Detecting and Tracking (People) Motion

Giovanni De Toni (197814)[1]

University of Trento, Italy, giovanni.detoni@studenti.unitn.it

Abstract. Detecting and tracking motion is one of the most common computer vision's tasks. Despite its apparent simplicity, this topic presents several tough spots and unexpected challenges which can be difficult to overcome. In our work, we describe a simple and fast region-based approach to perform motion detection and subsequent object tracking with a good grade of accuracy. Successively, a more advanced model based on the Kalman Filter is proposed in order to overcome some limitation of the first procedure. These solutions are then applied to a pedestrian tracking task. Finally, we discuss the performance and the shortcomings of these two methods.

Keywords: Tracking · Detection · Mixture of Gaussians · Background Subtraction · Kalman Filter · OpenCV

1 Introduction

Humans are really good at detecting and tracking objects while they move in a scene. However, this operation becomes much more complicated when achieved through a computer video system. Despite the difficulties to automatize such procedure, recognize and follow passing objects by just looking at video recordings is an incredibly useful feature for many real-world applications (e.g. surveillance systems, autonomous vehicles, crowds monitoring etc). Here we present a simple tracking system to detect and count moving objects, given a video recording, which is both computationally inexpensive and sufficiently accurate. We devised a procedure in order to detect motion in a reliable way, by applying background subtraction and shadow removal techniques. We then defined two methods to do dynamic tracking of detected targets by means of a region-based approach and of a more sophisticated Kalman-aided solution. Finally, we run some experiments on a computer-generated video recording of pedestrians moving in a scene. We registered for each video frame the number of pedestrians present in the scene and we also recorded the trajectories of three selected pedestrians. These values were confronted with the ground truth information used to generate the video in order to measure the accuracy of our architecture. The paper is structured in 4 main sections: Section 2 describes the conducted literature investigation displaying some related works on these topics. Section 3 outlines in detail the chosen methods used to solve the issue, from the objects detection (Mixture of Gaussians) to the tracking (region-based and Kalman-aided approaches). Section 4 describes the experimental settings and the results of the analyses done on our

implementation with the test video. Finally, Section 5 elaborates the results of the experiments and we try to reach some final conclusions and possible future improvements.

2 Related Works

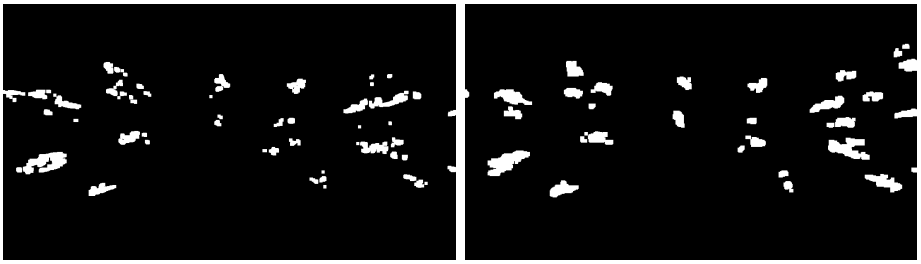
As the first step, we searched the literature in order to document ourselves about suitable methods and techniques that are usually applied to accomplish these tasks. We started from the motion detection. Several papers regarding background subtraction were published in previous years. Simple techniques such as frame differencing and adaptive background subtraction can be used if the scene presents fixed illumination and no background noise. Usually, Mixture of Gaussians (MOG) models are preferred, since they can to model complex, non-static backgrounds [11]. Because of this, MOG models went under extensive studies in order to increase their capabilities (for instance, [15] shows how to extend them in order to do shadows detection). However, MOG still presents several shortcomings (sensible to the adaptation rate, performance depends on the number of Gaussians employed etc.) and therefore other more advanced techniques were devised [6,8,13]. Shadows detection and removal was also a topic of several papers in which different techniques were proposed. For instance, HSV colour space conversion [4] or chromaticity and gradient correlation [9]. In order to do object (people) tracking, many different methods are employed. From using Neural Networks to detect the human shape (or to generate crowd count from images) [12,14] to feature-based or region-based approaches [3]. Researchers tried also to fuse together multiple algorithms in order to achieve stronger and more reliable predictors [10]. Finally, some papers discussed also how to recover from partial or total occlusions [7], which represents one the biggest problem of object tracking.

