Real-time Tracking with Selective DoP-RIEF Features for Augmented Reality

Yi Zhang
[1]SECE of Shenzhen Graduate School, Peking University
Shenzhen, China
[2]The Institute of Digital Media, School of EECS, Peking University
Beijing, China
zhangyi_0220@163.com

Ping Lu
[1]School of Information Science and Engineering, Southeast University,
Nanjing, China
[2]ZTE Corporation
Shenzhen, China
lu.ping@zte.com.cn

Jie Chen, Ling-Yu Duan
The Institute of Digital Media, School of EECS, Peking University
Beijing, China
{cjie, lingyu}@pku.edu.cn

Abstract—Real-time, accurate and robust target tracking on mobile devices is an important problem which can facilitate applications such as augmented reality. However, it is still unsolved, partly due to the mobile's computing limitations. Compressive tracker performs favorably against state-of-the-art algorithms in terms of efficiency, accuracy and robustness, but as limited by the speed of feature matching, it cannot achieve real-time tracking in mobile applications. In this paper, we propose a fast feature, i.e., selective Difference of Patch Robust Independent Elementary Features (DoP-RIEF). DoP-RIEF is a global feature which is related to BRIEF. It uses histogram to fit feature distribution because it is more flexible than Gaussian, and intermediate results for subsequent classification can be stored, avoiding duplication of operations. Feature selection further deletes features which are less discriminative and improves the feature quality. Through these two steps, the feature matching can be accelerated significantly and at the same time tracking accuracy and robustness are improved. Compared with compressive tracker on 17 publicly available sequences, our method outperforms it in terms of both robustness and accuracy. In addition, the speed is about 270 frames per second which is 8 times faster than the compressive tracker. To further evaluate our algorithm in natural scenes with obvious scale, rotation, and illumination variations, we test it on Stanford datasets and Peking University landmark datasets, and the accuracy is above 90%.

Keywords-real-time tracking; augmented reality; fast global feature; feature distribution fitting; feature selection

I. INTRODUCTION

Augmented Reality (AR) is an emerging technology to render and superimpose virtual information onto the real scene. It enhances the user experience in terms of visual and auditory sensations, and deepens understanding of real environment. Currently it has already been applied to education, military aviation, historic restoration, technical training, games and many other fields.

Tracking is a critical module in AR. Tracking speed, accuracy, and robustness directly impact on the quality of rendering and display, and larger tracking errors may lead to displayed errors. The main issues of tracking technology on mobile devices include:

a) Real-time: Augmented reality system requires virtual information to be displayed without any delay. Real-time directly restricts the availability of an augmented reality system;

b) Accuracy: Augmented reality systems needs tracking module to provide precise location of the target without jitters. Accuracy is a prerequisite for the correct fusion of the virtual information and real scene;

c) Robustness: Tracking module should handle occlusion, blur, and illumination, scale, rotation variations in complex natural scenes.

However, target tracking which restricts the availability of an AR system is still under developing and leaves many unsolved technical issues. The most important reason is the mobile's computing limitations. Moreover, there exist obvious scale, rotation, and illumination variations in natural scenes. In this work, we focus on the tracking module, aiming to improve the tracing accuracy and robustness with less computation time.

In classic computer-vision-based tracking methods, the target appearance model is learned first. Then the location (Kalman filter [1, 12], particle filter [2, 3], and so on) of the target area in the subsequent frames is estimated by matching with the learned model. The methods of describing the target appearance can be categorized as local feature description, global feature description, and hybrid (local and global) feature description.

In the methods based on local feature description, a set of local features are extracted from two adjacent frames firstly. Then, the pairs of matching features are found to determine the location of the target in next frame. SIFT [13] and SURF [15] are widely used because of the matching stability, and they are invariant with respect to image transformation and illumination. However, the speed of feature extraction and matching is not high enough, especially in mobile augmented reality or other real-time applications where tracking speed is a key issue. Therefore, other faster features would be adopted such as

