Real-time Depth-based Tracking Using A Binocular Camera

Leijie Zhang, Zhiqiang Cao, Xiangrui Meng, Chao Zhou, Shuo Wang

***Abstract*—Depth map provides rich information and it can be utilized in object tracking to handle some challenging problems in conventional RGB tracking such as occlusions and model drift. In this paper, we present a tracker that provides an effective real-time target tracking method based on a binocular camera. The proposed tracker is an extension of the popular KCF algorithm that leverages a circulant structure of tracking-by-detection with kernels for tracking. On this basis, we design a simple yet effective method to detect occlusions and recover tracking using noisy depth map obtained from a binocular camera. Firstly, one needs to identify exact and reasonable peak number of the target region's depth histogram and apply GMM model to evaluate the depth distribution. The occlusion is then detected based on the depth evaluation and the maximum response from KCF. When the occlusion happens, it is segmented to build the corresponding search region for recovery. The experimental results demonstrate the effectiveness of the proposed method and the superiority to the KCF tracker.**

I. Introduction

Visual object tracking is one of the most challenging tasks in the field of computer vision, with wide-ranging applications including human-computer interaction, surveillance and robotics, and so on. Recently, thanks to the popularity of low-cost RGB-D sensors (e.g. Microsoft Kinect), tracking can be made easier by using reliable depth data. The emergence of these has encouraged the researchers to combine color with depth data for designing robust trackers, which can effectively prevent model drift and handle occlusion with better adaptability to illumination variation. However, it is restricted to indoor applications with a shorter effective distance.

While depth-based tracking using the binocular camera is often ignored since it is unstable and time-consuming to obtain depth map, it has enormous advantages in outdoor with intense light irradiation and long distance away from the target. Some stereo systems are also presented for object tracking, e.g. [1-4], but it is not applicable for fast moving platform (e.g. mobile robots). Fortunately, faster and more accurate stereo matching algorithms have been proposed, such as BP [5, 6], SGBM [7]. And the algorithms can also been accelerated by parallel computing (e.g. GPU). Furthermore, some RGB trackers have achieved high-speed tracking, one of which is Kernelized Correlation Filters tracker (KCF) [8]. The KCF tracker combines high accuracy and fast processing speeds as demonstrated in [8, 9] where over 150fps processing was reported. Compared with other state-of-the-art trackers, e.g. camshaft [10], PF [11], CT [12], Struck [13], TLD [14], the KCF tracker can be further developed without considering the speed limit. Through these efforts, real-time tracking based on the binocular camera become possible. However, the direct usage of the KCF tracker cannot handle occlusions. In this case, the combination of KCF and depth map provides an effective solution.

In this paper, we present a real-time depth-based tracker using a binocular camera, which is based on, and improves upon, the existing color-only KCF tracker [8] with FHOG [17] to describe the target's features. In order to generate a depth map, the SGBM algorithm is adopted and a 640×480 depth map in over 10fps is acquired. Also, we analyze the depth distribution of target object, background, and occlusion. Our method mainly focuses on occlusion handling, though it can also increase the robustness of the KCF tracker including model drift and camera dithering. The key idea is to evaluate sudden changes in the target region's depth histogram so as to handle occlusions for recovery by searching some specific areas. Different from [15] that utilizes single Gaussian model to evaluate the depth distribution on the basis of KCF, we design a new scheme to detect occlusion and recover tracking from occlusion by the following procedures. The peak number of the depth histogram is identified so as to apply GMM (Gaussian Mixture Model) efficiently, and the search areas shall be reduced as largely as possible for fast recovery. To evaluate the proposed method, some scenarios with high diversity including deformable objects and various occlusion conditions are considered. Besides, we also compare our algorithm with the KCF tracker [8] on our dataset from a binocular camera.

The rest of the paper is organized as follows. Section II introduces the existing KCF tracker. The real-time depth-based tracking method based on a binocular camera is described in Section III in detail. In Section IV, experimental results are given. Section V concludes the paper.

This work is supported in part by the National Natural Science Foundation of China under Grants 61273352, 61233014, and in part by the National High Technology Research and Development Program of China (863 Program) under Grant 2015AA042201, and in part by the Beijing National Science Foundation under Grants 4161002, 4152054, and in part by the national defense basic research program under Grant B132011××××.

Leijie Zhang, Zhiqiang Cao, Xiangrui Meng, Chao Zhou, Shuo Wang are with State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. Zhiqiang Cao is the corresponding author. (e-mail: {zhangleijie2014, zhiqiang.cao, mengxiangrui2014, chao.zhou, shuo.wang}@ia.ac.cn)

II. Kernelized Correlation Filters Tracker

Henriques *et al.* [8] proposed the usage of the 'kernel trick' to extend correlation filters for fast RGB tracking. In the comprehensive RGB tracking benchmarking work in [9], KCF ranked first in terms of speed and accuracy compared to other approaches.

