Depth assisted Tracking Multiple Moving Objects under Occlusion

Anh Tu Tran† and Koichi Harada††

Graduate School of Engineering, Hiroshima University, Higashi-hiroshima, Hiroshima, 739-8521, Japan

Summary

In this paper, we have presented a novel tracking method aiming at detecting objects and maintaining their label/identification over the time. The main key factors of this method are to use depth information and different strategies to track objects under various occlusion scenarios. The foreground objects are detected and refined by background subtraction and shadow cancellation. The occlusion detection is based on information of foreground blobs in successive frames. The occlusion regions are projected to the projection plane XZ to analysis occlusion situation. According to the occlusion analysis results, different objects correspondence strategies are introduced to track object under various occlusion scenarios including tracking occluded objects in similar depth layer and in different depth layers. The experimental results show that our proposed method can track the moving objects under the most typical and challenging occlusion scenarios.

Key words:

Visual tracking, multiple object tracking, stereo tracking, occlusion analysis.

1. Introduction

Object tracking in the video sequence has played an important role in a research area of computer vision and a wide range of applications, such as video monitoring and surveillance, video conferencing and video summarization. Based on different camera configurations, objects can be tracked by using a single camera or stereo/multiple cameras. Object tracking with a single camera has studied in many literatures and difference methods have been developed such as tracking by model-based tracking method [1], appearance-based methods [2-4], feature-based tracking [5], and statistical methods [6-8]. Many algorithms can obtain good results in some cases, such as when the targets are separated. However, multiple object tracking is still a challenging task due to the non-rigid motion of deformable object, persistent occlusion and the dynamic change of object attributes, such as color distribution, shape and visibility. In the real scene, occlusion between objects often occurs. For example, in typical surveillance scenario a person is partially or fully occluded by other people. Unfortunately, these occlusions lead to failed tracking. Some classical frameworks have been extended to track multi objects. In the multi-object tracking system [9], level set method is used to handle contour splitting and merging. Extensive methods, i.e. Monte Carlo based probabilistic methods [10], game theory based approaches [11] and appearance model based deterministic methods [12, 13] have been presented to solve the mutual occlusion problem. Another attractive research direction is stereo or multiple camera based method. While object detection and tracking with a single camera is a well-explored topic, the use of multi-cameras technology for this purpose has been attracted much attentions recently due to the availability and low price of new hardware. A multi-camera system observes the scene from two or more different views, and obtains more comprehensive information than a monocular camera system, which can take the advantage of depth information to improve the tracking system performance. Some tracking methods focus on usage of depth information only [14], or usage of depth information on better foreground segmentation [15], or usage depth information as a feature to be fused in a maximum likelihood model to predict 3D object positions [16].

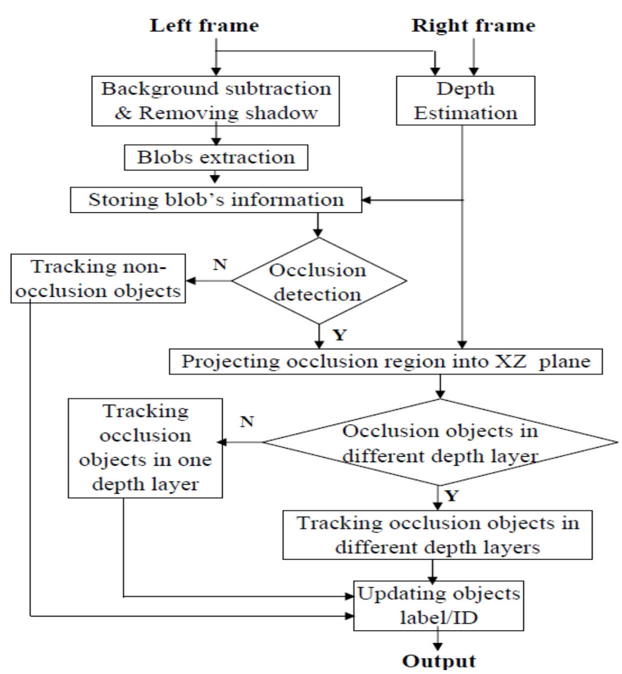


Fig. 1. The flowchart of proposed tracking method.

Manuscript received May 5, 2013
Manuscript revised May 20, 2013

In this work, we have presented a novel tracking method aiming at detecting objects and maintaining their label/identification across video frame sequence. The main points of this method are to use depth information and different strategies to track objects under various occlusion scenarios. Fig. 1 shows the flowchart of our tracking system.