978-1-4799-8688-0/15 $31.00 © 2015 IEEE
DOI 10.1109/BigMM.2015.30

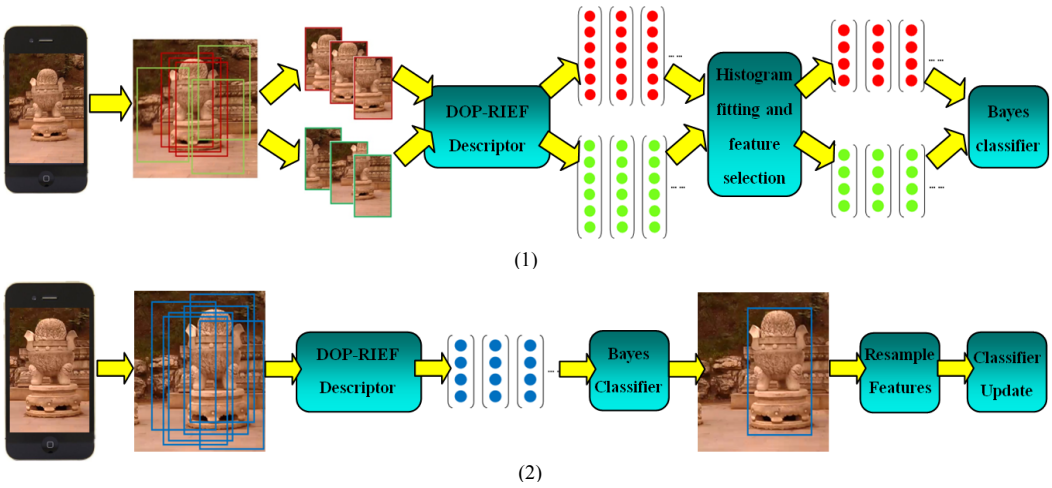


Fig.1. framework of our algorithm. (1) initial part of the tracker in the first frame. (2) tracking process for subsequent frames

FAST [16], BRISK [17], ORB [9] and so on. Wagner [14] modified SIFT and Fern features to make it suitable for fast tracking. However there is a drawback with the tracking methods based on these local features. That is, when the target or camera is moving fast in the scene, images will blur significantly, then in these situations local features may not be efficiently extracted, thereby affecting subsequent tracking.

Methods based on global statistical features tend to consider the targets as a whole and extract the overall statistical feature descriptors to represent the target appearance model. To some extent, these methods do not depend on local details, thus having better robustness when the target is blurred or partially occluded. In Comaniciu's work [1], histogram-based target representations were used and with kernel regularization and mean-shift algorithm, they got a fast and robust tracker. In Bradski's work [7], a computer vision color tracking algorithm called CAMSHIFT was developed and applied to tracking human faces. In Zhang's work [10], they proposed a compressive tracker, with a global appearance model based on features extracted from the multi-scale image feature space with data-independent basis. It runs in real-time on PC and performs favorably against state-of-the-art algorithms. However, as limited by the speed of feature matching, it cannot achieve real-time tracking in mobile applications.

Some methods extract both local and global features which are shown to improve the tracking accuracy. In Zhong's work [8], they proposed a robust appearance model that exploited both holistic templates and local representations. holistic templates are incorporated to construct a discriminative classifier that can effectively deal with cluttered and complex background. Local representations are adopted to form a robust histogram that considers the spatial information among local patches with an occlusion handling module. However, under certain constraints, such as mobile applications, these methods can not meet the real-time requirement, although the accuracy and robustness are satisfactory.

According to the above analysis about the three categories of methods on speed, accuracy and robustness, we focus on methods based on global features. In this paper, we propose a fast global feature: selective DoP-RIEF (Difference of Patch Robust Independent Elementary Features). This feature is a modified version of BRIEF (Binary Robust Independent Elementary Features) [6] and designed to describe the target with a global representation model. BRIEF is a local feature descriptor first proposed by Michael Calonder, and is very fast both to build and to match. The advantage of BRIEF is its high discrimination even when using relatively few bits, and it can be computed by simple intensity difference tests.

In our work, we take the advantages of BRIEF in speed and accuracy, and extend it from the pixel-level to block-level, modifying it to be a global feature which can describe the whole target. It uses histogram to fit feature distribution because it is more flexible than Gaussian, and intermediate results for subsequent classification can be stored, avoiding duplication of operations. Feature selection further deletes less discriminative features and improves the feature quality. Experiment results show that through histogram feature distribution fitting and feature selection, feature matching is accelerated significantly, and at the same time tracking accuracy and robustness are improved.

We use the system framework of compressive tracker and compare them in terms of speed, accuracy and robustness. Experiment results show that with selective DoP-RIEF, our algorithm is more robust than compressive tracker. The speed is about 270 frames per second on the PC with Dual-Core 2.93GHz CPU which is about 8 times faster, and the result is more accurate. To further evaluate our algorithm in natural scenes with obvious scale, rotation, and illumination variations, we test it on Stanford datasets and Peking University landmark datasets, and the accuracy is above 90%. It performs favorably against state-of-the-art algorithms and is more suitable for tracking on mobile devices for augmented reality.

