Efficient Point Feature Tracking based on Self-aware Distance Transform

Min-Gyu Park
mpark@gist.ac.kr
Kuk-Jin Yoon
kjyoon@gist.ac.kr

Computer Vision Lab.
Gwangju Institute of Science and Technology (GIST), Korea
http://cvl.gist.ac.kr

Abstract

In this paper, we propose a Self-aware Distance Transform (SDT) for efficient template-based point feature tracking. The proposed SDT encapsulates the relationship between autocorrelation coefficients and the distance from the best match; therefore, it can be used to automatically determine the size of a search region in each point feature. The proposed SDT returns the expected distance between the predicted position and the best match from a statistical viewpoint, which guarantees a certain level of successful tracking depending on the cross-correlation at the predicted position. If the SDT returns a large expected distance due to the abrupt motion of a feature or inaccurate prediction, we progressively expand the search region on a hexagonal lattice while also using the SDT to reduce unnecessary computations. The performance of the proposed tracking method based on the SDT was verified experimentally in terms of its accuracy, robustness, and computational efficiency by comparing the proposed method to other tracking methods.

1 Introduction

The tracking of point features is an essential problem in computer vision because the acquisition of feature correspondence from successive frames is a front-end step in many problems such as target tracking, simultaneous localization and mapping, and video stabilization. Feature tracking algorithms can be divided roughly into three categories, i.e., tracking-by-detection [8][2][11], tracking-by-template matching [16][7][5], and tracking-by-Lucas-Kanade-Tomasi tracker(KLT) [17][19][1][6]. Each of these methods has its own advantages and disadvantages.

Tracking-by-detection initially detects local features, e.g., corners [13], blobs [8], and maximum extreme points[4], in every frame and matches them using local feature descriptors such as SIFT [8][10][18] and SURF descriptors [2]. These descriptors are designed to ensure robustness against image transformations, e.g., rotation and scale, as well as perspective distortions in images. However, the processes of feature descriptor computation and feature matching are computationally demanding. Thus, previous studies have focused on improving the computational efficiency of these steps by reducing the dimension of descriptors [18], or by reducing the number of matching candidates using specific data structures [14]. Several implementations can operate in real-time [18], but the selection of appropriate local feature detectors and descriptor-matching algorithms is an important issue that

http://dx.doi.org/10.5244/C.26.32

affects tracking performance. The KLT tracker has been studied extensively [17][19][1][6] and used for feature tracking. KLT is based on the small motion assumption[1] and initially KLT computes the optical flow of a feature and performs nonlinear optimization. Thus, the estimation of the initial optical flow is an important problem. Pyramid-based [19], template matching-based [17], and sensor-based [6] approaches can be used to compute the initial motion accurately.

In addition to these approaches, template-matching techniques have also been used frequently for feature tracking, and for other vision problems such as visual tracking and stereo matching. This approach inherits the intrinsic characteristics of the template-matching problem, so previous studies have focused mainly on increasing the speed of matching [7][5], improving the computational search efficiency [16][5], and developing robust similarity measures. For example, the use of an integral image [7] is a well-known technique for increasing the matching speed during normalized cross-correlation (NCC) while a hierarchical approach [16] is also used widely for improving the search efficiency. However, the size of a search region is still retained as a predefined parameter, although the size of a search region affects the performance of an algorithm significantly. A larger search region increases the robustness of an algorithm against abrupt motions but it requires greater efforts in searching and avoiding false alarms. By contrast, a smaller search region increases both the efficiency and missing probability of an algorithm. Thus, there is a trade-off between computational efficiency and robustness, which depends mainly on the size of the search region.

To avoid this dilemma, we propose a Self-aware Distance Transform (SDT) with an efficient feature-tracking method. The aim of the SDT is to estimate the optimal search region size based on autocorrelation with a template in the initial frame. We use the spatial relationship of the cross-correlation coefficients relative to the best match as the function of a coefficient in the predicted position of a feature. Autocorrelation has been used to determine the optimal grid interval [15] and to determine the transitive bound of template matching [8] during video coding in previous studies, which were focused on increasing the speed of a full search algorithm (FSA).