In the first frame, the KCF tracker trains a model with the image patch at the initial position of the target. This patch is

larger than the target, to provide some context. For each new frame, the KCF tracker detects over the patch at the previous position, and the target position is updated to the one that yielded the maximum value. Finally, the KCF tracker trains a new model at the new position, and linearly interpolate the obtained values of the classifier parameter with the ones from the previous frame, to provide the tracker with some memory [8].

A. Fast Kernel Regression

Ridge Regression can achieve performance that is close to more sophisticated methods, such as Support Vector Machines, but it's simple and fast with a closed-form solution. The goal of training is to find a function $f(z) = w^T z$ that minimizes the squared error over samples x_i and their regression targets y_i,

$$\min_w \sum_i (f(x_i) - y_i)^2 + \lambda \|w\|^2 \quad (1)$$

where λ is a regularization parameter to control overfitting. Solving the regression with shifted samples enables fast computation with fast Fourier Transforms instead of expensive matrix algebra. The positive and negative samples are rearranged as X, each of whose rows denotes one sample. What we have just obtained is a circulant matrix, which are made diagonal by the Discrete Fourier Transform (DFT), regardless of the generating vector x. This can be expressed as

$$X = F^H diag(\hat{x}) F \quad (2)$$

where F, known as the DFT matrix, is a constant matrix that does not depend on x, and $\hat{x}$ denotes the DFT of the generating vector, $\hat{x} = F(x)$.

Applying kernel trick to allow more powerful classifier for non-linear situation, the closed-form solution to the kernelized version of Ridge Regression is given as follows.

$$\alpha = (K + \lambda I)^{-1} y \quad (3)$$

where K is the kernel matrix and α is the vector of coefficients α_i, that represent the solution in the dual space. If we use kernels such as Gaussian, linear, polynomial, K is a circulant matrix. It is possible to diagonalize Eq. (3) as in the linear case, and we have

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \quad (4)$$

where k^{xx} is the first row of the kernel matrix K.

B. Fast Detection

The circulant matrix trick can also be applied in detection to speed up the whole process. The patch z at the same location in the next frame is treated as the base sample to compute the response in Fourier domain, and a confidence map y can be obtained by:

$$y = F^{-1}(\hat{k}^{xz} \odot \hat{\alpha}) \quad (5)$$

the position with a maximum value in y can be predicted as new position of the target.

C. Fast Kernel Correlation

It has been proven that the kernel function of a circulant kernel matrix should be unitarily invariant (detailed proof can be found in [8]). Since dot-product and radial basis kernel functions are found to satisfy this condition, polynomial kernels and Gaussian kernels are usually applied.

If the kernel k is computed between x and x', the Gaussian kernel $k^{xx'} = exp\left(-\frac{1}{\sigma^2}(\|x - x'\|)^2\right)$ can be computed by:

$$k^{xx'} = exp\left(-\frac{1}{\sigma^2}(\|x\|^2 + \|x'\|^2 - 2F^{-1}(\hat{x}^* \odot \hat{x}'))\right) \quad (6)$$

If the feature space is multi-dimensional, the integration of multiple channels does not result in a more difficult inference problem. Thus, we merely have to sum over the channels when computing kernel correlation.

III. Proposed Tracking Algorithm

In this section, we develop our depth-based tracking algorithm using a binocular camera. The flowchart of the proposed tracking method is shown in Fig.1.

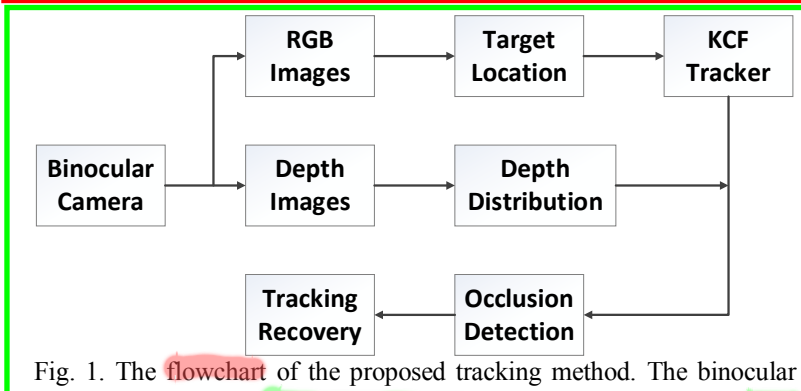


Fig. 1. The flowchart of the proposed tracking method. The binocular camera captures two left and right images simultaneously, and the depth map is obtained by stereo matching with the reference image. Then the occlusion can be handled during the process of the KCF in cooperation with the depth map analysis.