The rest of the paper is organized as follows: Section 2 gives the problem statement. Section 3 describes the details of our selective DoP-RIEF feature. Section 4 presents experiment results on some challenging sequences and the compared

results. We also discuss the extension of our algorithm to scale and rotation variations. Section 5 concludes the paper.

II. PROBLEM STATEMENT

As motivated in the previous section, we design our algorithm based on global features, and use the framework of compressive tracker for the tracking module. Fig. 1 shows the algorithm framework. It consists of two parts. The first part is tracking initialization, and the second one is tracking and updating.

In the first part, the exact location of the target in the first frame is given, and the location is represented by a rectangular box. A set of positive samples are selected at the closest distance to the target area, and a set of negative samples are selected far away from the target area. Then the features of all the samples are extracted, where each feature corresponds to an N-dimensional vector. Next, the most discriminative n dimensional features are selected from N-dimensional features. Finally a Bayes classifier is trained with the selected features of positive and negative samples.

In the second part, when looking for the target in the next frame, the first step is to predict the initial search location of the target, then take samples around the location, and extract the feature descriptors for each sample. With the extracted descriptors, the Bayes classifier is used to calculate all the response values. The one getting the maximum response value is selected as the new target location. Finally, new positive and negative samples are extracted according to the new target location and the classifier is updated with the new samples.

assuming all elements in a feature vector are independently distributed, and using a Bayes classifier, we can calculate the response value $res(v)$ for a given feature v with equation (1),

$$res(v) = \log\left(\frac{\prod_{i=1}^{n} p(v_i \mid y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i \mid y=0)p(y=0)}\right) \quad (1)$$

$y \in \{0, 1\}$ is a binary variable which represents the sample label. To simplify this equation, we also assume that the priori probabilities are equal, that is $p(y = 1) = p(y = 0)$. So equation (2) is finally used to calculate the classifier response.

$$res(v) = \sum_{i=1}^{n} \log\left(\frac{p(v_i \mid y=1)}{p(v_i \mid y=0)}\right) \quad (2)$$

According to the above process description, the speed and accuracy of feature matching, to a great extent, determine the classification performance. The classifier calculates response value res for each sample as the probability of containing the target, therefore, the dimensions of the feature and computational complexity of the probability will directly affect the speed of tracking. Meanwhile the accuracy of feature distribution fitting will directly affect the accuracy of the tracking algorithm. Based on this analysis, we focus on optimization of the feature quality, and propose a fast global feature, i.e. selective DoP-RIEF. Through feature distribution fitting and feature selection, the classification speed and accuracy are greatly increased, and then the performance of the tracking algorithm gets improved. The details of our selective DoP-RIEF will be given in the next section.

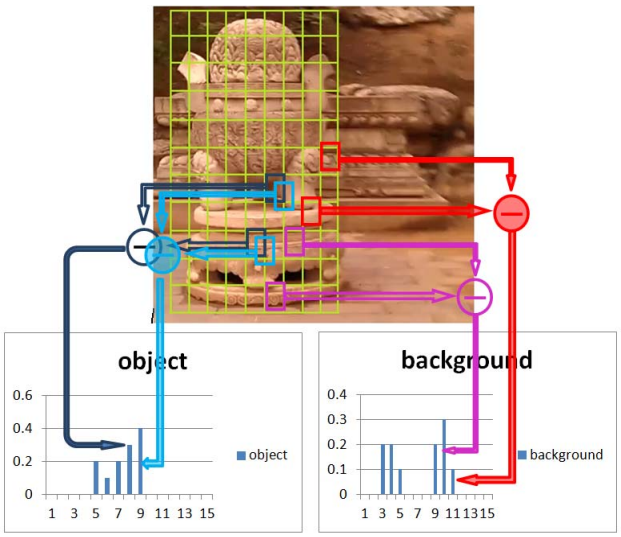


Fig.2. fitting process for each element of feature vector. Positive samples(object) use pairs near the target location, and negative samples(background) use pairs far away from the target location.

III. SELECTIVE DOP-RIEF

In this section, we will first give a brief review of BRIEF and then describe the details of DoP-RIEF in tracking process including feature extraction, feature distribution fitting, feature selection, classification, and updating.

