to the displacement that needs to be captured at each level.

3.3 Tracking Errors and KLT Iterations

Figure 3 shows the KLT tracks for a global in-plane rotation of 10°. Green dots represent tracks with error less than 1 pixel. Yellow dots represent tracks which have error between 1 and 5 pixels. Red dots represent tracks with error greater

than 5 pixels. When the window size is too small to capture the required displacement as in Fig. 3(a), the tracks drift away and result in high tracking error (represented by red tracks) or are pushed out of the frame (represented by blue dots). This is because with a smaller region of support the KLT estimates are easily skewed by noisy pixels within the search window. In contrast when the window size is too large as in Fig. 3(b), the estimates from the noisy pixels get averaged out and the displacement is comfortably captured, resulting in the KLT estimates being close to the final location. However, as there is distortion due to rotation, the far away neighbors tend to skew the KLT estimate and tracking error increases resulting in many yellow tracks. Figure 3(c) shows this trend when window size is varied, which implies that the window size needs to be optimal in order to reduce the chances of a track being red or yellow. Figure 3(d) shows that the average number of iterations is significantly higher for the lost red tracks. However, the yellow tracks converge fast and cannot be distinguished easily from the green tracks at the end of KLT. Therefore, the iterations can be used as an indicator to detect tracking errors caused by a window size that is too small to capture the displacement of the feature.

4 Adaptive Window Size for KLT

We propose an adaptive window strategy for KLT to improve its robustness in the presence of distortions due to rotations and scaling, such that an appropriate window size is chosen for each track resulting in higher number of green tracks and reducing the noisy tracks – both red and yellow. Figure 4 shows the flow of the proposed strategy.

As iterations can indicate tracking error associated with too small a window size, the proposed strategy starts with a small window size and increases the window size only if needed, as indicated by the KLT iterations. Specifically we evaluate the track with the current window size with the following checks:

1. Current track converged within the maximum iterations (*MaxIterations*): The *MaxIterarions* is a user-defined parameter that determines the number of iterations that KLT will run before giving up.
2. Fast convergence for two consecutive window sizes: Not all red tracks hit the maximum iterations. Sometimes, tracks converge fast for a certain window size at a local minimum. However increasing the window size, exposes more of the image and the track is pushed out of the local minimum. In order to prevent such early convergences, we check if the previous window size sampled also had a fast convergence. We set a minimum iterations threshold (=8 iterations) to declare the convergence as fast.
3. Forward-backward error as defined in [8] is within a threshold of 1 pixel: As a final check to reign in lost features (red tracks) that might have converged fast, we employ the forward backward error.

This adaptive window size strategy is applied to each feature at each level of the pyramid in the pyramidal KLT implementation so that an appropriate

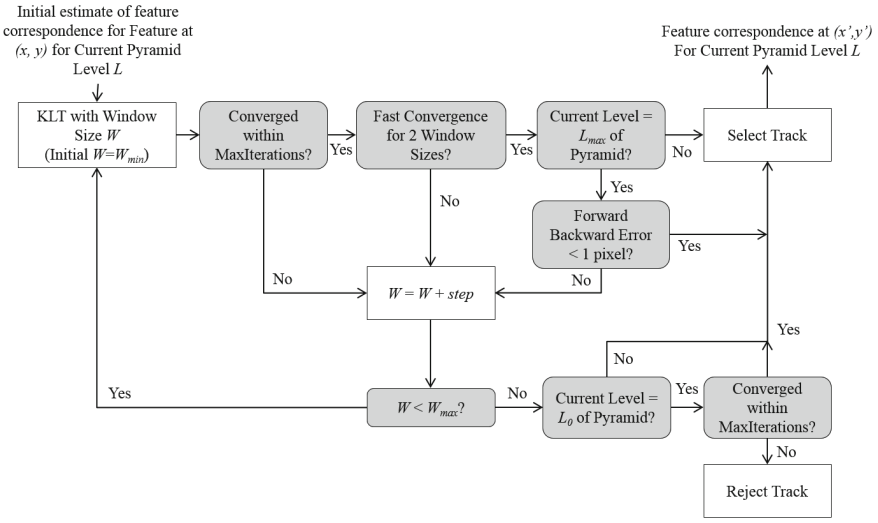

Fig. 4. Adaptive Window Strategy for Each Pyramid Level of the KLT Feature Tracking Algorithm

Fig. 5. Image data used for the simulated motions

window size is used depending on the displacement that needs to be captured at that level of the pyramid. Specifically, this allows the window size to grow as needed at the highest level when the displacement to be captured is unknown. Once the displacement has been captured, at the subsequent lower levels, smaller window sizes result in fast convergences avoiding noisy estimates from far away neighbors in the presence of distortion.