Specifically, the proposed depth-based tracking algorithm works as follows. The object of interest is manually selected in the first frame by a bounding box. Then we extract features of the target to initialize the KCF tracker with fast training, and obtain depth map by stereo matching. For one incoming video frame t, the KCF tracker executes fast kernel correlation and fast detection to find the max response, which decides on the new target position. Meanwhile the corresponding histogram is created from the depth map, and the number of peaks is computed with the findPeakNum algorithm. Afterwards the depth distribution, including the object, the background, or the occlusion, can be estimated effectively and efficiently with GMM. According to the depth distribution at frame *t*-1, *t* and the max response at frame *t*, the occlusion is detected and recovered. If there is no occlusion, the KCF tracker is adaptively updated until the task is completed.

Below we give a detailed description about each component of our method.

A. Depth Distribution

Because the depth map obtained from stereo matching is full of noise and not so reliable, the method of segmenting with k-means and region growing, referred to [15], does not work well for estimation of the depth distribution. We design a simple yet effective scheme to compute the depth distribution.

Algorithm 1 *findPeakNum*

Input: $H_{1\times256}$ the depth histogram.
Output: $peakNum$ the number of peaks.
$peakCenter$ the corresponding depth value of the peak.

$peakNum \leftarrow 0$
for all $H[i] \in H$ **do**
  if $\forall H[j] \in H[0 \ldots i] \neq 0$ **or** $\forall H[j] \in H[i \ldots 255] \neq 0$ **then**
    $H[i]$ *will be set to* 0 *when the loop is over.*
  end if
end for
while $H[index] \leftarrow max\{H\} > minPeakValue$ **do**
  $H_{accu} \leftarrow 0$
  $j \leftarrow index, i \leftarrow index$
  while $H[i] > minPeakValue$ **do**
    $H_{accu} = H_{accu} + H[i]$
    $i \leftarrow i - 1$
  end while
  while $H[j] > minPeakValue$ **do**
    $j \leftarrow j + 1$
    $H_{accu} = H_{accu} + H[j]$
  end while
  if $H_{acc} > 0.2$ **then**
    $peakNum \leftarrow peakNum + 1$
    $peakCenter[peakNum] \leftarrow index$
  end if
end while

Fig. 2. The algorithm to find peak number in depth histogram.

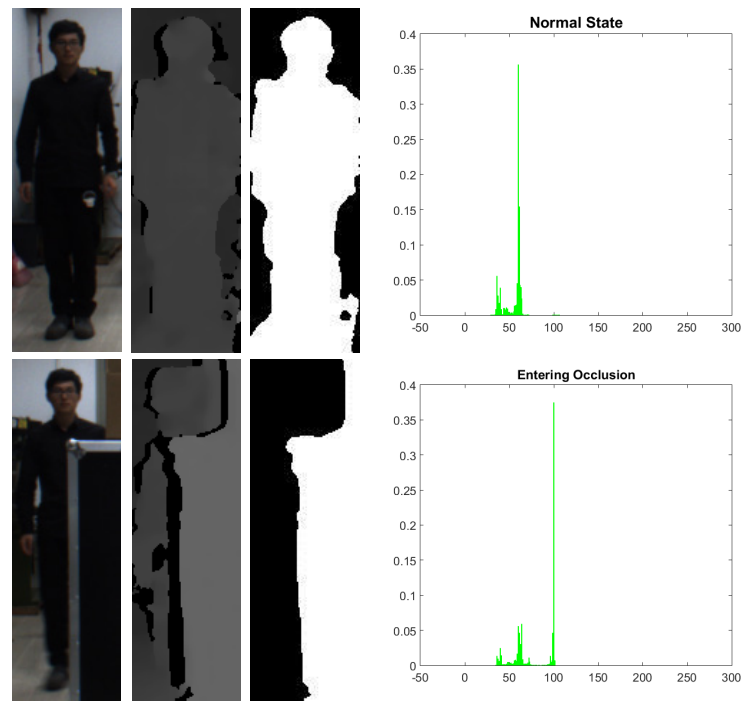


Fig. 3. Depth distribution inside the bounding box. The top row shows the distribution in normal state, and the bottom row shows the distribution when occlusion occurs. Each row from left to right represents the RGB image, the depth image, the segmented foreground image, and the depth

First we delete the bars of 0 and 255 as well as the bars which connect to these in the histogram. Then the findPeakNum algorithm (see Fig. 2 for more details) is applied to find reliable peak number, so as to identify the number of Gaussian functions in GMM model for computation accelerating.