A. Brief Review of BRIEF

BRIEF is first proposed by Michael Calonder, and is very fast both to build and to match. The advantage of BRIEF is its high discrimination even when using relatively few bits, and it can be computed using simple intensity difference tests. Test τ on patch p can be defined as

$$\tau(p; x_i, y_i) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{other} \end{cases} \quad (3)$$

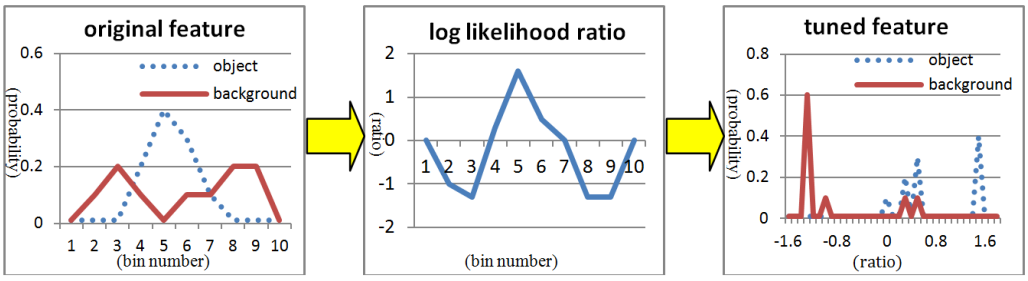
where $p(x)$ is the pixel intensity in a smoothed version of p at $x = (u, v)$ where u and v are the horizontal and vertical coordinates of x. Choosing a set of $n_d(x, y)$-location pairs uniquely defines a set of binary tests. And the BRIEF descriptor is the n_d-dimensional bit-string

$$f_{n_d}(p) = \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_i) \quad (4)$$

Generating a length n_d bit vector leaves many options for selecting the n_d test locations (x_i, y_i) in a patch of size S × S. Experiment result shows that Gaussian distribution $(0, \frac{1}{25}S^2)$ outperforms other sampling geometries.

B. Feature Extraction

In order to describe the target with global representation and take the advantages of BRIEF in speed and accuracy, our DoP-RIEF feature extends the pixel-level operations of BRIEF



(a) (b) (c)

Fig. 3. transform result of each step for feature selection. (a) the original feature histogram for object and background. (b) the log likelihood ratio calculated with the original histogram. (c) probability distribution of object and background with the tuned feature.

to the block-level operations. The target area is uniformly divided into R rows and C columns, where the value of each patch is the sum of all the pixel values in it. With integral image, the value of each patch can be efficiently calculated. N pairs of patches are selected and the difference of each pair is calculated as the value of the feature. The N-dimension vector forms the descriptor of the target. In most cases, the targets are in the center of the bounding boxes, so the Gaussian distribution is considered as the most suitable sampling geometry. We use the same parameters (μ, σ) as the ones of BRIEF and set X, Y to be independent, that is X~Gaussian$(0, \frac{1}{25}C^2)$, Y~Gaussian$(0, \frac{1}{25}R^2)$. The features are not binarized in order to retain enough information for feature distribution fitting in the next step.

C. Feature Distribution Fitting

The fitting accuracy of feature distributions for positive and negative samples directly affects the accuracy of the Bayes classifier. Compressive tracker assumes that the distribution of each feature is Gaussian. Therefore, it calculates the mean and variance of each feature according to the statistical result, and updates the mean and variance in the tracking process. But experiment results show that such assumption has two disadvantages:

a. In many scenes, feature distributions do not satisfy the Gaussian assumption, and may even have multiple peaks (Fig.3 (a)). So it is not always accurate to fit the distributions with Gaussian.

b. When the classification is calculated with a Gaussian function to get the probability, it would involve a large number of exponential and logarithmic arithmetic operations. Since the response value for each sample needs to be calculated, the computational complexity will increase greatly. For some applications that are limited by the computing resources, such as augmented reality on mobile devices, these operations would become a bottleneck.

Our DoP-RIEF uses histograms to fit the distributions of the features, and builds a look-up table for each logarithm item, avoiding the above two disadvantages. The algorithm is illustrated as follows. Suppose the location of the target is initialized in the first frame, and the center coordinate of the target is (x_0, y_0). Then a set of positive samples are selected near the center, with the center coordinates in the range (x0 + Δx_0, $y_0 + \Delta y_0$), where $|\Delta x_0| < tx_0$, and $|\Delta y_0| < ty_0$. The negative samples is far away from the target center, with the center coordinates in the range $(x_0 + \Delta x_1, y_0 + \Delta y_1)$, where $tx_1 < |\Delta x_1| < tx_2$, $ty_1 < |\Delta y_1| < ty_2$, $tx_1 > tx_0$, $ty_1 > ty_0$.