The rest of this paper is organized as follows: section 2 presents the proposed tracking method. Section 3 shows experimental results; and, finally, Section 4 concludes this paper.

2. Proposed tracking method

Our proposed tracking system is shown in Fig. 1 and it consists of below main steps.

2.1. Depth estimation

Depth estimation aims at calculating the structure and depth of objects in a scene from two views or a set of multiple views. This topic has been attracted extensive attentions in research communities. A comprehensive survey and evaluation of dense two-view stereo matching algorithms can be found in [17].

In this work, depth is estimated based on block matching algorithms proposal in [18]. This block matching technique is a one-pass stereo matching algorithm that uses a sliding sums of absolute differences window between pixels in the left image and the pixels in the right image. An example of depth image is shown in Fig. 2.

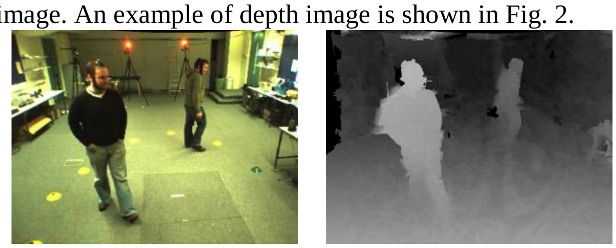


Fig. 2. Color image and depth image.

2.2. Foreground segmentation and shadow cancellation

Our method performs foreground segmentation to speed up the process of object tracking. There are many foreground segmentation algorithms for instance of Gaussian mixture model [19, 20]. In our method, we use simple technique based on absolute differences between current image and background image.

In some cases, we have the fixed cameras observing the scene, so we may have an image of the background of the scene. However, in most case this background is not readily available. Moreover, the background scene often evolves over time because for example the light condition might change or because of new object could be added or removed from the background. Therefore, it is necessary to dynamically build the background model by regularly updating it. This can be accomplished by computing moving average using the following formula:

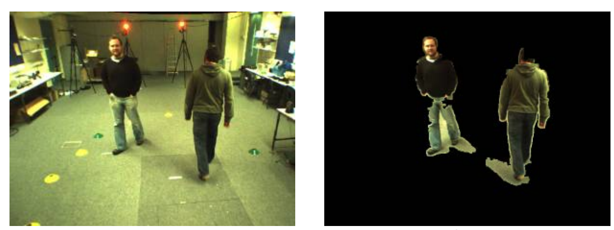
$$\mu_t = (1-\alpha)\mu_{(t-1)} + \alpha p_t, \tag{1}$$

where, p_t is pixel value ate a given time t, $\mu_{(t-1)}$ is the current average value, and α is called the learning rate and it defines the influence of the current value.

In our method, first a color background model is created by computing a moving average for each channel (R, B and G channels of color image) of each pixel of incoming frames (around 10 frames). The decision to define a foreground pixel is simply based on comparing the current frame with background model and then updating this. Specifically,

$$F(p) = \begin{cases} 0 & \text{if } \left|\mathrm{I}_c(p) - I_{bg}(p)\right| < t_H \\ \mathrm{I}_c(p) & \text{otherwise} \end{cases}, \tag{2}$$

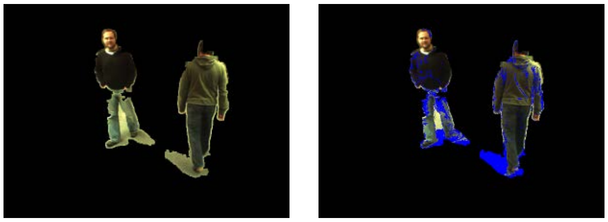
where, $F(p)$ is value of pixel p in foreground image, $\left|\mathrm{I}_c(p) - I_{bg}(p)\right|$ represents the absolute color difference between the color value at pixel p of current frame $I_c(p)$ and the color value of pixel p of background frame $I_{bg}(p)$ of R, G, B channels. t_H is threshold and for the each color channel this threshold can be set to $0.3 * I_{bg}(p)$. An example of foreground image is shown in Fig. 3.



a) input image b) segmented foreground image

Fig. 3. Foreground segmentation.