We analyze the reliability of our proposed SDT based on the autocorrelation coefficient statistics to determine the reliable range of the SDT. If the returned SDT value is outside this range, we progressively expand the search region on a hexagonal lattice while using the SDT to eliminate unnecessary computations. The expansion step is terminated if a reliable match is found. Therefore, the proposed tracking method increases the computational efficiency and robustness of feature tracking thanks to the proposed SDT.

In this paper, we provide a detailed description of our proposed algorithm in section 2 including the SDT and the feature-tracking framework. In section 3, we present our experimental verification of the proposed method, which is compared with other algorithms in terms of its time complexity, robustness, and accuracy. In section 4, we summarize and conclude the paper.

2 The Proposed Method

2.1 The Objective and our Framework

Let $I(x,y)$ and $T(x,y)$ denote the intensity of an image and the intensity of a template in an image coordinate, respectively. Our task is to find the position of the template in a source

[1] KLT assumes that the appearance of a template changes little in time and in space.

image by the maximizing similarity or minimizing dissimilarity measures of the template. If we use the normalized cross-correlation (NCC) of a template as the similarity measure, it can be represented as follows:

$$\begin{aligned} &\mathbf{x}^* = arg\max_{x,y\in\Omega_R} NCC(x,y), \\ &NCC(x,y) = \sum_i \sum_j \frac{(T(i,j)-\mu_T)(I(x+i,y+j)-\mu_I)}{\sigma_T \sigma_I}, \end{aligned} \tag{1}$$

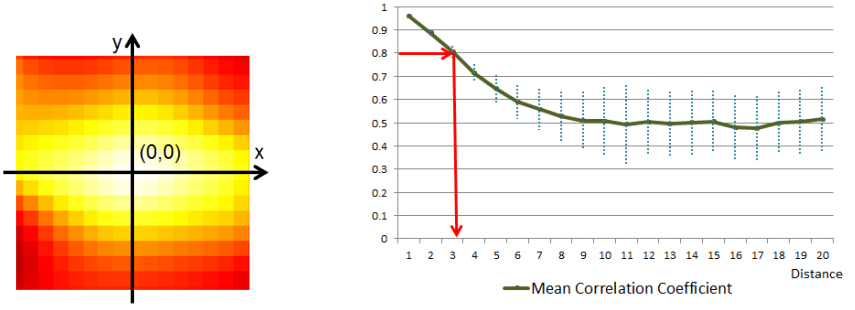
where Ω_R indicates an M-by-N search region and $\mathbf{x}^*$ is the vector notation of the best match position. In the rest of this paper, we use NCC as the similarity measure[2].

Algorithm 1 Overall procedure of the proposed tracking algorithm based on SDT

1: **Given**: $I_t(x,y)$ and $\mathbf{x}_t$ ▷ Initially, t is the time when the feature is extracted
2: Compute SDT ▷ See Section 2.2
3: $\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + (\mathbf{x}_t - \mathbf{x}_{t-1})$ ▷ Predict the feature position in the next frame
4: $\hat{d}_{t+1} = SDT(\hat{\mathbf{x}}_{t+1})$ ▷ Compute the predicted distance to the feature
5: **if** $\hat{d}_{t+1} \leq d_{max}$ **then**
6: Search for the best match $\mathbf{x}_{t+1}$ in $\hat{d}_{t+1}$ centered on $\hat{\mathbf{x}}_{t+1}$
7: If the best match does not satisfy the termination criteria, then $t = t+1$ and go to line 3.
8: **else**
9: Expand the search region in a hexagonal lattice (initially $l = 1$)
10: **for** $k = 1, ..., n$ **do** ▷ $n = 6l$, see Section 2.3
11: $\hat{d}_{t+1}^{(l,k)} = SDT(\mathbf{x}_{t+1}^{(l,k)})$ ▷ the indexing rule (superscript) is explained in Section 2.3
12: **if** $\hat{d}_{t+1}^{(l,k)} \leq d_{max}$ **then**
13: Search for the best match $\mathbf{x}_{t+1}$ in $\hat{d}_{t+1}^{(l,k)}$ centered on $\hat{\mathbf{x}}_{t+1}^{(l,k)}$
14: If the best match does not satisfy the termination criteria, then $t = t+1$ and go to line 3.
15: **else**
16: If we can increase l, $l = l+1$ and go to line 9.
17: ▷ The maximum l should be determined for the termination, which we set as 5
18: **end if**
19: **end for**
20: **end if**