3 Methods

The detection/tracking process was divided into several steps, each of them dealing with a specific issue/task, in order to deliver a modular infrastrcture. The processing pipeline phases are the following:

1. Shadow Removal and Background Subtraction;
2. Extraction of contours and associated histograms;
3. Object Tracking by using a Region-Based approach and a Kalman-aided approach;

The complete implementation was done using C++ and the OpenCV library (release 2.4.13) [5].



(a) Background subtraction without HSV shadow removal.

(b) Background subtraction with HSV shadow removal.

Fig. 1. Figure (a) shows the result of the background subtraction without the HSV conversion. We clearly see that the image is more noisy and the blob detection is less precise than Figure (b).

3.1 Background Subtraction

For each frame of the given video, the motion detection was performed using a Mixture of Gaussians technique described in the paper by Zivkovic [11] in order to do background subtraction (this is implemented as `BackgroundSubtractorMOG2` [1] inside OpenCV). This technique was used in order to provide a more robust and precise detector. In fact, it can easily absorb eventual sudden changes in the environment (for instance, illumination variations). The result of this procedure is a greyscale image, in which the moving areas are displayed as white "blobs" (Figure 2 shows the result of the entire processing pipeline on a video frame).

In order to avoid wrong motion detection caused by shadows cast by moving objects, each frame was also first converted from the RGB to the HSV space. Then, the hue (H) channel was extracted and the other components were discarded. This enabled us to remove eventual luminance variations which could have caused false positives. This conversion proved itself to be most useful because it improved the quality of the motion detection. Figure 1 compares the result of this procedure when applying or not the HSV conversion. The MOG method used is also able to detect shadows, which are highlighted in a different colour (a light grey). These were also removed by applying a binary thresholding. This shadow removal procedure generated a binary image in which the blobs were represented as white areas. Finally, a sequence of opening and dilating operators with a rectangular shaped structuring element was then applied in order to remove the eventual noise and, more generally, to improve the blobs shapes.

3.2 Contours Extaction

As a second step, we applied a procedure to extract the contours from the binary frame containing the previously detected blobs (we used the OpenCV `findContours` [2] method). We filtered also the contours in order to reduce eventual noise by selecting only those which had an area above an empirical-measured