However, the segmented foreground image includes noise affected by the shadow. The shadow regions are the parts of moving objects. Shadow detection and removing will be used to refine the foreground. To avoid the effects of shadows, a shadow detection described in [21] is imployed. More detail about this method, please refer to [21]. The result of the shadow detection is shown in Fig. 4.



a) segmented foreground image

b) shadow detection (the blue color pixels are the detected shadow)

Fig. 4. Shadow regions detection

2.3. Blobs extraction and blobs' information store

First, we try to extract the blobs of objects from segmented foreground image. Blob extraction is performed on the foreground binary image by connected component labeling using CvBlobLib library [22]. The foreground binary image can obtained from simple threshold operation followed by the application of eroded and dilated operation on the segmented foreground image. CvBlobsLib provides two basic functionalities: extracting 8-connected components, referred to as blobs, in binary or grayscale images using Chang's contour tracing algorithm [23], and filtering the obtained blobs to get the objects in the image that we are interested in. In our method, we remove any blobs that have area small than 100 pixels.

For every frame, after extracting blobs, information about blob is stored in a structured record for later processing steps. The blob's information includes total number of blob (NB), blob's center coordination (x, y), blob area (BA), and the average depth value of blob $(Z_{\min})$. Blobs are also given temporally identification (ID).

We define two kinds of distance: the distance between blob i and blob j in the same frame t and the distance between blob i of frame t and blob j of frame $(t-1)$.

The distance $d(i_t, j_t)$ between blob i and blob j in the same frame t is computed by:

$$d(i_t, j_t) = \sqrt{(x_{i_t} - x_{j_t})^2 + (y_{i_t} - y_{j_t})^2}, \tag{3}$$

where, (x_{i_t}, y_{i_t}) and (x_{j_t}, y_{j_t}) are the center coordination of blob i and blob j at frame t, respectively.

Similarly, the distance $D(i_t, j_{(t-1)})$ between blob i of frame t and blob j of frame $(t-1)$ is calculated by:

$$D(i_t, j_{(t-1)}) = \sqrt{(x_{i_t} - x_{j_{(t-1)}})^2 + (y_{i_t} - y_{j_{(t-1)}})^2}, \tag{4}$$

where (x_{i_t}, y_{i_t}) and $(x_{j_{(t-1)}}, y_{j_{(t-1)}})$ are the center coordination of blob i at frame t and blob j at frame $(t-1)$ correspondingly.

2.4. Occlusion detection

We detect the occlusion in current frame t according to blobs' information at frame t and previous frame $(t-1)$. It is based on two clues. The first clue comes from the shortest distance between blobs at the same frame $(t-1)$ and the second one is the difference of number of blobs at frame t and $(t-1)$. First, we find the shortest distance $d(i_{(t-1)}, j_{(t-1)})$ between blobs in frame $(t-1)$, assuming that it occurs between blob m and blob n, i.e. $d_{\min}(n_{(t-1)}, m_{(t-1)}) = \min\{d(i_{(t-1)}, j_{(t-1)})\}$.

We define an occlusion flag f_{occ_start}. This flag gets value t if occlusion is found at frame t and otherwise it gets value -1. Specially,

$$f_{occ\_start} = \begin{cases} t & \text{if } d_{\min}(n_{(t-1)}, m_{(t-1)}) < d_{threshold} \text{ and } NB_t < NB_{(t-1)} \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

where, $NB_{(t-1)}$ and NB_t are the total number of blobs in frame $(t-1)$ and frame t respectively. $d_{threshold}$ is the threshold of blob distance at the same frame.

Similarly, we also detect when the occlusion terminates. The end of occlusion is checked based on the shortest distance between blobs at the current frame t and the difference of number of blobs at current frame and previous frame $(t-1)$. We define the end of occlusion flag f_{occ_end} as following:

$$f_{occ\_end} = \begin{cases} t & \text{if } d_{\min}(n_t, m_t) < d_{threshold} \text{ and } NB_{(t-1)} < NB_t \\ -1 & \text{otherwise} \end{cases}, \tag{6}$$

2.5. Object tracking

According to the result of occlusion detection, the tracking objects can be dividing into two types: tracking objects without occlusion and tracking objects under occlusion.

2.5.1. Tracking objects without occlusion

The video objects correspondence under non-occlusion is obtained through the shorted distance $D(i_t, j_{(t-1)})$ between blobs in previous frame and blobs in the current frame. This distance between blobs in previous frame and blobs in the current frame is calculated by Eq.(4). For instance, once a foreground blob m at frame t (B_t^m) finds its corresponding blob $B_{(t-1)}^n$ in frame $(t-1)$, its label or identification (ID) is updated correspondingly to the ID of blob $B_{(t-1)}^n$. Specially,

$$\begin{aligned} &ID \text{ of } B_t^m \equiv ID \text{ of } B_{(t-1)}^n \\ &\text{if} \begin{cases} D(B_t^m, B_{(t-1)}^n) = \min\{D(B_t^m, B_{(t-1)}^j)\}, j = 1,2,..,NB_{(t-1)} \\ D(B_t^m, B_{(t-1)}^n) < D_{thres} \end{cases} \end{aligned} \tag{7}$$

where B_j^k denotes the blob k at frame j; $NB_{(t-1)}$ is number of blobs in frame $(t-1)$; D_{thres} is distance threshold.