The different peak number indicates different depth distribution. Assuming larger depth value of the pixel maps into the point closer to the camera, we can summarize as two kinds of cases whether there exist occlusions or not. If no occlusion is detected, when the peak number is 1, the depth of the object is just the value obtained from GMM. For the cases where the peak number is 2 and 3, we conclude that the depth of the object equals to the value that has the minimum difference from the last updated object depth. And if the max depth value is not occupied by the object, it's regarded as the depth of the occlusion. It shall be noted that the depth information does not update for other values of the peak number. Once the occlusion is detected, the occlusion depth is updated the value if peak number is 1; and when peak number comes to 2, 3, or other value, the processing is the same as the case of tracking. The process of the depth map is presented in Fig. 3.

B. Occlusion Detection

From the depth distribution described in Fig. 3, we can figure out the object depth value and the fraction of pixels belonging to each component in the region of object window. The fraction of belonging to each component is calculated through GMM with the value of pixels in $[\mu - 3\sigma, \mu + 3\sigma]$ and the pixels connected to each other in terms of the target window. The occlusion appears when the depth value of the component closest to the camera changes significantly, and the number of peaks usually increases. The occlusion is detected if

$$(f(z)_{max} < \lambda_1) \wedge \left((\Phi(\Omega_{obj}) < \lambda_{obj}) \vee (\Phi(\Omega_{occ}) > \lambda_{occ})\right) \quad (7)$$

where $\Phi(\Omega_{obj})$ and $\Phi(\Omega_{occ})$ are the fraction of pixels belonging to Ω_{obj} and Ω_{occ} respectively. The first term in Eq. (7) reduces false detections of occlusion in the case of objects moving fast away from or towards the camera. The other two terms in Eq. (7) indicate that the occlusion is detected under the condition that the fraction of pixels belonging to the object is less than a specific value and the fraction of pixels belonging to the occlusion is greater than a specific value.

C. Tracking Recovery

As shown in Fig. 3, the component closest to the camera is extracted from the depth map of the target window, and the occlusion region in the whole depth map can be determined through occRegionGrowth algorithm (see Fig. 4 for more details). During occlusion, a search region Ω_{search} is defined, and its response and as well as the new depth distribution is computed to detect the re-appearance of the target. Actually, the target object will re-emerge with high probability in those edges areas of the occlusion. We identify the region where target candidates are searched as

$$\Omega_{search} = \eta_1 \Omega_{occ} - \eta_2 \Omega_{occ} \quad (8)$$

where η_1, η_2 are the proportion parameters to enlarge and shrink the occlusion region respectively and $\eta_1 > \eta_2$. As the patch is ξ (ξ is usually designed as 1.2~2) times larger than the target window, η_1, η_2 are not required to be much larger or much smaller than 1 for the search region decreasing. And the tracking is resumed when

$$(f(z)_{max} > \lambda_1) \wedge \left((\Phi(\Omega_{obj}) > \lambda_{obj})\right) \quad (9)$$

where The first term in Eq. (9) makes sure the object is the target tracked before. And the second term implies that the fraction of pixels belonging to the target should be greater than a specific value. For extracting more exact search region, we add a judgement before tracking recovery that when Eq. (9) is

Algorithm 2 *occRegionGrowth*

Input: im the depth image.
occ_{model} mean and variance of the occlusion.
obj_{window} the region of the track window.
Output: im_{occ} the occlusion region.

$im_{occ} \leftarrow im$
$\mu \leftarrow occ_{model}[1]$, $\Delta \leftarrow 3 \times occ_{model}[2]$
for $i = 0$ **to** $row(im_{occ}) - 1$ **do**
  for $j = 0$ **to** $col(im_{occ}) - 1$ **do**
    if $im_{occ}[i][j] \in [\mu - \Delta, \mu + \Delta]$ **then**
      $im_{occ} \leftarrow 1$
    else
      $im_{occ} \leftarrow 0$
    end if
  end for
end for
for all $im_{occ}[i][j] \neq 0$ **do**
  Label the different regions from 1 to lable.
end for
for all $region[i]$, $i \in [0, label - 1]$ **do**
  if $region[i] \cap obj_{window} = \varnothing$ **then**
    for all $im_{occ} \in region[i]$ **do**
      $im_{occ} \leftarrow 0$
    end for
  end if
  $im_{occ} \leftarrow max\{regions\ of\ im_{occ}\}$
end for

Fig. 4. The algorithm to generate occlusion areas.

nearly satisfied (just decrease λ_2 and λ_{obj} a little), the tracking is activated but the model is still not updated until Eq. (9) is completely satisfied. In order to reduce computation during the search procedure, the depth distribution of the target window is calculated only in the window with the max response.