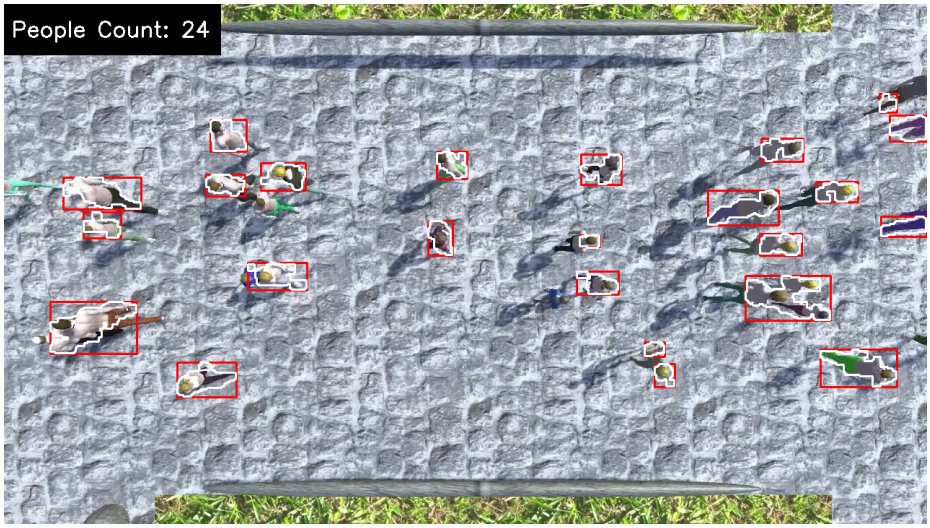


Fig. 2. This figure shows the results of the motion detection algorithm on a single frame of the video. Here the found contours (white) and bounding boxes (red) were applied to the original frame. On the top left corner, we can see the people counter.

threshold. The threshold was selected in order to maximize the number of pedestrians found and to minimize the false positive (for instance, the detection of small parts of these pedestrians). For each of these filtered contours, we computed also their histogram signature and we generated also their corresponding bounding boxes (for visualization purposes only).

As an additional output of the procedure, we also recorded and provided the number of blobs detected for each frame of the video.

3.3 Region-Based Object Tracking

The object tracking was done using a simple region-based approach. The procedure is the following. We first defined three sets: $C(t)$, the set of contours detected at frame t through MOG, $O(t)$, the set of current tracked objects at frame t, and $D(t)$, the set of disappeared objects at frame t.

At $t = 1$, we set $O(t) = C(t)$. This means that all the newly detected contours are added to the set of objects we need to keep track of.

At frame $t > 1$, the position of objects inside $O(t) \backslash D(t)$ is updated by searching inside $C(t + 1)$ a contour that closely resembles one of the tracked objects o. The association between an object $o \in O(t) \backslash D(t)$ and a contour $c \in C(t + 1)$ is based on a proximity basis and by also looking at the histogram signatures. More specifically, we check if the Euclidean distance between the centroids of o and c is below a certain threshold and again if the correlation coefficient of their histograms is above a certain threshold. The thresholds were chosen empirically

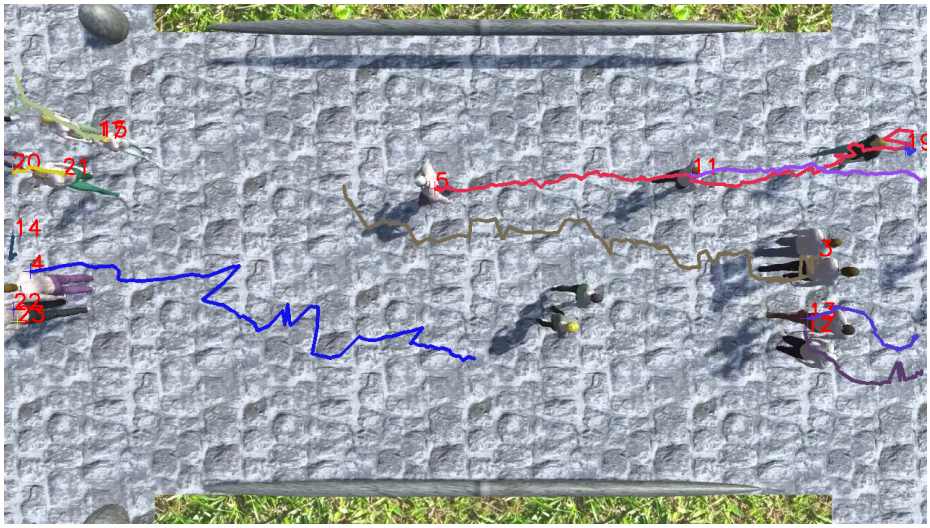


Fig. 3. This figure shows the results of the Kalman-aided tracking algorithm on a single frame of the video. Here the trajectories and the ids of the tracked object are displayed on the original frame. We can already see some issues with the tracking procedure. Two pedestrians in the middle of the scene are not tracked and the pedestrians with id 3 are considered just as one entity (because of merging issues).

by taking the values which held the best experimental results. It is important to note that a contour c is assigned just to the closest object o (best matching). After each association, the centroid and histogram of o are updated with the values of its matched c (this is especially useful to account for small colour variation of the new contour).