5 Evaluations

In order to evaluate the proposed adaptive window size strategy for KLT, we compare it with these baselines: (a) Conventional translation-model KLT tracking algorithm with a fixed window size W_{fixed}, referred to as *Conv KLT* (b) Affine-model KLT as in [6] with a fixed window size W_{fixed}, referred to as *Affine KLT* henceforth. The proposed method is referred to as *Adaptive KLT*.

We use the KLT implementations in OpenCV v2.4.9 for the conventional (*calcOpticalFlowPyrLK*) and affine KLT (*cvCalcAffineFlowPyrLK*) baselines. We implemented the *Adaptive KLT* by modifying the *calcOpticalFlowPyrLK* function. For the baseline KLT algorithms we used these parameter values:

W = (31,31), *MaxIterations* = 20, *minimum Eigen threshold* = 0.0001. For the *Adaptive KLT*, W_{min} = (5,5), W_{max} = (31,31), *step* = 2, *minIterations* = 8. All other parameters are the same as the baseline algorithms.

For our evaluations we selected features using the Shi-Tomasi feature detector [12] and applied each of the variants of the KLT tracker. The correspondence found for each feature is compared with the ground truth and the tracking error is computed. If the error is below 1 pixel, the track is labeled *useful*, else it is labeled as *noisy*. The robustness of the tracker to various motions is determined by the extent to which it can maximize the number of *useful* tracks and minimize the number of *noisy* tracks.

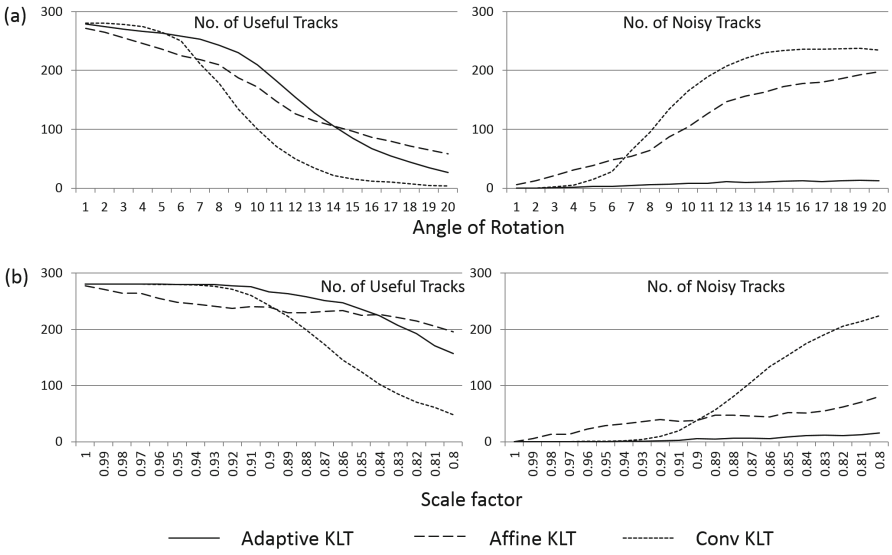

Fig. 6. Number of useful and noisy tracks for varying (a) rotation angles and (b) scale factors

A range of in-plane rotations of angles 0–20° and scale factor of 0-0.8 is applied on the images shown in Fig. 5. The results are shown in Fig. 6 as an average over all the images for a 3-level pyramid implementation. For angles in the range of 1–5°, all the KLT variants perform similarly. In the range of 6–15°, *Conv KLT* sees a drastic drop in the no. of useful features, and a corresponding increase in the no. of noisy features. This indicates an increase in the overall tracking error. The *Affine KLT* is able to salvage many of the noisy tracks and hence shows an improvement in the no. of useful tracks compared to *Conv KLT*. However the *Adaptive KLT* is the most successful in eliminating the noisy tracks and matches the *Affine KLT* in the number of useful tracks reported, thus generating the cleanest feature set among all the KLT variants considered for this range of angles. Beyond 15°, all the 3 variants of KLT are unable to deal with the distortion incurred.

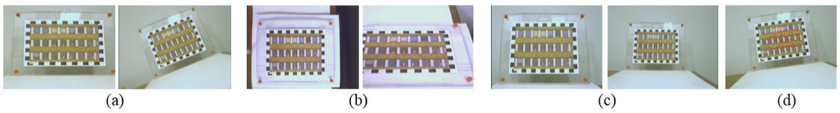

Fig. 7. Sample frames from the tracking dataset used for our evaluations (a) Rotation (b) Perspective Distortion (c) Zoom (d) Shi-Tomasi features (red) selected within the texture area (Color figure online)