2.5.2. Tracking objects under occlusion

The main idea of our tracking method is that the object label or identification (ID) is maintained constantly during occlusion and after they switch their positions.

When occlusion occurs, we can detect and extract the occlusion region. We also can detect and separately extract a list of objects that are non-occlusion objects in previous frame but overlaying each other in current frame.

In order to track the objects under occlusion, depth information is used to analysis the occlusion situation. First, the occluded regions are projected to the ground plane XZ according to their horizontal position and their depth gray level (more detail in the next subsection). Then according to the XZ plane, the occlusion objects can be divided into two types based on the depth ranges: a) in the different depth layer or b) in the same depth layer. We are dealing with these situations as following parts.

2.5.2.1. Project occlusion regions into ground plane XZ

Each foreground pixel in occlusion region has $3D$ information obtained from depth map. These pixels are projected to the ground plane XZ according to their horizontal position and their depth gray level, where X is the width of the depth map and the range of Z is $[0, 255]$. The projected point, which is located at (x,z), is defined as $p(x,z)$. The value at position $p(x,z)$ of projection plane is the total number of points at position x in the depth maps that have same gray level (depth value z). Fig. 5 illustrates the image plane XY and ground plane XZ.

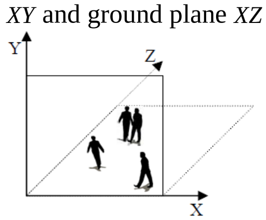
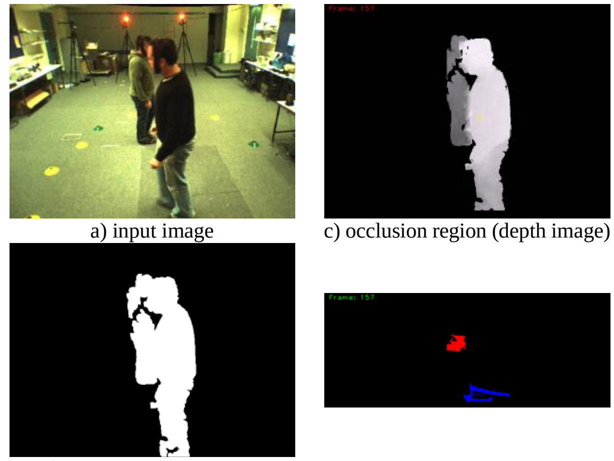


Fig. 5. The image plane XY and ground plane XZ

In order to remove noisy points, if the value at point $p(x,z)$ is less than threshold T_1 the point $p(x,z)$ will be discarded. Then we also apply morphological operations (dilating and eroding operation) to remove noisy points and connect nearby points. The remaining points in XZ plane are grouped in to the blobs that are based on connected component analysis technique using CvBlobLib library [22]. If a projected blob is small than threshold T_2, it is consider a noise and it will be removed. The projected blobs are defined as $\{PB_j \mid (j = 1,2,...,m)\}$, where m is total number of projected blobs. Each projected blobs PB_j is mask as object regions. Fig. 6 shows an example of projected blobs in XZ plane.

As mention before, according to the projected blobs in XZ plane, the occlusion objects can be divided into two types based on the depth ranges: a) in the different depth layers or b) in the one depth layer.



a) input image

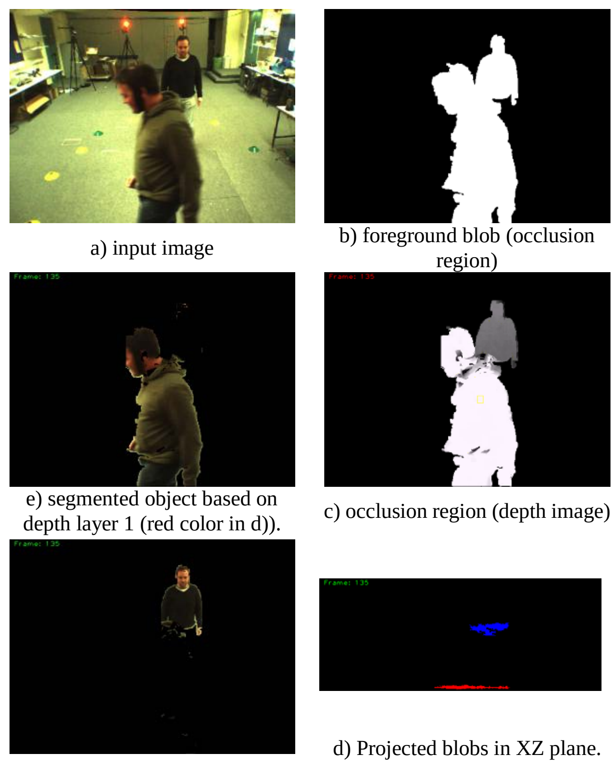
c) occlusion region (depth image)