IV. Experimental Results

The performance of the proposed depth-based tracking method is evaluated in this section. The algorithm is implemented with the combination of the KCF codes provided by [8] computer with Intel i5-2400 CPU 3.10 GHz processor. It achieves the processing speed of about 20fps at the resolution of 640 × 480 pixels. The dense depth images from a PointGrey BB2-08S2C-38 binocular camera are generated with 10 fps by the SGBM algorithm [7]. The 10 fps processing can satisfy the real-time applications. . The parameters used in the algorithm are as follows: λ_1=0.35, λ_{obj}=0.3, λ_{occ}=0.25, η_1=1.3, η_2=0.7.

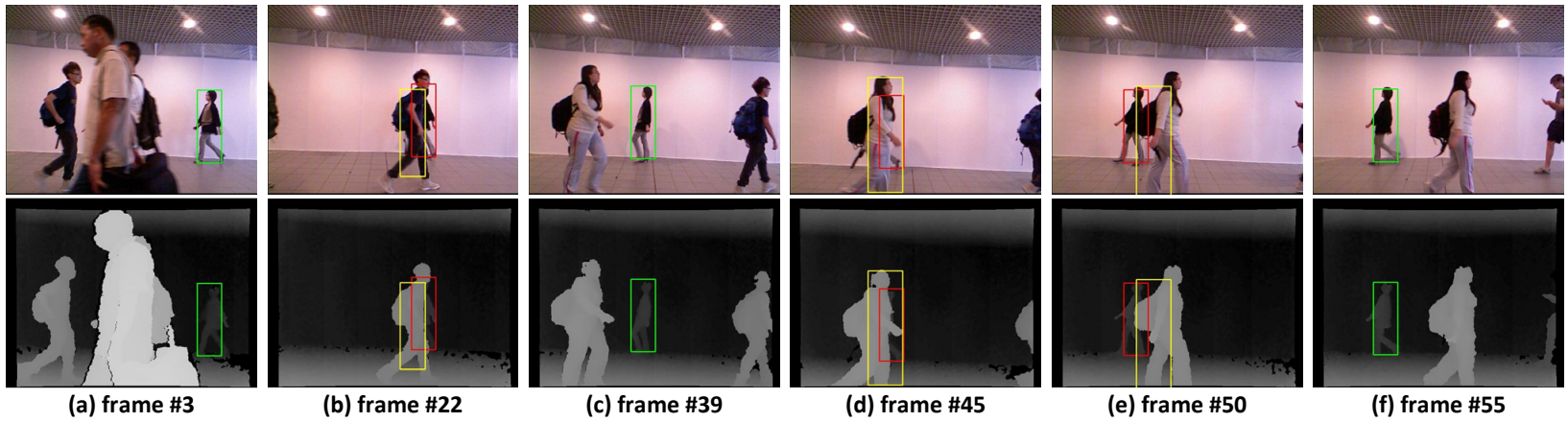
We first validate our method using the Princeton Dataset [16], which is RGB-D tracking datasets recorded with Microsoft Kinect v1. This dataset presents complex background clutter and intermittent occlusions. The tracking results of the proposed method are shown in Figs. 5 and 6, where the green box is the tracking window that indicates there is no occlusion and the tracking can be trust; the yellow box denotes the occlusion region. The red box is used to describe the target with a high probability during the occlusion, and the red box moving represents the activation of the tracking without model updating. Since the depth map provided by Microsoft Kinect v1 is stable, high-precision and has less black holes, the proposed method is not surprisingly proved to be effective for occlusion handling.

In the following, we test our method on our dataset from a binocular camera. The performance of our tracking algorithm is shown in Figs. 7-9. Take Fig. 9 of tracking a little toy as an example. Obviously, the depth maps from our dataset have stronger noise and less stable than those from RGB-D cameras. However, the proposed method still can achieve the tracking smoothly.

Furthermore, we compare our proposed method with the KCF tracker. The comparison results are shown in Fig. 10 and Fig. 11, where the blue box denotes the results of the KCF tracker. Fig. 10 demonstrates the case of occlusion handling while Fig. 11 presents the case of camera dithering. One can see that when the occlusion (the blue box) appears, the model for tracking stops updating (the red box), and the search region generates proposals for tracking recovery. In addition, if the camera is heavily shaking, the maximum response from tracking will be below a threshold and the depth distribution will change significantly, which can also activate occlusion handling. These experiments demonstrate that the proposed method is more robust and effective than the KCF tracker.