The overall procedure of our framework is shown in Alg. 1. After a new feature (and the corresponding template) has been extracted, we build the SDT function via the autocorrelation step. A detailed description of the construction of the SDT function is provided in section 2.2. Next, we predict the position of a feature in a successive frame using the constant velocity prediction model and we compute the expected distance to the best match using the SDT. If the prediction is sufficiently reliable (i.e., the cross-correlation is high), we search for the best match within the expected distance predicted by SDT using either a three-step search (TSS) algorithm [5] or exhaustive search. However, if the expected distance is large and outside the reliable range (which is described later), we expand the search region progressively until we find the correct solution. To expand the search region, we use a honeycomb structure instead of a rectangular model, which increases the search efficiency. The detailed procedure used for search region expansion is explained in section 2.3.

[2] It is possible to use any other robust similarity measure with the proposed method.



(a) (b)

Figure 1: Illustration of the SDT; (a) the result of autocorrelation and (b) the relationship between the mean values of the autocorrelation coefficients and distance. The dotted vertical line indicates the variance of the autocorrelation coefficients at the same distance.

2.2 Self-aware Distance Transform (SDT)

Determining the optimal size of a search region is an important but difficult problem because it is impossible to predict how much of a feature will move into the next frame without external measurements. The SDT tackles this problem by spatial autocorrelation with a template to estimate the proximity of the best match in a successive frame by comparing the correlation coefficient at a predicted position with the autocorrelation information. After extracting a feature and its corresponding template, we immediately perform autocorrelation of the image and the template using Eq. 1. We then generate a set of groups based on the distance from the best match as follows:

$$\begin{aligned} F &= \{F_1, F_2, ..., F_{M-1}, F_M\}, \\ F_k &= \{C(\mathbf{p}) | round(\sqrt{p_x^2 + p_y^2}) = k\} \text{ for } 1 \leq k \leq M, \end{aligned} \tag{2}$$

where F_k is a set of correlation coefficients with the same distance from the best match, $\mathbf{p} = [p_x \ p_y]^T$ is a relative position vector centered on the best match $(0,0)$, $C(\mathbf{p})$ is the autocorrelation coefficient at $\mathbf{p}$, and k ranges from 1 to the predefined constant M. This constant is tuned automatically during the last step, so the selection of this value is not a significant problem. Rather than using a continuous distance, we discretize the distance values for the group of pixels and this distance is computed using a Chamfer distance transform. Next, we compute the mean and variance of each group as follows:

$$\begin{aligned} \mu_k &= \frac{1}{|F_k|} \sum_{C(\mathbf{p}) \in F_k} C(\mathbf{p}), \\ \sigma_k^2 &= \frac{1}{|F_k|} \sum_{C(\mathbf{p}) \in F_k} (C(\mathbf{p}) - \mu_k)^2, \end{aligned} \tag{3}$$

where μ_k and σ_k indicate the mean and standard deviation of the autocorrelation coefficients at the distance k, while $|F_k|$ represents the cardinality of a set of correlation coefficients. These two statistics are the essence of SDT because they are used to the compute the optimal size of a search region. Figure 1 shows the autocorrelation result and the relationship between the mean and distance values. This relationship is used as a function of an NCC coefficient, which allows the size of a search region to be determined automatically at each prediction step. For example, Fig. 1 shows that the best match is probably within 3 pixels if

the correlation value is 0.8. Finally, the SDT is defined as a function of a real valued vector (the position of a feature), which yields a positive integer value as follows:

$$\begin{aligned} &SDT : \Re^2 \rightarrow N^+ \text{ s.t.} \\ &\hat{d}_{t+1} = arg\min_{1\leq k\leq M} |\mu_k - C_{t+1}(\hat{\mathbf{x}}_{t+1})|, \end{aligned} \tag{4}$$

where $C_{t+1}(\hat{\mathbf{x}}_{t+1})$ indicates the NCC coefficient of a predicted position (we use a subscript to indicate that this computes the NCC between the template and a successive image at time $t+1$) while the expected distance is computed by minimizing the difference between the mean value and the NCC coefficients. Indeed, the SDT can be used for any other prediction models; we use the constant velocity model for the prediction[3]. The expected distance contains uncertainty that is proportional to corresponding variance σ_k where k equals $\hat{d}_{t+1}$. To avoid unreliable estimation of expected distance, therefore, we restrict the range of the valid expected distance, $(0, d_{max}]$ is determined as follows:

$$\begin{aligned} &d_{max} = arg\max_k s_k\sigma_k, \\ &\text{where} \quad s_k = \begin{cases} 1 & \text{if } \sigma_k < \delta_D, \\ 0 & \text{otherwise}, \end{cases} \end{aligned} \tag{5}$$

where s_k is an indicator variable, which is 1 for variances lower than δ_D, i.e., the predefined threshold value. This threshold is one of the main parameters that affect the tracking performance and it was verified experimentally, as shown in section 3.