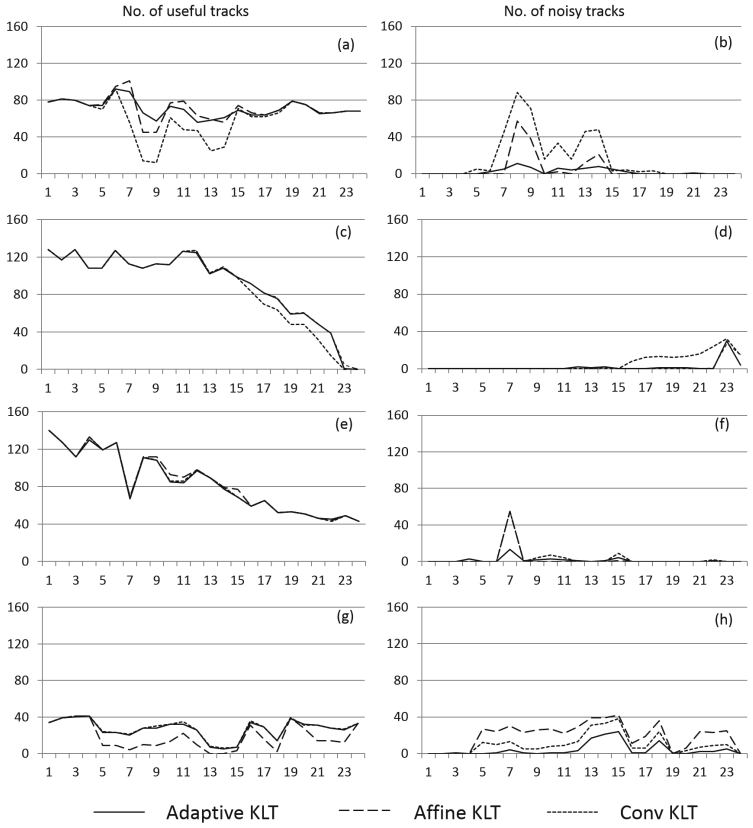

Fig. 8. Robustness Results for Tracking Dataset in [4] (a)-(b) Rotation (c)-(d) Perspective Distortion (e)-(f) Zoom (g)-(h) Panning

We also provide results with a ground truth annotated tracking dataset in [4] shown in Fig. 7, for the videos that incur distortions – rotation, zoom, perspective distortion and panning – for the texture "bu" (building). The Shi-Tomasi features considered are all in the rectangular texture area and features close to the boundary are ignored to eliminate boundary effects as in Fig. 7(d). Features are selected in every frame and tracked in the next frame. As the frame-to-frame motion is very small for this dataset, we skip every other frame and show the

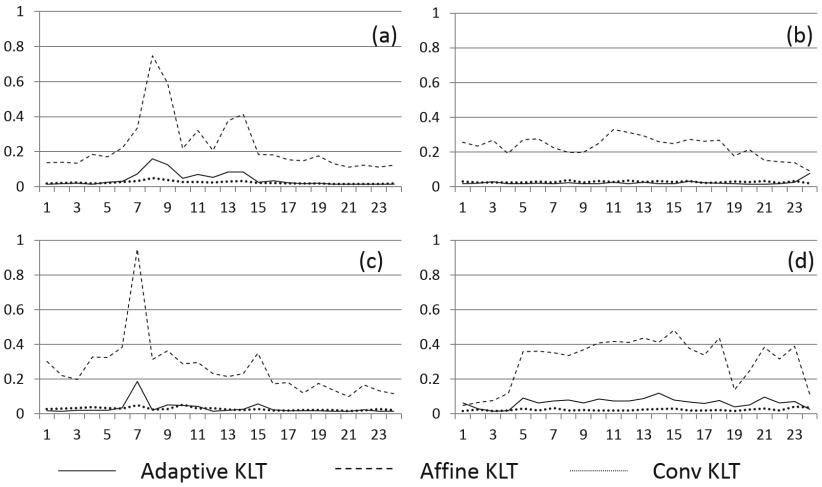

Fig. 9. Computation time (a) Rotation (b) Perspective Distortion (c) Zoom (d) Panning motions

results for this wider baseline video sequence. We use a 2-level pyramid for all the KLT methods.