b) foreground blob (occlusion region)

d) projected blobs in XZ plane

Fig. 6. Projected foreground blobs in XZ plane.

2.5.2.2. Tracking occluded objects in different depth layers



a) input image

b) foreground blob (occlusion region)

e) segmented object based on depth layer 1 (red color in d)).

c) occlusion region (depth image)

f) segmented object based on depth layer 2 (blue color in d)).

d) Projected blobs in XZ plane. (occluded objects in different depth layers).

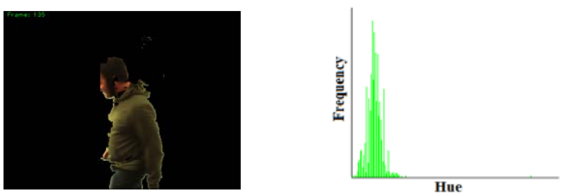
Fig. 7. An example of image segmentation by means of depth ranges.

Fig. 6 shows the case of occluded objects is in different depth layers. Once occluded objects have different depth layer, they can be segmented in color image by means of their depth ranges. An example of color image segmentation by means of depth ranges is shown Fig. 7.

In our method, object correspondence under different layer is based on Bhattacharyya distance [24] between the color histograms. In statistics, the Bhattacharyya distance measures the similarity of two probability distributions. In our case, Bhattacharyya distance represents the similarity between two normalized histograms. The Bhattacharyya distance is calculated by:

$$BD = \sqrt{1 - \sum_{b=1}^{N} \sqrt{p_b q_b}}\,, \tag{8}$$

where, BD denotes Bhattacharyya distance; p and q are the two normalized color histograms; N is number of bin in histogram.

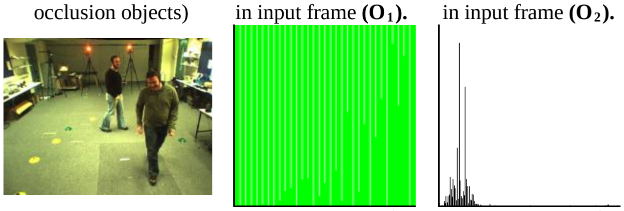


a) color image b) color histogram

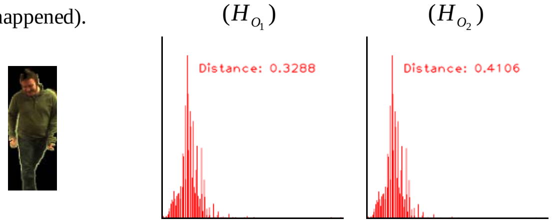
Fig. 8. An example of color histogram.

Let $O^k = \{O_1^k, O_2^k, ..., O_m^k\}$ is the occluded objects at frame k and $U^n = \{U_1^n, U_2^n, ..., U_m^n\}$ denotes the existing non-occluded objects in previous frame n (the frame before occlusion is found). The color histogram of each occluded objects and existing non-occluded objects are $H_{O_i^k}$ and $H_{U_j^n}$, respectively. In this paper, color histograms are created from hue component of HSV color space. Fig. 8 shows an example of the color histogram.

a) input frame (having occlusion objects) b) segmented obj. 1 in input frame **(O_1).** c) segmented obj. 2 in input frame **(O_2).**



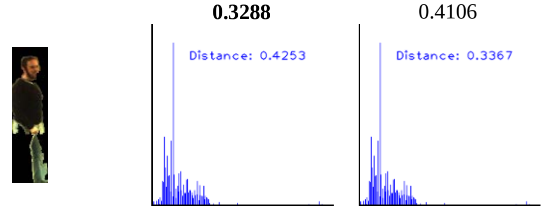
d) Previous frame (before occlusion happened). b1) Histogram of object O_1 in b) (H_{O_1}) c1) Histogram of object O_2 in c) (H_{O_2})



e) object 1 before occlusion happened e1) Histogram of object U_1 in e) (H_{U_1})

distance between distance between

(U_1) H_{O_1} and H_{U_1} is **0.3288** H_{O_2} and H_{U_1} is 0.4106



f) object 2 before occlusion happened **(U_2)** f1) Histogram of object U_2 in f) (H_{U_2})

distance between H_{O_1} and H_{U_2} is 0.4253 distance between H_{O_2} and H_{U2} is **0.3367**

Fig. 9. Bhattacharyya distance between two histograms.