Since the elements in a feature vector are independent of each other, each distribution can be fitted without referring to others. After the division of the target into R rows and C columns, the size of each patch is set to be pw height and pc width in pixels. Each histogram contains h bins. For grayscale images, the difference for each pair of patches is in the range [$-pw * pc * 256$, $pw * pc * 256$]. And these values can be discretized into the bins of the histogram. For the value v, the num of bin it belongs to is calculated as follows.

$$binLength = \frac{pw * pc * 2 * 256}{h} \tag{5}$$

$$binNum = \frac{v}{binLength} + \frac{h}{2} \tag{6}$$

In equation (5), *binLength* is the range covered by each bin. In order to avoid negative values, the result is increased by half of h in equation (6) to adjust the *binNum* to a non-negative value.

In order to fit the distribution of each element in the feature vector, the feature values for positive and negative samples are accumulated separately with histograms. Finally, the histograms are normalized so that the sum of all the bins is 1. These histograms are more accurate and suitable to fit the distributions. The fitting process is shown in Fig.2.

D. Feature Selection

Another advantage of using histogram for feature distribution fitting is that it is convenient for feature selection. Features with higher degree of discrimination can be selected to get higher tracking accuracy. With feature selection, the algorithm can adapt to different applications which require different speed. That is, on mobile devices less features would be used, while in PC applications more features would be used.

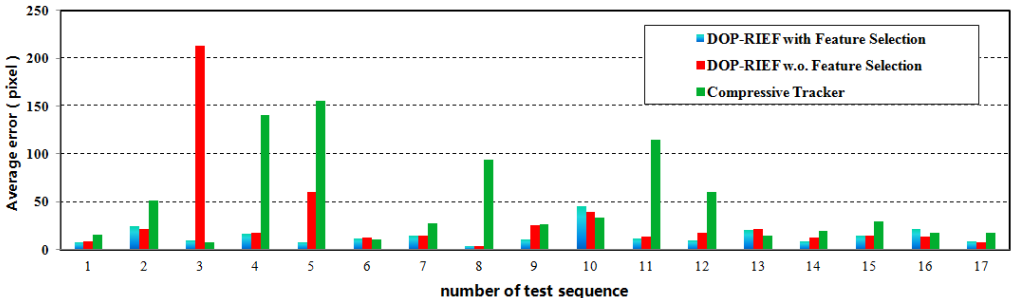


Fig. 4. Average error for each sequence. Y axis is average error, X axis is the sequence num: 1.sylv, 2.board, 3.bolt, 4.lemming, 5.skating, 6.football, 7.singer, 8.car, 9.faceooc2, 10.girl, 11.animal, 12.shaking, 13.twinning, 14.david, 15.faceocc, 16.coke11, 17.dollar. For each sequence, we compare compress tracker(green), our algorithm without feature selection(red) and our algorithm with feature selection(blue).

With the histogram of the positive and negative samples, we can refer to Robert's work [11] for feature selection. In order to compute the discrimination for each element of the feature vector, the histograms are transformed first. The transformation is computed as a log likelihood ratio of the feature value distributions for object versus background. This tuned feature is formed as the log likelihood ratio of the class conditional feature distributions. The log likelihood ratio of for each bin i is calculated as equation (7).

$$ratio(i) = \log\left(\frac{obj(i)}{bkg(i)}\right) \tag{7}$$

$obj(i)$ is the value of the ith bin in the object histogram, and $bkg(i)$ is the value of the ith bin in the background histogram. In order to prevent the error of division by zero, $obj(i)$ is set to be $\max\{obj(i), \delta\}$, and $bkg(i)$ is set to be $\max\{bkg(i), \delta\}$, where δ is a small value.

In order to measure the discrimination that tuned feature induces between object and background classes, the two-class variance ratio can be calculated. We could proceed by reaccumulating new class conditional distributions for the tuned feature and then calculate the variance with the new distributions. But for efficiency, the distributions $obj(i)$ and $bkg(i)$ that already computed for the features can be used. Through equation (8)

$$\mathrm{var}(x) = E(x^2) - (Ex)^2 \tag{8}$$

the variance of the tuned feature for the object with respect to object class distribution can be computed as

$$\mathrm{var}(obj) = \sum_i obj(i) * ratio^2(i) - \left[\sum_i obj(i) * ratio(i)\right]^2 \tag{9}$$

For background, the variance is

$$\mathrm{var}(bkg) = \sum_i bkg(i) * ratio^2(i) - \left[\sum_i bkg(i) * ratio(i)\right]^2 \tag{10}$$

Variance of the feature over both object and background is calculated by equation (11) and (12).

$$all(i) = \frac{obj(i) + bkg(i)}{2} \tag{11}$$

$$\mathrm{var}(all) = \sum_i all(i) * ratio^2(i) - \left[\sum_i all(i) * ratio(i)\right]^2 \tag{12}$$

Finally, the discrimination for the feature can be defined as

$$dis = \frac{\mathrm{var}(all)}{\mathrm{var}(obj) + \mathrm{var}(bkg)} \tag{13}$$

In this way, features with larger *dis* can be selected with higher priority for subsequent classification. Fig.3 shows the transformation result of each step for feature selection.

