Fast Polygons Fusion for Multi-Views Moving Object Detection from Overlapping Cameras

Mikaël A. Mousse[1,2], Cina Motamed[1] and Eugène C. Ezin[2]

[1]Laboratoire d'Informatique Signal et Image de la Côte d'Opale
Université du Littoral Côte d'Opale, France
E-mail : {mousse, motamed}@lisic.univ-littoral.fr
[2]Unité de Recherche en Informatique et Sciences Appliquées
Institut de Mathématiques et de Sciences Physiques, Bénin
E-mail : eugene.ezin@imsp-uac.org

RÉSUMÉ. Dans cet article, nous proposons un algorithme de fusion rapide de polygones pour la détection d'objets mobiles par le biais dans un réseau de caméras. Après la détection des pixels de premier plan de chaque camera, nous approximons les contours détectés par des polygones. Ces polygones sont projetées dans le plan de référence. Après cela nous proposons une approche de fusion efficiente dans le but d'obtenir une détection multi caméras. Les Différents résultats sur un jeu de données publique sont présentés et analysés. La détection des objets mobiles à travers la vue de chaque caméra est obtenue en utilisant un algorithme basé sur le codebook.

ABSTRACT. In this paper, we propose a fast polygons fusion algorithm to address the problem of moving object detection from overlapping cameras. Once the foreground pixels are detected in each view, we approximate their contours with polygons and project them into the reference plane. After this, we propose an efficient fusion approach to fuse polygons in order to obtain a multi-views foreground area. The different results on open video dataset are presented and analyzed. Each foreground information is obtained by using a codebook based moving object detection algorithm.

MOTS-CLÉS : Détection d'objet, Codebook, Caméras avec vues chevauchant, Fusion d'informations

KEYWORDS : Motion detection, Codebook, Overlapping camera, Information fusion

1. Introduction

In computer vision community, the use of multi-camera takes a lot of scope. Indeed, motivations are multiple and concern various domains as the monitoring and surveillance of significant protected sites, the control and estimation of flows (car parks, airports, ports, and motorways). Because of the fast evolution in the fields of data processing, communications and instrumentation, such applications become possible. These kind of systems require more cameras to cover overall field-of-view. They reduce the effects of objects dynamic occlusion and improve the accuracy of estimation of foreground zone.

According to Xu et al., existing multi-camera surveillance systems may be classified into three categories [6]. The system in the first category fuses low-level information. In this category, multi camera surveillance systems detect and/or track in a single camera view. They switch to another camera when the systems predict that the current camera will not have a good view. In the second category, system extracts features and/or even tracks targets in each individual camera. After this, we integrate all features and tracks in order to obtain the global estimates.These systems are of intermediate-level information fusion. The system in the third category fuses high-level information. In these systems, individual cameras don't extract features but provide foreground bitmap information to the fusion center. Detection and/or tracking are performed by a fusion center [1, 2, 3, 6, 7]. This paper will focus on the approaches in the third category. In this category some algorithms have been proposed. Authors in [2] proposed to use a planar homographic occupancy constraint to combine foreground likelihood images from different views in order to resolve occlusions and to determine regions on the ground plane that are occupied by people. In [3], authors extended the ground plane to a set of planes parallel to it, but at some heights off the ground plane to reduce false positives and missed detections. The foreground intensity bitmaps from each individual camera are warped to the reference image by authors in [1] and the set of scene planes are at the height of people heads. The head tops are detected by applying intensity correlation to aligned frames from the different cameras. This work is able to handle highly crowded scenes. Yang et al. detect objects by finding visual hulls of the binary foreground images from multiple cameras [7]. These methods fully utilize the visual cues from multiple cameras and are robust in coping with occlusion. However the pixel-wise homographic transformation at image level slows down the processing speed. In order to overcome this drawback, Xu et al. proposed an object detection approach via homography mapping of foreground polygons from multiple camera [6]. They approximate the contour of each foreground region with a polygon and only transmit and project the vertices of the polygons. The foreground regions are detected by using Gaussian mixture model. After the projection of the polygons vertices, they rebuilt each foreground map in the reference image by considering as foreground pixels all pixels lying inside a polygon. The multi-view object detection is obtained by considering the pixels which have been detected to be a foreground pixels in n different polygons (n is the number of cameras). This method provides good results [6]. In [5], authors also propose an algorithm based on polygons fusion for moving object extraction.