For each occluded object O_i^k, we calculate the Bhattacharyya distance $BD(O_i^k, U_j^n)$ between color histogram of this object and color histogram of every object U_j^n in U^n and then find the shortest distance $BD_{\min}$. The Bhattacharyya distance $BD(O_i^k, U_j^n)$ is computed according to Eq.(8), specially:

$$BD(O_i^k, U_j^n) = \sqrt{1 - \sum_{b=1}^{N} \sqrt{H_{O_i^k}(b) H_{U_j^n}(b)}}\,, \tag{9}$$

The occluded object O_i^k will update its ID according to U_j^n if $BD(O_i^k, U_j^n) = BD_{\min}$. Fig. 9 shows the numeric results of calculating Bhattacharyya distance between two histograms.

Fig. 10 illustrates an example of tracking occluded objects in different depth layers.

a) input frame (having occlusion objects) b) output (tracking occlusion objects in different layers)

Fig. 10. Tracking occluded objects in different depth layers.

2.5.2.3. Tracking occluded objects in one depth layer

Fig. 11 shows an example of occluded objects in similar depth layer.

a) input frame (having occlusion objects) c) occlusion region (depth image) d) projected blobs in XZ plane (occlusion objects in one depth layer)

Fig. 11. An example of tracking occluded objects in one depth layer.

When occlusion objects have similar depth range or full occlusion, it is difficult to segment and track multiple objects as above technique. To deal with this problem, we propose the tracking method based on camshift (Continuously Adaptive Mean shift) algorithm [25] as following part.

Assuming that there are m occluded object O_i^k $(i = 1,2,...,m)$ in the occlusion region R_{occ}^k, their existing corresponding tracks in the previous frame are $U_j^{(k-1)}$. Our algorithm has following steps:

(i) Pre-computing the color histogram H_U for every existing track U_j^n, and the color histogram H_R for occlusion region. Here we calculate a hue histogram from HSV color space.

(ii) Based on the average depth values, sorting the list of object $U_j^{(k-1)}$ so that the object with biggest depth $U_{foremost}^{(k-1)}$ (i.e. the shortest distance from camera) goes first.

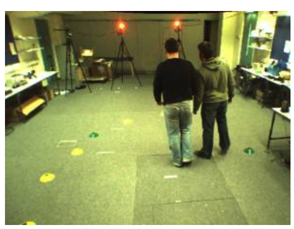
(iii) Calculating a back projection of a hue plane of occlusion region using the pre-computing histogram of $U_{foremost}^{(k-1)}$. Based on the back projection image, finding in R_{occ}^k the corresponding object of $U_{foremost}^{(k-1)}$ using camshift algorithm. Label it as $O_{foremost}^k$ with ID according to the ID of $U_{foremost}^{(k-1)}$.

Fig. 12 shows an example of projection image.

(iv) Removing the $O_{foremost}^k$ in R_{occ}^k. Selecting the next track in the sorted $U_j^{(k-1)}$ list and running step (iii) to find the next O_j^k in R_{occ}^k.

(v) Repeating step (iv) until all object in O^k in R_{occ}^k finds their corresponding track.

a) Previous frame (before occlusion happened).



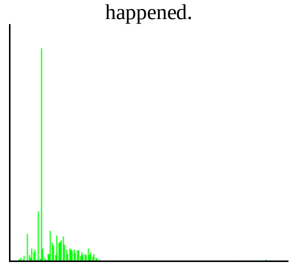
d) input frame (having occlusion objects)

b1) object 1 with bigger depth

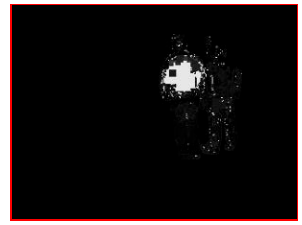
b2) Object 2

e) occluded objects in input frame

b) Two objects before occlusion happened.



c) Histogram of object 1 in b1).