2.3 Addressing the Abrupt Motion of Features

There are two possible situations when using the SDT. First, the use of the cross-correlation coefficient at the predicted position is adequate if the expected distance is less than d_{max}. Thus, we only search the small search region determined by the SDT. Second, the precise estimation of an accurate distance is not feasible if the expected distance is greater than the d_{max}. Therefore, we define the abrupt motion of features as a sufficient condition for matching ambiguity and we assume that one of the main reasons for local motion estimation failure is the abrupt motion of features. Other causes of the performance degradation can be considered during this step, such as a change in appearance or occlusion, but we do not include any relevant schemes such as template update. To address the abrupt motion of features, we progressively expand the search region until we find the correct solution. This approach ensures an optimal search region size, although the motion of features is abrupt, e.g., a feature moves 20 pixels during one time step. However, deciding when to stop remains a critical issue.

To expand the search region, we use a hexagonal lattice centered on the predicted position. Initially, there is a single hexagon, as shown in Fig. 2 (left). If the predicted distance is greater than d_{max}, we increase the degree of the honeycomb by one, as shown in Fig. 2 (right). The distance between hexagons is determined by d_{max} because the hexagonal lattice guarantees equidistance among neighboring hexagons. Thus, the use of a hexagonal lattice increases the effectiveness of search compared with a rectangular lattice. Each hexagon in the honeycomb is indexed as follows:

$$\mathbf{x}^{(l,k)} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}, \quad n = 6l, \tag{6}$$

[3] A better prediction model can increase tracking performance; however, this paper mainly focuses on the SDT.

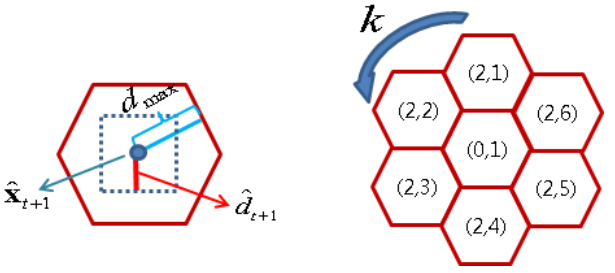


Figure 2: Illustration of a honeycomb structure and its corresponding indices.

where **x** denotes the position of a feature in the image coordinates, and the superscripts l and k indicate the level and the number of a hexagon, where l starts from zero and k begins with one for convenience. The level is indexed from the center while the number is indexed from the top in a counter-clockwise manner. The number of hexagons at a specific level is proportional to the level, i.e., six times the level. While increasing the size of the search region, we use the SDT to determine the size of the search region and we reject inappropriate candidate regions by comparing the SDT value to d_{max}. Thus, the actual number of computations is reduced significantly by using the SDT, although the size of the search region increases exponentially.

We use two criteria to terminate search and tracking. In conventional approaches, a threshold value (based on the similarity value) is used to decide whether we accept the current estimate as the solution. A common choice for this threshold is 0.8 for NCC, although a single constant value cannot guarantee a correct decision. Any cross-coefficient value above this threshold may be a solution, especially if the feature prediction is not accurate but it still yields high cross-correlation. To overcome this problem, we define a second termination criterion by assuming small differences in the correlation coefficients as follows:

$$\hat{c}_t - \hat{c}_{t+1} < \delta_S, \tag{7}$$

where $\hat{c}_t$ indicates the NCC coefficient with the best match at time t while δ_S is the threshold value of the difference in the correlation coefficients.