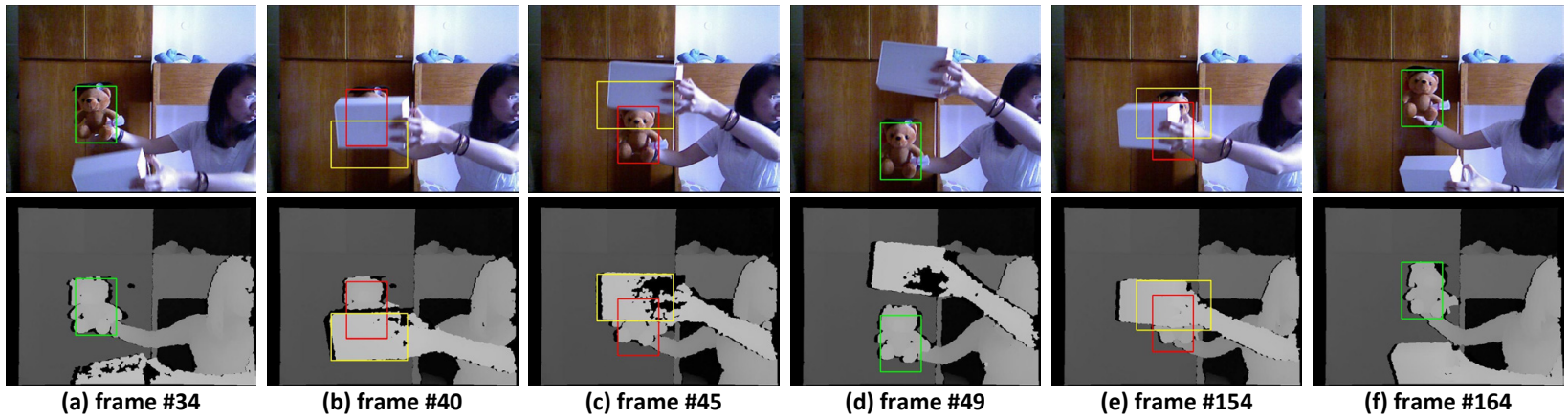
V. Conclusion

In this paper, we have proposed a depth-based tracking method using a binocular camera, which can effectively and efficiently solve the problem of occlusion, and also increase the robustness for camera dithering. Our experimental results demonstrate the correctness of the proposed method and the superiority compared to KCF algorithm. In the future, we will focus on further optimizing and increasing the speed of the algorithm, especially in the process of the stereo matching. Besides, we will apply the proposed method to robots for target tracking in complex environments.



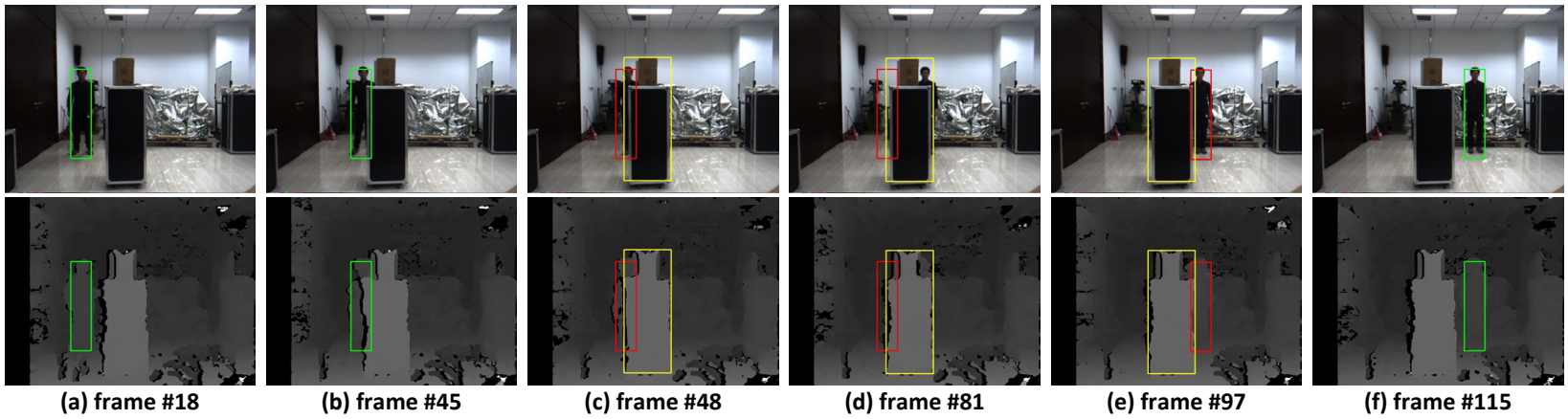
(a) frame #3 (b) frame #22 (c) frame #39 (d) frame #45 (e) frame #50 (f) frame #55

Fig. 5. Tracking results of the proposed method on a sequence named "walking_occ1" of the Princeton dataset. The green box is the tracking window that indicates there is no occlusion and the tracking can be trust; the yellow box denotes the occlusion region. The red box is used to describe the target with a high probability during the occlusion, and the red box moving represents the activation of the tracking without model updating.