E. Classification and Updating

Since the histograms for positive and negative samples have been generated, the logarithm item in equation (2) can be calculated easily for each v_i. The advantage of using histogram is that these values need to be calculated only once and can be stored in a look-up table. Then, in the classification stage, the classifier only needs to calculate the bin to which a given feature belongs, with equation (6), and looks up the table for the logarithm value. In this way, the classification calculations are accelerated significantly.

As the target moves in the scene, the appearance and background will gradually change. Therefore, the feature histograms should be incrementally updated in the tracking process. The updating method is illustrated as follows:

Suppose that after tracking the object for i frames, the histogram of the object is (o_1, o_2, … … , o_n), and the histogram of the background is (b_1, b_2, … … , b_n). In the (i+1)th frame, the histogram are (o_1', o_2', … … , o_n') and (b_1', b_2', … … , b_n') for object and background respectively. Then the histograms are incrementally updated to (o_1 * *rate1* + o_1'* (1 – *rate1*), o_2 * *rate1* + o_2' * (1-*rate1*), ... …, o_n * *rate1* + o_n'* (1- *rate1*)) and (b_1 * *rate2* + b_1'* (1 – *rate2*), b_2 * *rate2* + b_2' * (1-*rate2*), ... …, b_n * *rate2* + b_n'* (1- *rate2*)) where *rate1* and *rate2* are the learning parameters and their values are between 0 and 1. Different learning rates can be set for object and background respectively.

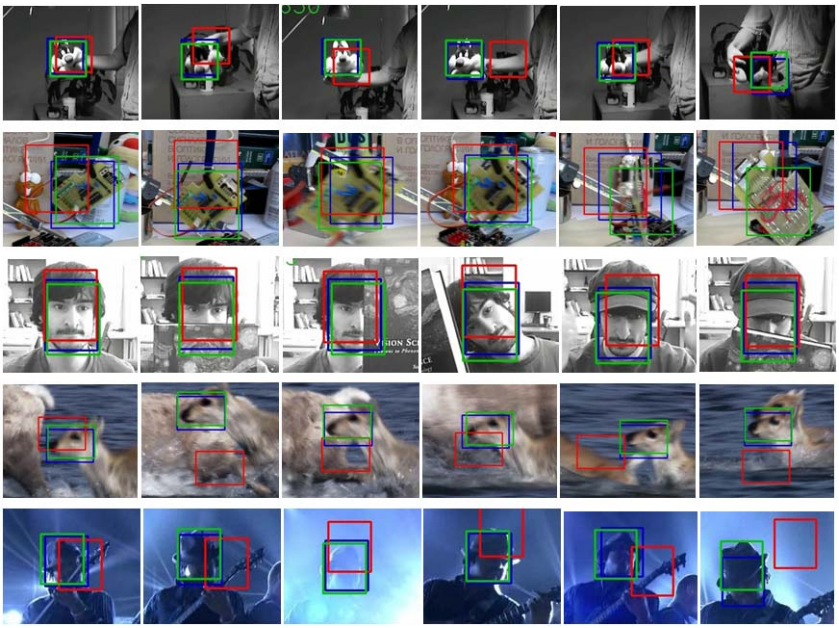


Fig. 5. Tracking results compared with compressive tracker. Green boxes are the ground truth, blue boxes are the results of our algorithm, and the red boxes are the results of compressive tracker. Each row for a sequence. Row 1: sylv. Row 2: board. Row 3: faceocc2. Row 4: animal. Row 5: shaking.

IV. EXPERIMENTS

At the beginning of this section, we compare our algorithm with compressive tracker on 17 challenging sequences(sylv, board, bolt, lemming, skating, football, singer, car, faceooc2, girl, animal, shaking, twinning, david, faceocc, coke11, dollar). In Zhang's work, they have already compared compressive tracker with fragment tracker [4], the online AdaBoost method [5], the Semi-supervised tracker [18], the MILTrack algorithm [19], the l_1-tracker [20], the TLD tracker [21], and the Struck method [22]. Compressive tracker outperforms these trackers. Therefore, in this paper, we only compare our algorithm with compressive tracker. For fair comparison, we use the source code provided by the author with tuned parameters for best performance. Our algorithm and compressive tracker are both complemented in C++ and run on a Dual-Core 2.93GHz CPU with 3.5 GB RAM. In order to evaluate the ability to handle situations of scale and rotation variations, we also extend our algorithm and test it on Stanford datasets(18 sequences) and Peking University landmark datasets(20 sequences).