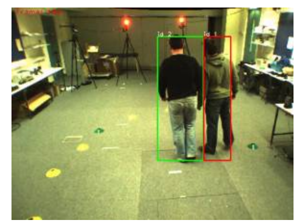


f) Projection image of occluded region in input frame using pre-computing histogram in c).

Fig. 12. An example of projection image.

Fig. 13 demonstrates a result of tracking occluded objects in one depth layer.

a) input frame (having occlusion objects)



b) output (tracking occlusion objects in different layers)

Fig. 13. An example of tracking occluded objects in one depth layer.

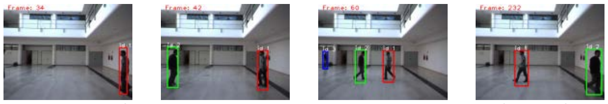
In practice, when projecting partly occluded objects or fully occluded objects with similar depth ranges into XZ plane, we will obtain only one blob/region in XZ plane. In the case, the objects in similar depth range are partly occluded, the above algorithm will work well (see an example in Fig. 13). However, the fully occluded objects current frame t will reappear as partial occluded objects bind their occluder in the later frame, so it will be tracked by our method (see example in Fig. 17).

3. Experimental Results

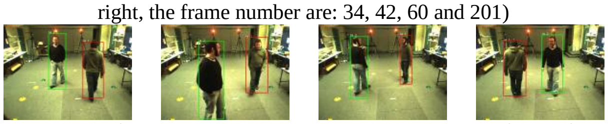
In this section, we show the experimental results to evaluate the proposed tracking method. We evaluate the tracking performance by the capability of detecting and maintaining constant ID of foreground objects during the occlusion and after the occlusion over.

The proposed tracking method has been test on some video sequences. The input of out method is a pair of video sequence and the output is the left video sequence in which has a set of moving objects labeling with ID and bounding boxes with different color.

Fig. 14 shows results of tracking non-occlusion objects. In the example 14a), in the three frames (from left to right) each object appears one after another and in the fourth frame, object with $ID = 3$ has gone out the scene. These results illustrate the ability of our algorithm in detecting object, assigning and maintaining the objects ID .



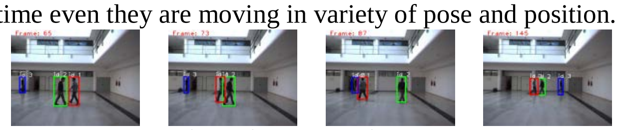
a) tracking of non-occluded objects in “Gym sequence” (from left to right, the frame number are: 34, 42, 60 and 201)



b) tracking of non-occluded objects in “Room sequence” (from left to right, the frame number are: 0, 19, 113 and 250, respectively.

Fig. 14. Result of tracking non-occluded objects.

We demonstrate the result of tracking of occluded objects in different depth layers in Fig. 15. Our proposed method can successful detect the object under partial occlusion. All of objects have constantly label over the time even they are moving in variety of pose and position.



a) “Gym sequence”: from left to right, the frames indexes are 65, 73, 87 and 145, respectively.

a) “Room sequence”: from left to right, the frame indexes are 127,129, 132 and 135, respectively.

Fig. 15. Tracking occluded objects in different depth layers.

Tracking the occluded object under the similar depth layer is shown in Fig. 16. These examples show that the proposed algorithm can detect and track a partial occluded object that is equivalent to at least one half a human body.

a) “Gym sequence”: from left to right, the frames indexes are 206, 207, 208 and 209, respectively.

a) “Room sequence”: from left to right, the frame indexes are 236,237, 238 and 239, respectively.

Fig. 16. Tracking occluded objects in one depth layer.

Fig. 17 shows the case an object is fully occluded and in the similar depth layer with its occluder and our system cannot detect it. However, in the future time when this object reappears as partial occluded object, the system can detect and maintain its ID. This example demonstrate the capability of proposed algorithm in term of maintain constant label for object over the video sequence.

“Room sequence”: from left to right, the frame indexes are 67,72, 73 and 74, respectively.

Fig. 17. Tracking fully occluded objects.

4. Conclusions

In this paper, we have presented a novel tracking method aiming at detecting objects and maintaining their ID over the time. The main key factor of this method is to use depth information to help to track objects under various occlusion scenarios. Different object tracking strategies are apply according to occlusion situation including finding correspondence object based on Bhattacharyya distance between two histograms and using camshift based algorithm with the help of object depth ordering. The presented experimental results have confirmed the capability of our proposed objects tracking algorithm under the most typical and challenging occlusion scenarios.