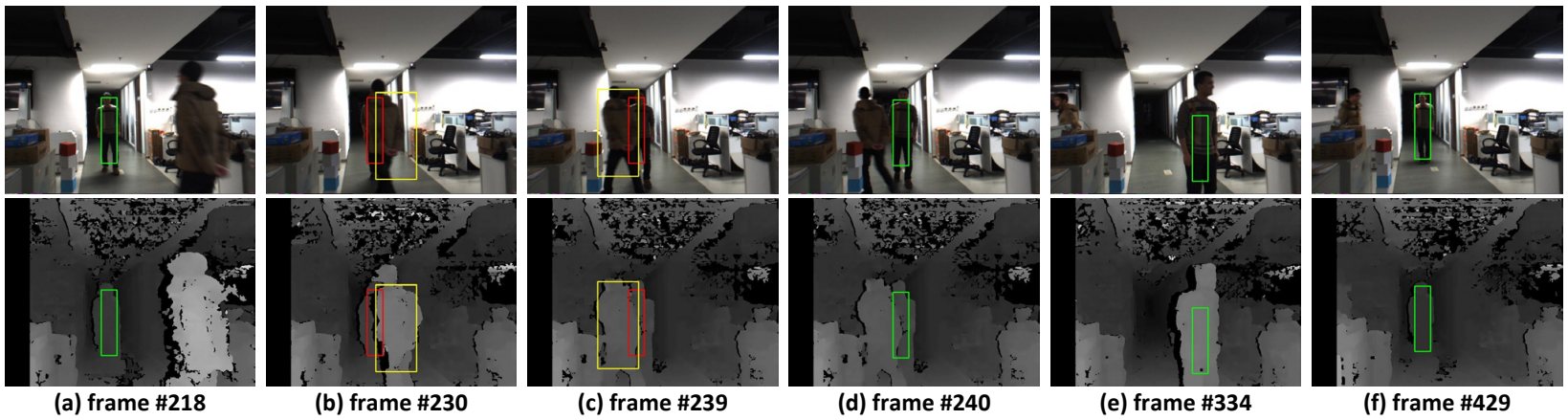
(a) frame #34 (b) frame #40 (c) frame #45 (d) frame #49 (e) frame #154 (f) frame #164

Fig. 6. Tracking results of the proposed method on a sequence named "bear_front" of the Princeton dataset.



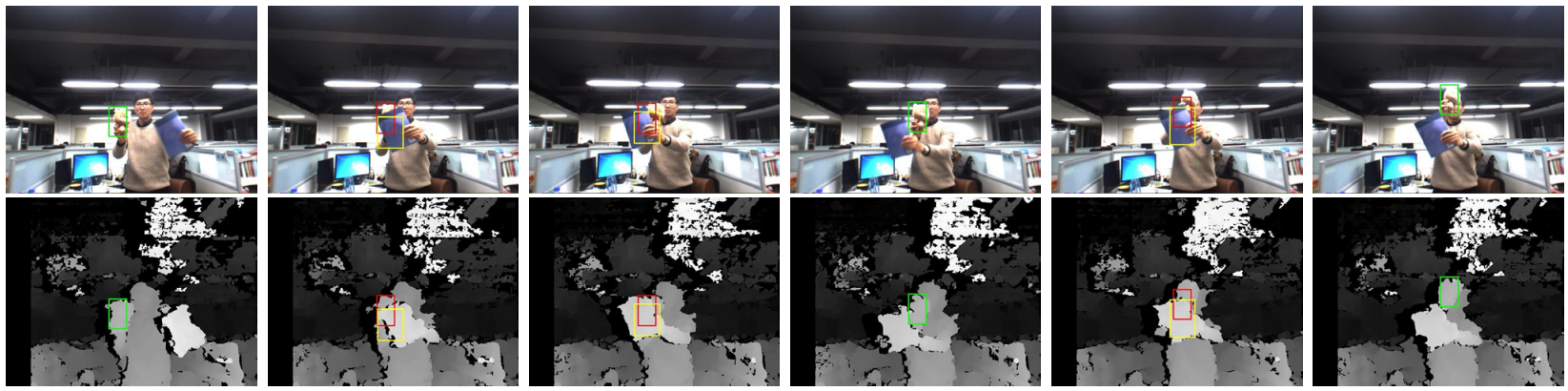
(a) frame #18 (b) frame #45 (c) frame #48 (d) frame #81 (e) frame #97 (f) frame #115

Fig. 7. Tracking results of the proposed method on our dataset. A man walks through an occlusion (frame #48-#97), and the occlusion is detected (yellow box). Finally the tracking is recovered.



(a) frame #218 (b) frame #230 (c) frame #239 (d) frame #240 (e) frame #334 (f) frame #429

Fig. 8. Tracking results of the proposed method on your dataset. A man blocks another man while walking. The occlusion denoted as the man is detected with the yellow box (frame #230-#239), and the other man is tracked by the red box during occlusion.