In this work, we propose a new strategy based on polygons which reduces the complexity of polygons fusion. Indeed a major challenge in computer vision is to get a real time system. Then it is important to reduce the complexity of each part of a computer vision system. This paper consists of four sections. The second section presents the po-

lygons fusion approach. The third section presents experimental results. Conclusion and future works are presented in section four.

2. Polygons Fusion Approach

In this section, we present our fusion approach for moving object detection in a multi camera system. The goal of our algorithm is to extract the relevant vertices of the polygons associated with the various objects in each view.

Let us consider a scene observed by n ($n \geq 2$) cameras with overlapping views. The multi-view moving object detection in the ground plane is the intersection of the single views foreground polygons projection. Then using our approach, we automatically detect the vertices of the polygons resulting from intersections. In our approach, the multi-view foreground map is represented by a codebook $\mathcal{D} = \{d_1, d_2, .., d_L\}$ and each codewords d_i, $i = 1, ..., L$ represents a polygons resulting from intersection. Each codewords consists of two vectors. The first vector $index_i$ contains the identifiers of the polygons which form the intersection whereas the second $content_i$ contains the vertices of the polygon resulting from intersection. In this part, we call vector a sequence container representing arrays that can change in size.

Firstly, we consider two camera views and we project the vertices of their polygons in the ground plane by using the principles of the homography. So if $\mathcal{V}_1 = (v_{11}, v_{21}, ..., v_{k1})$ is original polygon in the single view plan, the projected polygons becomes $\mathcal{V}_1 = (v'_{11}, v'_{21}, ..., v'_{k1})$ with $v'_{11}, v'_{21}, ..., v'_{k1}$ which are respectively the projection of $v_{11}, v_{21}, ..., v_{k1}$. Among the two views, we select one view and we seek its projected points which belong to a projected polygon from the view of the second camera. When we find a point which verify this condition, we compare it to the current codebook to determine which codeword d_m (if any) it matches (m is the matching codeword's index). To determine which codeword will be the best match, we create a vector from different polygons identifiers (belonging polygon identifier, origin polygon identifier) and compare it to the first vector of each codewords. Two vectors will be considered as equivalent if all the elements in one of the vector is necessarily in the second. If there is no match, we create a new codeword d_k by setting $index_k$ to the vector issues from different polygons identifiers (belonging polygon identifier, origin polygon identifier) and creating $content_k$ in which we insert the point. This part is resume in Algorithm 1. In this algorithm the two views V_1, V_2 are considered and we select the view V_2 in order to seek its projected points which belong to a projected polygon from the view of the second camera.

After that in each codeword d_i, we have $index_i$ which contains the information about polygons which form the intersection and $content_i$ which contains one point of the intersection. From this point we rebuild the intersection. For that we update the codeword d_i. We rewrite the projected polygon to which the point belongs by taking this vertex as the first point of the polygon. For example, if $\mathcal{V}_1 = (v'_{11}, v'_{21}, ..., v'_{k1})$ represents the projected polygon and v'_{31} is the point then the rewriting gives $\mathcal{V}_1 = (v'_{31}, v'_{41}, ..., v'_{k1}, v'_{11}, v'_{21})$. By using this polygon, we check from the first segment if in the ground plane a segment has

1. $inCod(index_n, \langle id_k, id_i \rangle)$ returns true if $index_n$ contains only id_k and id_i.