3 Experimental Evaluation

We conducted two experiments, where the first evaluated the SDT and the second evaluated the performance of feature tracking in terms of the computational efficiency and robustness of tracking. In this experiment, we set the parameters as follows: NCC termination threshold $\delta_{NCC} = 0.8$; δ_D ranged from 0.04 to 0.32 with an interval of 0.04; $\delta_S = 0.05$; a template size of 15-by-15; the fixed search regions were 7-by-7, 15-by-15, 21-by-21, 31-by-31, and 51-by-51; and the maximum layer of a honeycomb structure was 5. We captured four video sequences using a hand-held camera while walking, running, and jumping in various environments. We manually extracted the ground truth feature points and refined them using local window searches in a 5-by-5 neighboring region.

(a) Walking 1 (b) Walking 2 (c) Running (d) Jumping

Figure 3: First frames of test video sequences. We extracted 50 features from each sequence in the first frames using a corner detector [13]. The average displacements of features were (a) 3.4 px, (b) 4.3 px, (c) 7.2 px, and (d) 9.7 px. The maximum displacements were (a) 21.0 px, (b) 44.7 px, (c) 53.8 px, and (d) 67.74 px. The standard deviations of the displacements were (a) 2.3 px, (b) 4.7 px, (c) 5.1 px, and (d) 7.5 px. Each sequence contained 400 frames, except the running sequence (500 frames).

3.1 SDT Evaluation

We evaluated the accuracy of the SDT in terms of prediction errors and true positive ratios. We computed the prediction error by calculating the expected distance and the ground truth displacement of a feature as follows:

$$\begin{aligned} &\mathrm{e}_t = |\hat{d}_{t+1} - d^*_{t+1}| \text{ s.t.} \\ &\hat{d}_{t+1} = SDT(C_{t+1}(\mathbf{x}^*_t)) \text{ and } d^*_{t+1} = ||\mathbf{x}^*_{t+1} - \mathbf{x}^*_t||_2, \end{aligned} \tag{8}$$

where the accuracy of the SDT was based on the difference between the predicted distance and the ground truth displacement. $\mathbf{x}^*_t$ indicates the ground truth position of a feature at time t and $\hat{d}_{t+1}$ is the expected distance at time t. $C_{t+1}(\mathbf{x}^*_t)$ indicates the NCC coefficient of a template in position $\mathbf{x}^*_t$ at time $t+1$. The average prediction errors of the SDT are shown in Tab. 1 and these errors occur mainly due to the abrupt motion of a feature which draws several peaks as well as large prediction errors as shown in Fig. 4, which also clearly suggests that the proposed SDT approximates the actual displacement of features during egomotion.

We also searched for the best match within the expected distance using the TSS algorithm [5]. If the correct solution was found, we counted this as a true positive, whereas they were counted as false positives if this was not the case. Without constraining the valid range of the SDT, true positive ratios were greater than 90% for the walking sequences and greater than 70% for the jumping and running sequences, which contained numerous abrupt feature motions. These ratios are computed using all the ground truth points over the entire sequence (the ground truth track of a feature is less than the sequence length) and show that the SDT itself ensures sufficiently narrow search regions in many cases. Approximately 10-30% percentage of the false positive cases significantly degrade tracking performance, however, and these cases greatly depend on the selection of the control parameter δ_D, which determines the valid range of the SDT. Therefore, we evaluate the effect of δ_D while varying the value of δ_D, as shown in Fig. 5. As we restricted the value of δ_D, which increased the frequency of progressive hexagonal expansion steps, the valid range of SDT decreased and the false positive ratio was closes to 0%. There is a tradeoff between the computational complexity and reliability; smaller δ_D decreased the false positive ratios and vice versa. Therefore, this value should be determined to suit specific purposes; we can use a large value if rapid computation is more important, whereas a smaller value can reduce the number of false positives.