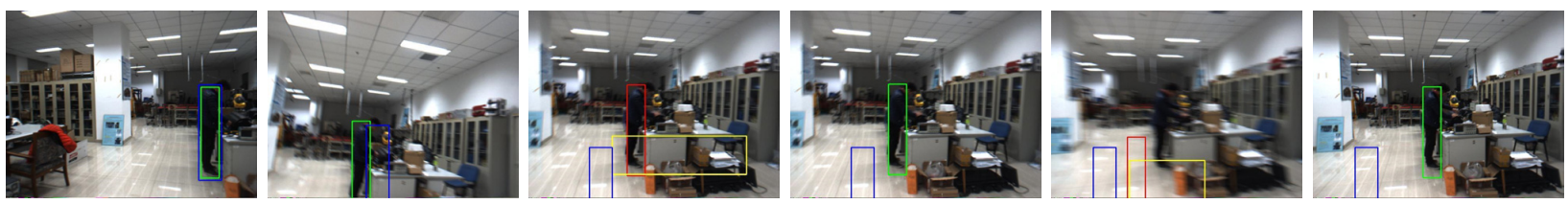


(a) frame #55 (b) frame #67 (c) frame #83 (d) frame #89 (e) frame #191 (f) frame #196

Fig. 9. Tracking results of the proposed method on our dataset. A little toy is tracked and the occlusion, which is a book, is detected (frame #67-#83). Finally the tracking is recovered.

(a) frame #1 (b) frame #64 (c) frame #70 (d) frame #72 (e) frame #100 (f) frame #131

Fig. 10. The comparison of our method with KCF tracker in terms of occlusion. When the occlusion occurs (frame #72), the KCF tracker (blue box) is out of work. While our method detects the occlusion and recovers the tracking.



(a) frame #1 (b) frame #161 (c) frame #246 (d) frame #247 (e) frame #270 (f) frame #271

Fig. 11. The comparison of our method with KCF tracker in terms of camera dithering. Our method can deal with the camera dithering (frame #161-#270), while the KCF tracker (blue box) cannot recover once it loses tracking.

REFERENCES

[1] L. J. Cao, C. Wang, and J. Li, "Robust depth-based object tracking from a moving binocular camera," *Signal Processing*, vol. 112, pp. 154-161, 2015.

[2] Y. Ye, S. Ci, Y. W. Liu, "Binocular Video Object Tracking with Fast Disparity Estimation," in *Conf. on Advanced Video and Signal Based Surveillance.* pp. 183-188, 2013.

[3] K. Zhu, Q. Q. Ruan, T. W. Yang, "Tracking and Measuring a Moving Object with a Binocular Camera System," in *International Conference on Signal Processing*, pp. 1414-1419, 2008.

[4] W. C. Lee, P. Sebastian, "Stereo Vision Tracking System," in *Conf. on Future Computer and Communication.* pp. 487-491, 2009.

[5] J. Sun, N. Zheng, and H. Shun, "Stereo matching using belief propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25(7), pp. 787-800, 2003.

[6] P. Felzenszwalb, D. Huttenlocher, "Efficient belief propagation for early vision," *International Journal of Computer Vision*, 70(1), pp. 41-54, 2006.

[7] H. Hirschmuller, "Stereo Processing by Semi global Matching and Mutual Information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(2), pp. 328-341, 2008.

[8] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 37(3), pp.583-596, 2015.

[9] Y. Wu, J. Lim, and M. Yang, "Online Object Tracking: A Benchmark," in *Computer Vision and Pattern Recognition,* pp. 2411-2418, 2013.

[10] G. J. Dai, Z. Yun, "A Novel Auto-Camshift Algorithm Used in Object Tracking," in *Chinese Control Conference*, pp. 369-373, 2008.

[11] H. X. Chu. Z. Y. Xie, X. J. Nie. "Particle Filter Target Tracking Method Optimized by Improved Mean Shift," in *International Conference on Information and Automation*, pp. 991-994, 2013.

[12] K. Zhang, L. Zhang, M. H. Yang, "Fast Compressive Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 36(10), pp. 2002-2015, 2014

[13] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence*, doi: 10 1109/TPAMI.2015.2509974 , 2015.

[14] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(7), pp. 1409-1422, 2012.

[15] M. Camplani, S. Hannuna, M. Mirmehdi, "Real-time RGB-D Tracking with Depth Scaling Kernelised Correlation Filters and Occlusion Handling," *in British Machine Vision Conference.* 2015

[16] S. Song, and J. Xiao, "Tracking revisited using RGBD camera: Unified benchmark and baselines," in *International Conference on Computer Vision*, pp. 233-240, 2013.

[17] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 32(9), pp. 1627-1645, 2010.