If we were not able to find a matching object o for a contour c, it means that c could be a new trackable object entering the scene. To account for possible wrong detection, we start to track a new contour $c \in C(t+1)$ only if it is closer to one of the borders of the scene frame (again, we check that the Euclidean distance between the c's centroid and the frame border is lower than a threshold). If a contour c appears near one of the frame's borders, we are reasonably sure that it is a new object entering the scene. Therefore, we add c to $O(t+1)$.

Moreover, if an object $o \in O(t) \backslash D(t)$ did not found any suitable next contour $c \in C(t+1)$, it is marked as missing. If a tracked object o stays missing for more than a certain amount of frames then it is considered to have exited the scene and it is not tracked anymore (it is added to $D(t+1)$).

Algorithm 1 shows the simple region-based procedure. Note that the algorithm pseudocode presented here misses some small improvements or general logging facilities (like recording the trace for each tracked human for visualization purposes, etc.). For more detailed information please check the source code provided and its documentation.

3.4 Kalman-aided Object Tracking

Together with the simple solution described above, a more sophisticated method was tried in order to increase the quality of the tracking. The second approach consisted of using a Kalman filter (one personalized for each human detected) to predict the next position (centroid) at frame $t+1$ of all the tracked object $o \in O(t) \backslash D(t)$. The contour association procedure remained the same, but it was done by checking these newly predicted centroids, instead of the previous ones. Once an association between $o \in O(t) \backslash D(t)$ and $c \in C(t+1)$ is made, the Kalman estimate is then corrected with the new "measurement" (the centroid of c) and the object position is updated accordingly.

4 Experiments

4.1 Settings

Some experiments were conducted in order to find the best initialization parameters and thresholds such that to have optimal detection and tracking capabilities. Table 1 shows the adopted final configuration.

The system used to run the experiments was an Ubuntu 14.04 virtual machine (it was executed through the 5.2.22 version of the Virtualbox appliance), with a 1.8 GHz single processor and 2 GB of RAM. The algorithm was also configured in order to do the tracking and detection for each frame of the video.

Table 1. Final parameters for the algorithm

Parameter	Description	Value
t_h	Threshold for the histogram correlation coefficient.	0.2
t_e	Threshold for the euclidean distance difference between contours.	40
t_b	Threshold for the euclidean distance difference between the border and a contour.	20
t_d	Disappearence rate after which an object o is not tracked anymore (n. of frames).	200

4.2 Video Description

The video which we used to run our experiments was a computer-generated simulation of pedestrians entering and exiting the scene acquired from top view. The video used MPEG-4 compression format and it had a frame size of 1280x720 pixels. The original video colour space was the YUV. The size of the entire video was about 12 MB on disk with a duration of 50,876 seconds (all these information were extracted by using the `mediainfo` bash command).

All pedestrian moved in one direction only (without changing it) with almost constant velocity. The scene presented constant illumination. Apart from moving pedestrians, there was no other source of motion or noise (like trees, water, birds, etc.).

Algorithm 1 Track objects given a set C of contours.

Require: current video frame t, new detected contours C, previously detected object O, disappeared objects D