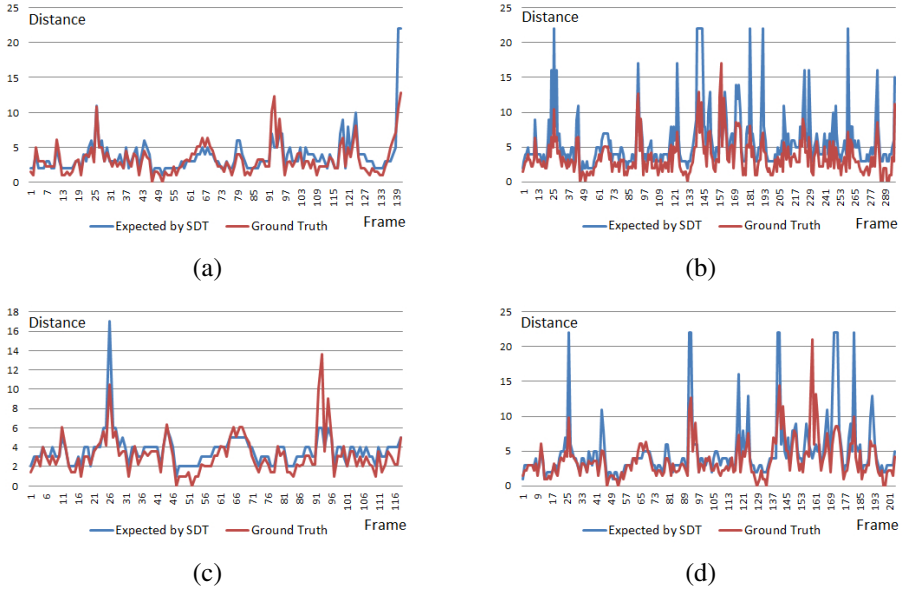


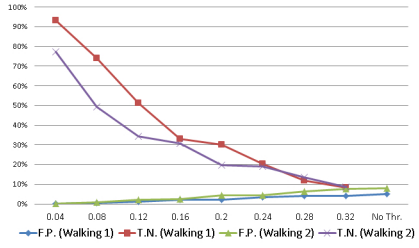
Figure 4: Comparison of the expected distances (blue lines) and the ground truth distances (red lines) for features in the walking 1 sequence.

Table 1: Evaluation of the SDT in terms of accuracy (average error) and correctness (true positive ratio). True positive ratios are computed as the ratio of correctly estimated cases to the number of entire ground truth points.

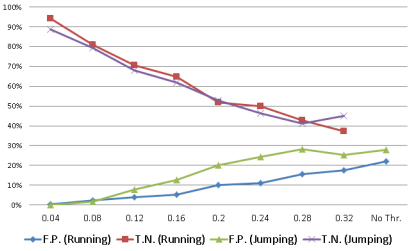
	Walking 1	Walking 2	Running	Jumping
Avg. Error	1.742	1.7846	3.044	4.504
T.P. Ratio	95%	92%	78%	72%

3.2 Feature Tracking Evaluation

To evaluate the features tracked using the SDT, we compared the lifetime of features and the time complexity of template matching. The lifetime of a feature is the length of its track; if the feature is correctly tracked then a longer track is preferable than shorter ones. Thus, we compared the average lifetimes of features to evaluate the robustness of the method against abrupt feature motions. Generally, the search of small region is computationally efficient, but it improves the missing probability of tracking dut to faster motion of feature than the size of search region. With our data sequences, the proposed method produced competitive results in terms of the robustness of tracking against abrupt motions, as shown in Fig. 6 (b). However, the proposed method did not significantly increase the number of searching operations as opposed to its robustness of tracking, as shown in Fig. 6 (a). Compared to conventional template matching, which uses fixed sizes of search regions, the time complexity of the proposed method was in between searching 3-by-3 region and 11-by-11 region while showing better results than searching 31-by-31 region; mainly, this improvement is because of the adaptively changed size of search regions. The time complexity tended to increase with the proportion of abrupt feature motions.



(a) Walking 1 and Walking 2 Sequences



(b) Running and Jumping Sequences

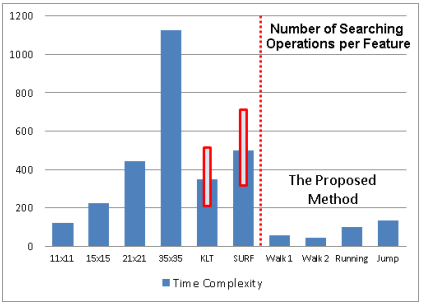
Figure 5: Comparison of the false positive ratios and true negative ratios using different values for δ_D.