Algorithm 1: Codebook initialization

$L \leftarrow 0$ ($\leftarrow$ means assignment), $\mathcal{D} \leftarrow \emptyset$ (empty set)
for *each projected polygons* id_i *of the view* V_2 **do**
  for *each each vertex* $v_{j\_id_i}$ *of the polygon* id_i **do**
    if $v_{j\_id_i}$ *is inside the projected polygon* id_k *of the view* V_1 **then**
      Find the codeword d_m in $\mathcal{D} = \{d_n | 1 \leq n \leq L\}$ matching to $v_{j\_id_i}$ based on condition (a)
        (a) inCod [1]$(index_n, \langle id_k, id_i \rangle) = true$
    if $\mathcal{V} = \emptyset$ *or there is no match* **then**
      $L \leftarrow L + 1$
      create codeword d_L by setting parameter $index_L \leftarrow (id_k, id_i)$ and $content_L \leftarrow (v_{j\_id_i})$

an intersection with any segment of the second polygon of the codeword. If we don't find an intersection then we update codeword by adding at the end of the vector $content_i$ the point at the second end of said segment and the initial polygon is considered as default polygon during this part of the process. But if we find an intersection, we add two points at the end of the vector $content_i$: the first point is the intersection and the second is the point of the segment from the second polygon which belongs to the first polygon of codeword. In this case, the second polygon becomes default polygon. We repeat the search for intersection between segments from two different polygons by using the default polygon segment that comes from the last point which is inserted into $content_i$ until obtaining the first point of the codeword. We take care to avoid to include this point again. After realizing these instructions on each codeword, our codebook contains information about the polygons that form intersections using the two chosen views and the vertices of polygons representing these intersections. This part is resume in Algorithm 2.

For each of the remaining cameras (if any remain), we rebuild the codebook. In order to perform it, we consider the contents of the vector $content$ of each codeword of the immediately previous codebook as the vertices of a polygon and the concatenation of the contents of the vector $index$ of the codeword as the identifier of this polygon. All polygons from this codebook are considered as part of an imaginary camera view. And we use the process for codebook building for two different views (process which is explained in previous algorithms (Algorithm 2 and Algorithm 3)) to build the new codebook by using our imaginary camera view and the new input camera view.

Using this method we obtain automatically the vertices of the polygons resulting from intersection. The multi view moving objects detection are then obtained by set as foreground the pixels which are inside these polygons. The ray casting algorithm proposed by Sutherland et al. [8] has been used in order to resolve point-in-polygon problem.

1. default segment is the segment which is obtained by considering in default_polygon, default_point and the vertex that follows its.

Algorithm 2: Extraction of intersection vertices

for *each codeword* d_n *(with* $index_n = (id_k, id_i)$ *and* $content_n = (v_{j\_id_i})$*) of codebook* $\mathcal{D}$ **do**
  Rewrite the projected polygon id_i by taking $v_{j\_id_i}$ as the first point of the polygon.
  default_point $\leftarrow v_{j\_id_i}$, default_polygon_id $\leftarrow id_i$.
  repeat
    if *the default segment*[2] *has an intersection with an other segment from a second polygon forming* c_i **then**
      default_polygon_id $\leftarrow \overline{default\_polygon\_id}$ (identifier of the second polygon which forming c_i).
      intersect_point $\leftarrow$ intersection of the two segments.
      default_point $\leftarrow$ point of the segment from the default_polygon_id which belongs to $\overline{default\_polygon\_id}$.
      if *intersect_point* $\neq v_{j\_id_i}$ **then**
        update d_n by inserting intersect_point at the end of $content_n$.
      if *default_point* $\neq v_{j\_id_i}$ *and default_point* $\neq$ *intersect* **then**
        update d_n by inserting default_point at the end of $content_n$.
    else
      default_point $\leftarrow$ point at the second end of said segment.
      if *default_point* $\neq v_{j\_id_i}$ **then**
        update d_n by inserting default_point at the end of $content_n$.
  until *default_point* $= v_{j\_id_i}$

3. Experimental Results and Performance Evaluation

In this section, we present the performance of the proposed approach. Firstly we present the experimental environment and results. After that we present and analyze the performance of our system.