if $t == 1$ **then** // **Add all first detected contours as tracked object**
  for contour $c \in C$ **do**
    Create a new object o with a new id
    $o.position \leftarrow c.centroid$
    $o.histogram \leftarrow c.histogram$
    $o.disappeared \leftarrow 0$
    $O.push\_back(o)$
  end for
else
  for contour $c \in C$ **do** // **Try to pair each contour with a tracked object**
    $winner\_object \leftarrow null$
    for object $o \in O \backslash D$ **do**
      $o.disappeared \leftarrow o.disappeared + 1$;
      if $correlation(o.histogram, c.histogram) > t\_h$ **then**
        if $e\_distance(o.position, c.centroid) < t\_e$ **then**
          if the new object o is better than $O[winner\_object]$ **then**
            $winner\_object \leftarrow o.id$
          end if
        end if
      end if
    end for
    if $winner\_object != null$ **then**
      // **Update the features of the matched object.**
      $O[winner\_object].position \leftarrow c.position$
      $O[winner\_object].histogram \leftarrow c.histogram$
      $O[winner\_object].disappeared \leftarrow 0$
    else
      // **Add new object to track if it appeared near the borders.**
      $distance\_left \leftarrow e\_distance(c.centroid, border\_left)$
      $distance\_right \leftarrow e\_distance(c.centroid, border\_right)$
      if $distance\_left < t\_b$ or $distance\_right < t\_b$ **then**
        Create a new object o with a new id
        $o.position \leftarrow c.centroid$
        $o.histogram \leftarrow c.histogram$
        $o.disappeared \leftarrow 0$
        $O.push\_back(o)$
      end if
    end if
  end for
  for humans $o \in O \backslash D$ **do** // **Stop to track disappeared objects.**
    if $o.disappeared \geq t\_d$ **then**
      $D.push\_back(o)$
    end if
  end for
end if

4.3 Results

Note: A complete video which displays the pedestrian counter and the trajectories is available at this link: https://drive.google.com/open?id=1NgB8z0o5mlLQfVUR7G9xDpf9cTA5UjNs

Generally, the entire procedure was relatively fast and was computationally efficient (we checked both CPU usage and memory consumption). The region-based approach was slightly faster than the Kalman-aided, but we noticed in both cases a minor slowdown when there were too many pedestrians on the scene. Figure 3 shows a sample frame of the video in which we see the tracking procedure working.

We also computed the Root Mean Square Error (RMSE) for the people counting task. In the case of the tracking, we took three pedestrians (10,36 and 42) and we compared their recorded trace with the ground truth by computing for each frame the displacement of each predicted trajectory point from the real ones. We then computed the mean and the standard deviation of each trajectory.

The background subtractor (MOG+HSV Shadow Removal) was able to discover successfully most of the pedestrian crossing the scene, the RMSE of the detected pedestrian number with respect to the ground truth is around 3.94. The tracking part gave instead very different results. The region-based approach performed well only in such situations in which the pedestrians are not occluded by too many other people. If that situation arises, then the trajectory is detected wrongly (there are occlusion issues in which the contours $c \in C(t+1)$ are assigned to the wrong object $o \in O(t) \backslash D(t)$. Moreover, the region-based approach performed badly over the detection of the selected pedestrians. The Kalman-aided procedure produced instead much better. For instance, the tracking performance over the selected pedestrians outperforms the one obtained with only the region-based approach, by showing smaller displacement values and a smaller standard deviation. Table 2 shows the final results.

Table 2. Mean and standard deviation of the detected trajectories with respect to the real ones.

Pedestrian Id	Region-Based	Kalman-aided
10	24.79 $\pm$ 12.06	25.51 $\pm$ 10.78
36	613.58 $\pm$ 399.51	81.09 $\pm$ 103.69
42	301.30 $\pm$ 320.88	18.52 $\pm$ 12.45

5 Conclusions

The previous experiments showed that we can easily build a detection/tracking system which can give us modest performances without employing cutting-edge hardware or software. Moreover, the procedure is completely agnostic regarding