In addition, we evaluated the performance of feature tracking with KLT-[1], SURF-[2], and NCC-based template matching in terms of time complexity, actual running time, and lifetime. We used the KLT code from the web site of S. Birchfield [3] and the OpenCV 2.4[4] versions for SURF and NCC matching. With our data sequences, the tracking of 100 features using the proposed method required about 3–10 ms, depending on the motion of features. This tracking time can be boosted by using a simpler similarity measure or by adopting fast template matching techniques. With KLT, the tracking time was about 40 ms, while the OpenCV implementation was about 30% faster. The tracking time with SURF is about 60–70 ms, which included the feature detection time. The tracking time increased linearly with NCC matching, depending on the searching operations of template matching, as shown in Fig. 6. Thus, the search of a 9-by-9 region required about 7 ms, while search of a 21-by-21 region required about 37 ms, when using the template-matching function in OpenCV. We can see the relative time complexity of KLT and SURF in Fig. 6 (a) based on actual running time; two red bars indicate that this comparison can be different depending on implementations and the total number of features to track. We also compared the lifetime of features tracked by KLT to the proposed method; the KLT showed robust and accurate results in a smoothely moving sequence such as the walking 1 sequence, however, the presence of abrupt motions significanlty degraded the tracking performance of KLT, as shown in Fig. 6 (b).

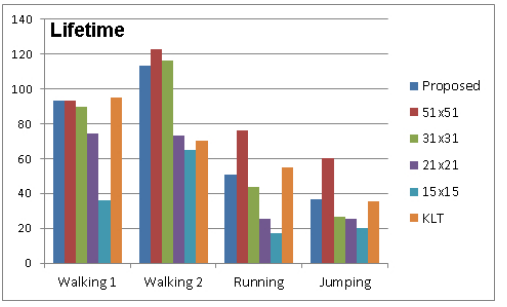
4 Conclusions

In this paper, we presented a Self-Aware Distance Transform (SDT) for robust and fast feature tracking based on template matching. The SDT is computed during the autocorrelation step and it determines the size of the search region at every step. With the SDT, we can use a small search region when the feature motion is small, otherwise we can enlarge the search region size as the motion of a feature increases. To address the abrupt motion of features, we proposed the progressive expansion of the search region using a hexagonal lattice with two termination criteria. We experimentally verified that the proposed method is computationally efficient and that it maintains robustness during tracking.

[4] http://opencv.org



(a) Time complexity



(b) Average lifetime of features

Figure 6: Comparison of the time complexity in terms of the number of search operations (a) and the lifetime of features with various search region sizes (b).

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2009-0065038).

References

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Pittsburgh, PA, July 2002.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[3] Stan Birchfiled. Klt: An implementation of the kanade-lucase-tomasi feature tracker, http://www.ces.clemson.edu/ stb/klt/.

[4] M. Donoser and H. Bischof. Efficient maximally stable extremal region (mser) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 553 – 560, june 2006.

[5] Yu-Wen Huang, Ching-Yeh Chen, Chen-Han Tsai, Chun-Fu Shen, and Liang-Gee Chen. Survey on block matching motion estimation algorithms and architectures with new results. *J. VLSI Signal Process. Syst.*, 42(3):297–320, March 2006.

[6] Myung Hwangbo, Jun-Sik Kim, and T. Kanade. Inertial-aided klt feature tracking for a moving camera. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1909 –1916, oct. 2009.

[7] J. P. Lewis. Fast normalized cross-correlation, 1995.

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, November 2004. ISSN 0920-5691.

[9] A. Mahmood and S. Khan. Exploiting transitivity of correlation for fast template matching. *Image Processing, IEEE Transactions on*, 19(8):2190 –2200, Aug. 2010.

[10] J. M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison, 2009.

[11] Duy nguyen Ta, Wei chao Chen, Natasha Gelfand, and Kari Pulli. Surftrac: Efficient tracking and continuous object recognition using local feature descriptors. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[12] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 723 –730, june 2010.

[13] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition (CVPR)*, Ithaca, NY, USA, 1993. Cornell University.

[14] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[15] S. Sun, D. Haynor H. Park, and Y. Kim. Fast template matching using correlation-based adaptive predictive search. In *International Journal of Imaging Systems and Technology*, volume 13, pages 169–178, 2003.

[16] Steven L Tanimoto. Template matching in pyramids. *Computer Graphics and Image Processing*, 16(4):356 – 369, 1981.

[17] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[18] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *Visualization and Computer Graphics, IEEE Transactions on*, 16(3):355 –368, may-june 2010.

[19] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.