We show the number of useful and noisy feature tracks for all the KLT variants in Fig. 8. Rotation, as shown in Fig. 8(a), (b), incurs the most inter-frame distortions among all the 3 motions considered. *Conv KLT* suffers in frames 7–9 and 13–15 and *Affine KLT* improves the number of useful features, however it is unable to keep the noisy tracks to a minimum. *Adaptive KLT* outperforms the *Affine KLT* both in the number of useful and noisy tracks. For perspective distortion, the number of features goes down towards the end of the video sequence as very less area of the texture is seen. *Conv KLT* suffers in terms of robustness, but both *Affine* and *Adaptive KLT* show similar robustness. The zoom video sequence in Fig. 8(e), (f) has a zoom-out motion, and therefore the area covered by the texture goes down as the video progresses. This is reflected in the no. of tracks reported by all KLT variants falling. In frames 6–8 both the *Conv*and *Affine KLT* suffer in terms of number of noisy tracks but *Adaptive KLT* keeps the noisy tracks to a minimum. In the panning video sequence in Fig. 8(g), (h), the displacement is large and the distortion in minimal. *Adaptive KLT* has comparable performance with the *Conv KLT*, but *Affine KLT* is unable to handle the large displacement. Overall, the robustness results show that even though *Affine KLT* is designed to allow distortions in the template patch, it is still outperformed by *Adaptive KLT*. This can be attributed to the large fixed search window size that is used with *Affine KLT* which prevents it from reaching the accuracy, that a more controlled optimal window size reached by the *Adaptive KLT* can offer.

We measured the computation times on a 3.5 GHz Intel (R) Xeon (R) desktop computer. As shown in Fig. 9, *Affine KLT* being the most computationally

complex of all the methods considered here, results in the highest computation times – incurring an average computation time 7x times more than *Adaptive KLT*. As *Adaptive KLT* iteratively samples various window sizes, in the presence of distortion, the computation time is marginally higher than the *Conv KLT* which uses a single fixed window. The *Affine KLT* is only able to operate at an average of 5 frames per second (FPS), while the *Adaptive KLT* and *Conv KLT* operates at an average of 39.6 FPS and 41.5 FPS respectively. This shows that the proposed *Adaptive KLT* is not only robust against distortions but it can also be used in real-time applications.

6 Conclusions

In this paper, we have proposed novel adaptive window size strategy for the classical Kanade-Lucas-Tomasi (KLT) feature tracker in order to make it robust against distortions due to rotations and scaling. We show that the search window size determines the accuracy of the KLT feature tracker, and in the presence of distortions around the feature due to rotation and scaling, this window size needs to adapt to the displacement to be captured. The proposed adaptive strategy adopts a controlled selection of window size by sampling various window sizes starting with a small value.

By monitoring the KLT performance in terms of iterations for each window size, our proposed method can determine failure to capture the displacement, and lands at a near-optimal window size when the iterations stabilize. This way, it is able to capture the displacement with a large-enough window size and yet avoids the effects of distortion in the patch. Our evaluations on a tracking dataset show that the proposed strategy significantly outperforms the conventional fixed-window KLT in terms of robustness against rotation and scaling, and achieves the robustness of the more complex affine KLT with 7x faster runtime.

References

1. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of International Joint Conference on Artificial intelligence 1981, pp. 674–679 (1981)
2. Bouguet, J.-Y.: Pyramidal implementation of the Lucas Kanade feature tracker. Intel corporation, Microprocessor research labs (2000)
3. Tanathong, S., Lee, I.: Translation-based KLT tracker under severe camera rotation using GPS/INS data. IEEE Geosci. Remote Sens. Lett. **11**, 64–68 (2014)
4. Gauglitz, S., Höllerer, T., Turk, M.: Evaluation of interest point detectors and feature descriptors for visual tracking. Int. J. Comput. Vis. **94**, 335–360 (2011)
5. Hwangbo, M., Kim, J.-S., Kanade, T.: Inertial-aided KLT feature tracking for a moving camera. In: International Conference on Intelligent Robots and Systems, pp. 1909–1916 (2009)
6. Bouguet, J.-Y.: Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm. Intel Corporation (2001)

7. SanMiguel, J.C., Cavallaro, A., Martínez, J.M.: Adaptive online performance evaluation of video trackers. IEEE Trans. Image Process. **21**, 2812–2823 (2012)
8. Kalal, Z., Mikolajczyk, K., Matas, J.: Forward-backward error: automatic detection of tracking failures. In: 20th International Conference on Pattern Recognition (ICPR), pp. 2756–2759 (2010)
9. Sheorey, S., Keshavamurthy, S., Yu, H., Nguyen, H., Taylor, C.N.: Uncertainty estimation for KLT tracking. In: Jawahar, C.V., Shan, S. (eds.) ACCV 2014 Workshops. LNCS, vol. 9009, pp. 475–487. Springer, Heidelberg (2015)
10. Okutomi, M., Kanade, T.: A locally adaptive window for signal matching. In: 3rd International Conference on Computer Vision, pp. 190–199 (1990)
11. Kim, J.-S., Hwangbo, M., Kanade, T.: Realtime affine-photometric KLT feature tracker on GPU in CUDA framework. In: 12th IEEE International Conference on Computer Vision Workshops, pp. 886–893 (2009)
12. Shi, J., Tomasi, C.: Good features to track. In: IEEE Computer Vision and Pattern Recognition, pp. 593–600 (1994)