A. Experimental Setup

In the initial step, the bounding box of the target is divided into R=30 rows and C=30 columns uniformly. The initial feature length N is set to 200. For feature distribution fitting, the number of bins in a histogram is set to h=30. For feature selection, the threshold for *dis* is set to 1.5, which means that only the elements with *dis* larger than 1.5 will be used for classification. The learning rate *rate1* and *rate2* are set to 0.85. For fair comparison, we use the same sampling parameters for positive and negative samples as the ones of compressive tracker.

B. Accuracy Analysis

We use the center location error to evaluate the two algorithms. This metric can be measured with manually labeled ground truth data. We compute the distance between the center coordinate of the ground truth and our algorithm for each frame, and then get the average error for the whole sequence. We also evaluate the accuracy of our algorithm without feature selection. Fig.4 shows the quantitative results, and Fig.5 shows screenshots of some tracking results.

Our tracker gets higher accuracy in most of the sequences than compressive tracker. The average error of compressive tracker is 49.05 pixels, while the average error of our algorithm is 14.35 pixels. Both compressive tracker and our algorithm use the same Bayes classifier to calculate the response value as the probability of containing the target. Therefore, the accuracy of the tracker mainly depends on the accuracy of the feature distributions and the discrimination of the features. Through our selective DoP-RIEF, the feature distributions are fitted with histograms which are more flexible than Gaussian and can be used to estimate much more types of distributions. On the other hand, the quality of the features gets further improved through feature selection. In the process of feature selection, only high-quality features are retained, because their discrimination between target and background are higher than a predefined threshold. In this way, the noise interference is reduced.

In the sequences of lemming, skating and car, the average errors of compressive tracker are very large because the tracker loses the targets. In the sequence of bolt, our algorithm without feature selection also loses the target. Our algorithm with

feature selection gets stable performance and the average errors are relatively small.

However, DOP-RIEF with feature selection perform worse than that without feature selection in a few sequences. The reason is that with feature selection, the features with low discrimination have been deleted, which may not be totally useless for tracking. In general, selective DoP-RIEF provides the Bayes classifier with higher quality features.

C. *Efficiency Analysis*

In order to accurately determine the target location, a sufficient number of samples need to be obtained in each frame. In our experiment, the average search radius is 20 pixels and we will get more than 1000 samples. Therefore, the computational complexity of calculating the response values will be the bottleneck of tracking efficiency. Since histograms are used to store logarithm items for classification in a look-up table. Then, in classification stage, the classifier only needs to calculate the bin to which a given feature belongs, and looks up the table for the logarithm value. In this way, the classification calculations are accelerated significantly. The average speed of our algorithm without feature selection is 0.0045 seconds per frame. With feature selection, as the features with low discrimination are deleted, the average speed is increased to 0.0037 seconds per frame, while the average speed of compressive tracker is 0.031 seconds per frame. So our algorithm is about 8 times faster than compressive tracker. Fig. 6 shows the comparison result. With this speed, our algorithm may track 270 frames per second on a PC with Dual-Core 2.93GHz CPU and therefore it is more suitable for mobile applications.

D. *Scale and Rotation Variations*

Scale and rotation variations of the targets exist in most of augmented reality applications, so the tracker should be able to track the targets and simultaneously handle scale and rotation variations. On the basis of the framework of our algorithm, we expand the search range. In addition to searching the target with a box as large as the one in the previous frame, 4 more extra boxes are used: the first one with larger size, the second one with smaller size, the third one with a left-hand rotation, the fourth one with a right-hand rotation. All the response values are calculated, and the box with the maximum response value is chosen as the location area of the target. The experiment results show that the additional 4 boxes are enough for handling scale and rotation variations. In our experiments, the size change is one tenth of the original box and the angle of rotation changes is 3 degrees.

We test our algorithm on the Stanford datasets(18 sequences) and Peking University landmark datasets(20 sequences). Both datasets involve scale, rotation and illumination variations. In terms of accuracy evaluation, we use the $score = \frac{area(ROI_T \cap ROI_G)}{area(ROI_T \cup ROI_G)}$, where ROI_T is the tracking bounding box and ROI_G is the ground truth bounding box. If the *score* is larger than 0.75 in one frame, the tracking result is considered as a success. And the results show that the accuracy of our algorithm is above 90%. Fig.7 and Fig.8 show screenshots of some tracking results.