the objects we want to track and could be easily extended to filter out the non-wanted components (for instance, by using an SVM to discriminate if a contour c detected represents a certain "object" of interest: a human being, an animal, a car etc.) However, our method presents several pitfalls which can be hardly solved completely by simply tuning the various thresholds defined previously. For instance, the background subtraction is not able to detect all the moving pedestrians and it produces sometimes wrong detections (mainly caused by splitting and merging problems). Nevertheless, the tracking part is the one which requires the most care. One main issue of our proposed methods is that they cannot discriminate objects in presence of occlusions. When two blobs merge together it becomes difficult to distinguish between them and to predict their next position, even if we consider the histogram information. One common solution to this problem is to avoid to update the model during occlusions. The objects positions are refreshed periodically only after n frames so that the associations are done only at frame $t + n$, $t + 2n$ and so on. This will enable us to postpone the decision at a time when the blobs (hopefully) will have divided themselves. By selecting other features rather than contours' centroids and histograms we may be able to improve the accuracy of the tracking phase. Template-based approaches could also be used, since if we wanted to track only human, then we would be better of by implementing a system which detects all the moving objects which have features resembling a human (head, torso, legs, etc.). In conclusion, the results obtained are satisfactory enough for our simulated setting, but they could be greatly improved by using more sophisticated methods. Moreover, we did not try our implementation on a real-world recording, therefore we are not able to define how our implementation would behave in a real setting with a noise scene (illumination variation, noise, non-linear object movements etc.).

References

1. Backgroundsubtractormog2 class definition. `https://docs.opencv.org/ref/2.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html`, accessed: 27-11-2018
2. findcontours method definition. `https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours\#findcontours`, accessed: 27-11-2018
3. Corvee, E., Bremond, F.: Body parts detection for people tracking using trees of histogram of oriented gradient descriptors. In: 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance. pp. 469–475 (Aug 2010). https://doi.org/10.1109/AVSS.2010.51
4. Cucchiara, R., Grana, C., Piccardi, M., Prati, A.: Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In: Image Analysis and Processing, 2001. Proceedings. 11th International Conference on. pp. 360–365. IEEE (2001)
5. Itseez: Open source computer vision library. `https://github.com/itseez/opencv` (2015)
6. Kim, K., Chalidabhongse, T.H., Harwood, D., Davis, L.: Real-time foreground-background segmentation using codebook model. Real-Time Imaging **11**(3), 172

– 185 (2005). https://doi.org/https://doi.org/10.1016/j.rti.2004.12.004, http://www.sciencedirect.com/science/article/pii/S1077201405000057, special Issue on Video Object Processing
7. Lerdsudwichai, C., Abdel-Mottaleb, M., Ansari, A.N.: Tracking multiple people with recovery from partial and total occlusion. Pattern Recognition **38**(7), 1059 – 1070 (2005). https://doi.org/https://doi.org/10.1016/j.patcog.2004.11.022, http://www.sciencedirect.com/science/article/pii/S0031320305000191
8. Panda, D.K., Meher, S.: Detection of moving objects using fuzzy color difference histogram based background subtraction. IEEE Signal Processing Letters **23**(1), 45–49 (Jan 2016). https://doi.org/10.1109/LSP.2015.2498839
9. Sanin, A., Sanderson, C., Lovell, B.C.: Improved shadow removal for robust person tracking in surveillance scenarios. In: 2010 20th International Conference on Pattern Recognition. pp. 141–144 (Aug 2010). https://doi.org/10.1109/ICPR.2010.43
10. Siebel, N.T., Maybank, S.: Fusion of multiple tracking algorithms for robust people tracking. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) Computer Vision — ECCV 2002. pp. 373–387. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
11. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149). vol. 2, pp. 246–252 Vol. 2 (June 1999). https://doi.org/10.1109/CVPR.1999.784637
12. Tang, S., Andriluka, M., Andres, B., Schiele, B.: Multiple people tracking by lifted multicut and person reidentification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3539–3548 (2017)
13. Van Droogenbroeck, M., Barnich, O.: Vibe: A disruptive method for background subtraction. Background Modeling and Foreground Detection for Video Surveillance pp. 7–1 (2014)
14. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 589–597 (2016)
15. Zivkovic, Z.: 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 Vol.2 (Aug 2004). https://doi.org/10.1109/ICPR.2004.1333992