3.1. Experimental Results

For the validation of our algorithm, we tested it on video sequence that contains significant lighting variation, dynamic occlusion and scene activity. Both qualitative and quantitative evaluations have been carried out by using the PETS'2001 dataset[3]. We selected sequence "**Dataset 1**" which are also used in other researches works. The size of each frame is (768×576). The experiment environment is Intel® Core i7 CPU L 640 @ $2.13GHz \times 4$ processor with 4GB memory and the programming language is C++.

During our experiment, we use foreground pixels detection algorithm presented in Mousse et al. [4] for each single view foreground pixels extraction The foreground polygons is obtained by finding the convex hull of the foreground pixels. Each region can be approximated by a polygon. The polygon is obtained by finding the convex hull of all contours detected in threshold image. The convex hull or convex envelope of a set X of points in the Euclidean plane or Euclidean space is the smallest convex set that contains X. For instance, when X is a bounded subset of the plane, the convex hull may be vi-

3. Available online at `http://www.cvg.reading.ac.uk/PETS2001/pets2001-dataset.html`

sualized as the shape enclosed by a rubber band stretched around X. Some segmentation results are presented in Figure 1.

3.2. Performance Evaluation and Discussion

Xu et al. demonstrated the efficiency of the use of single views polygons and of their intersections in a ground plane for multi-view objects detection [6]. Our experiment results also confirm that the polygon projection results are very close to those from the bitmap projection. Due to this, we only evaluate the performance of our system by using the processing time as metric. Xu et al. proved that their algorithm faster than the existing algorithms [6]. So the discussion about the processing time of our proposed algorithm is done by comparing its with the processing time of Xu et al.'s algorithm. Then, the overall