However, the proposed algorithm can work only in an indoor or medium sized environment since the reliability of depth information diminished in proportion to the distance from camera and only when the moving velocity of objects are slow. In the future work, to construct a robust moving object tracking system in both indoor and outdoor environment, we will study to use more object’s features to classify and track the objects.

References

[1] D. Koller, K. Danilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, pp. 257-281, 1993.

[2] T. Hai, H. S. Sawhney, and R. Kumar, "Object tracking with Bayesian estimation of dynamic layer representations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 75-89, 2002.

[3] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1296-1311, 2003.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564-577, 2003.

[5] J. Shi and C. Tomasi, "Good features to track," in *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593-600.

[6] Y. Rui and Y. Chen, "Better Proposal Distributions: Object

Tracking Using Unscented Particle Filter," *in Proceedings of the Computer Vision and Pattern Recognition, 2001 (CVPR 2001)*, pp. 786-793.
[7] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision,* vol. 29, pp. 5-28, 1998.
[8] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1997, pp. 495-501.
[9] N. K. Paragios and R. Deriche, "A PDE-based level-set approach for detection and tracking of moving objects," in the *Sixth International Conference on Computer Vision,* 1998, pp. 1139-1145.
[10] Z. Tao, R. Nevatia, and W. Bo, "Segmentation and Tracking of Multiple Humans in Crowded Environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, pp. 1198-1211, 2008.
[11] X. Zhou, Y. F. Li, and B. He, "Game-Theoretical Occlusion Handling for Multi-Target Visual Tracking," *Pattern Recognition,* vol. 46, pp. 2670-2684, 2013.
[12] V. Papadourakis and A. Argyros, "Multiple objects tracking in the presence of long-term occlusions," *Computer Vision and Image Understanding,* vol. 114, pp. 835-846, 2010.
[13] M. Wu, X. Peng, Q. Zhang, and R. Zhao, "Segmenting and tracking multiple objects under occlusion using multi-label graph cut," *Computers & Electrical Engineering,* vol. 36, pp. 927-934, 2010.
[14] E. Parvizi and Q. M. J. Wu, "Multiple Object Tracking Based on Adaptive Depth Segmentation," in *Canadian Conference on Computer and Robot Vision,* 2008 *(CRV '08)*, pp. 273-277.
[15] S. J. Krotosky and M. M. Trivedi, "On Color-, Infrared-, and Multimodal-Stereo Approaches to Pedestrian Detection," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, pp. 619-629, 2007.
[16] R. Okada, Y. Shirai, and J. Miura, "Object tracking based on optical flow and depth," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems (IEEE/SICE/RSJ)*, 1996, pp. 565-571.
[17] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International Journal of Computer Vision,* vol. 47, pp. 7-42, 2002.
[18] K. Konolige, "Small Vision Systems: Hardware and Implementation," in *Robotics Research*, Y. Shirai and S. Hirose, Eds., ed: Springer London, 1998, pp. 203-212.
[19] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition,* 2004 *(ICPR 2004)*, vol.2, pp. 28-31.
[20] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection," in *Video-Based Surveillance Systems*, P. Remagnino, G. Jones, N. Paragios, and C. Regazzoni, Eds., ed: Springer US, 2002, pp. 135-144.
[21] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 25, pp. 918-923, 2003.
[22] *OpenCV Wiki, cvBlobLib,* . Available: http://opencv.willowgarage.com/wiki/cvBlobsLib
[23] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding,* vol. 93, pp. 206-220, 2004.
[24] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society,* vol. 35, pp. 99-109, 1943.
[25] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *the Fourth IEEE Workshop on Applications of Computer Vision,* 1998 *(WACV '98)*, pp. 214-219.

Anh Tu Tran is a PhD candidate of the Graduate School of Engineering at Hiroshima University. He received the B.E in 2001 from Hanoi University of Technology, Vietnam, and received the ME in 2007 from RWTH Aachen University, Germany. His current research is mainly in area of computer vision and image processing

Koichi Harada is a professor of the Graduate School of Engineering at Hiroshima University. He received the BE in 1973 from Hiroshima University, and MS and Ph.D in 1975 and 1978, respectively, from Tokyo Institute of Technology. His current research is mainly in the area of computer graphics. Special interests include man-machine interface through graphics; 3D data input techniques, data conversion between 2D and 3D geometry, effective interactive usage of curved surfaces. He is a member of ACM, IPS of Japan, and IEICE of Japan.