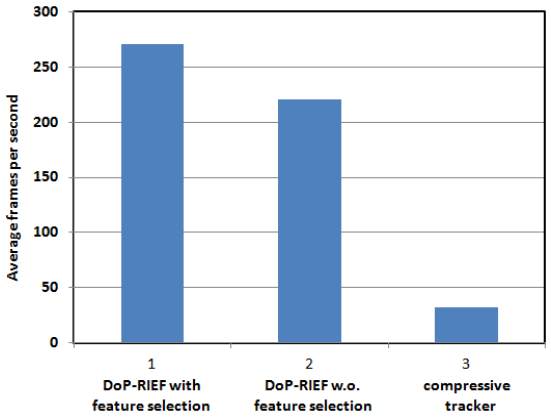


Fig.6. speed comparison result among our algorithm with feature selection, our algorithm without feature selection, and compressive tracker.

V. CONCLUSION

In this paper, we proposed the selective DoP-RIEF which can be used for mobile real-time tracking in augmented reality. Following the framework of feature extraction, feature distribution fitting with histogram, feature selection, classification, and updating, our algorithm gains higher speed and accuracy, compared with compressive tracker. Our algorithm has better robustness in scenes of illumination variation and image blur. With the expanded search range, it can also handle rotation and scale variations. Therefore, with the advantages of speed, accuracy and robustness, our algorithm is more suitable for applications on mobile devices with limited computational resources.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under grant 61271311 and National High-tech R&D Program of China (863 Program) under grant 2015AA016302.

REFERENCES

[1] D. Comaniciu, V. R. Member, and P. Meer. Kernel-based object tracking. PAMI, pp. 564–575, 2003.

[2] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: a cascade particle filter with discriminative observers of different life spans. PAMI, pp. 1728–1740, 2008.

[3] P. P´erez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. ECCV, pp. 661–675, 2002.

[4] Adam, A., Rivlin, E., Shimshoni, I.: Robust fragements-based tracking using the integral histogram. CVPR, pp. 798–805, 2006.

[5] Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via online boosting. BMVC, pp. 47–56, 2006.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, Brief: Binary robust independent elementary features. Computer Vision ECCV, vol. 6314, pp. 778–792, 2010.

[7] Bradski G R. Computer Vision Face Tracking for use in a Perceptual User Interface. Intel Technology Journal, pp. 1–15, 1998.

[8] Zhong W, Lu H, Yang M. Robust object tracking via sparsity-based collaborative model. Computer Vision and Pattern Recognition (CVPR), IEEE Conference on. IEEE, pp. 1838–1845, 2012.

Fig. 7 Experiment results on Stanford datasets

Fig. 8 Experiment results on Peking University landmark datasets

[9] Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF. Computer Vision (ICCV), IEEE International Conference on. IEEE, pp. 2564–2571, 2011.

[10] Zhang K, Zhang L, Yang M. Real-Time Compressive Tracking. Computer Vision – ECCV, pp. 864–877, 2012

[11] Collins R T, Liu Y, Leordeanu M. Online selection of discriminative tracking features. Pattern Analysis and Machine Intelligence, IEEE Transactions on, pp. 1631–1643, 2005.

[12] R. Kalman, A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, March, pp. 35–46, 1960.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, vol.60, pp.91–110, 2004.

[14] Schmalstieg D, Drummond T, Mulloni A, et al. Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. Valzaon and Omr Grah Ranaon on, pp.355–368, 2010.

[15] H. Bay, T. Tuytelaars, and L. Gool. SURF: Speeded up robust features. ECCV, pp. 404–417, 2006.

[16] E. Rosten and T. Drummond. Machine learning for high speed corner detection. in 9th Euproean Conference on Computer Vision, vol. 1, pp. 430–443, 2006

[17] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariantscalable keypoints. ICCV, pp. 2548–2555, 2011.

[18] Grabner, H., Leistner, C., Bischof, H.: Semi-supervised On-Line Boosting for Robust Tracking. ECCV, pp. 234–247, 2008.

[19] Babenko, B., Yang, M.-H., Belongie, S.: Robust object tracking with online multiple instance learning. PAMI , pp. 1619–1632, 2011.

[20] Mei, X., Ling, H.: Robust visual tracking and vehicle classification via sparse representation. PAMI , pp. 2259–2272, 2011.

[21] Kalal, Z., Matas, J., Mikolajczyk, K.: P-n learning: bootstrapping binary classifier by structural constraints. CVPR, pp. 49–56, 2010.

[22] Hare, S., Saffari, A., Torr, P.: Struck: structured output tracking with kernels. ICCV, pp. 263–270, 2011.