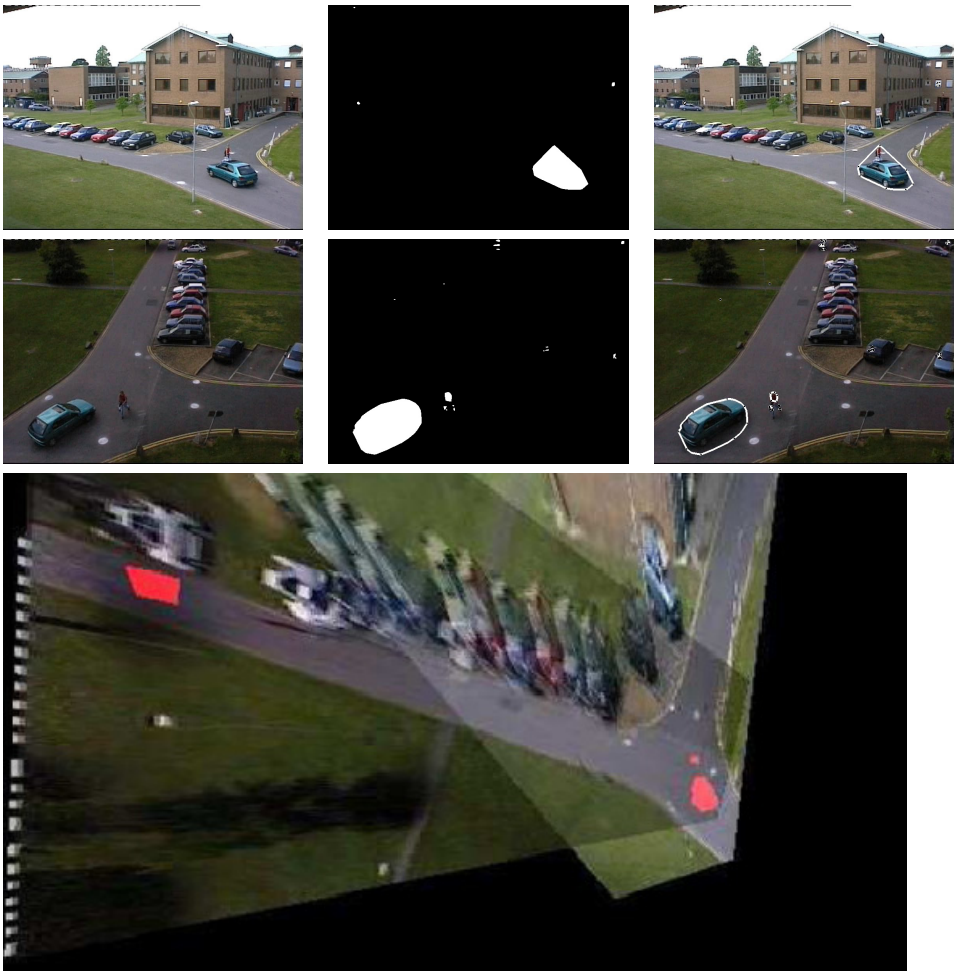

Figure 1 : The first two rows show each camera views. In these rows, the first column presents the original frame, the second column shows the foreground maps in single view and the third column presents a foreground approximation with polygons The third row shows the segmentation result using a multi-view informations.

Tableau 1 : Global performance evaluation.

Score	Xu et al Algorithm [6]	Proposed algorithm
Processing times (f/s)	65.82	**73.97**

execution time of the two algorithms. It is expressed in frames per second. Regarding the comparison of overall performance, the obtained values are reported in Table 1. According to these values we can conclude that our proposed algorithm is faster than algorithm suggested by Xu et al. The difference between the two execution times will increase when the number of cameras will increase and/or the number of objects observed by several cameras will become much larger. In fact with more cameras and/or more objects we will obtain more polygons. The complexity of the fusion process strongly depends on the number of cameras and/or the number of foreground objects.

4. Conclusion

In this work, we have proposed a fast algorithm for object detection by using overlapping cameras. In each camera, we use an improvement of codebook based algorithm to get foreground pixels. The single moving object detection algorithm integrates superpixels segmentation in original codebook and extends its on pixel level. In order to obtain the multi-view moving object detection, we propose a fusion approach which enables to determine quickly the polygons resulting from intersection of single views polygons. The experiment results have shown that the use of our fusion method reduces the computational complexity of multi-view moving object detection.

5. Bibliographie

Eshel, R., Moses, Y. : Homography based multiple camera detection and tracking of people in a dense crowd. *18th IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

Khan, S.M., Shah, M. : A multi-view approach to tracking people in crowded scenes using a planar homography constraint. *9th European Conference on Computer Vision*, 2006.

Khan, S.M., Shah, M. : Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31*, 2009.

Mousse, M.A. , Ezin, E.C. , Motamed, C. : Foreground-background segmentation based on codebook and edge detector. *10th International Conference on Signal Image Technology & Internet Based Systems*, 2014.

Mousse, M.A., Motamed, C., Ezin, E.C. : Fast moving object detection from overlapping cameras. *International Conference on Informatics in Control, Automation and Robotics*, 2015.

Xu, M., Ren, J., Chen, D., Smith, J., Wang, G. : Real-time detection via homography mapping of foreground polygons from multiple cameras. 18th IEEE International Conference on Image Processing, 2011.

Yang, D.B., Gonzalez-Banos, H.H., Guibas, L.J. : Counting people in crowds with a real-time network of simple image sensors. *9th IEEE International Conference on Computer Vision*, 2003.

Sutherland, I.E., Sproull, R.F., Schumacker, R. A. : A characterization of ten hidden surface algorithms. *ACM Computing Surveys (CSUR)*, 1974.