GPU Accelerated Left/Right Hand-Segmentation in First Person Vision

Alejandro Betancourt[1,2(✉)], Lucio Marcenaro[1], Emilia Barakova[2], Matthias Rauterberg[2], and Carlo Regazzoni[1]

[1] Department of Engineering (DITEN), University of Genova, Genova, Italy
abetan16@gmail.com
[2] Department of Industrial Design, Eindhoven University of Technology, Eindhoven, Netherlands

Abstract. Wearable cameras allow users to record their daily activities from a user-centered (First Person Vision) perspective. Due to their favourable location, they frequently capture the hands of the user, and may thus represent a promising user-machine interaction tool for different applications. Existent First Person Vision, methods understand the hands as a background/foreground segmentation problem that ignores two important issues: (i) Each pixel is sequentially classified creating a long processing queue, (ii) Hands are not a single "skin-like" moving element but a pair of interacting entities (left-right hand). This paper proposes a GPU-accelerated implementation of a left right-hand segmentation algorithm. The GPU implementation exploits the nature of the pixel-by-pixel classification strategy. The left-right identification is carried out by following a competitive likelihood test based the position and the angle of the segmented pixels.

Keywords: Egovision · Hand-segmentation · GPU · Hand-detection · Wearable cameras

1 Introduction

Computer Vision and video analysis are nowadays two of the most explored topics in computer science. The increasing computational power, the data availability and the recent algorithmic developments are quickly attracting high-tech companies and computer scientist to develop systems to process and understand video streams. In particular, wearable cameras, by taking advantage of a privileged location, stand out as one of the most promising video perspectives nowadays. The videos recorded from this point of view are referred as First-Person Vision (FPV) or Egocentric videos [6].

The 90's idea of a device that can understand our surroundings and provide valuable assistance is nowadays technically possible. During the last couple of years, several successful applications of wearable cameras have been proposed in different fields such as Law Enforcement [10], Medical Applications [13], Lifelogging [14], among others. Due to the tight link with the user and its advantageous

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part I, LNCS 9913, pp. 504–517, 2016.
DOI: 10.1007/978-3-319-46604-0_36

location, wearable cameras are commonly pointed out as a promising strategy to enhance user-machine interaction by exploiting the hands usage as interaction instrument.

In seek of such intuitive interaction, hand-based methods constitute the most explored field in EgoCentric videos. Hand signals have been used for several purposes, for example, to understand conscious interactions with the device [1], to infer the gaze of the user without using eye-trackers [9], to infer the active objects in the scene [20], among others.

The authors in [4] propose a unified structure to develop hand-based methods in egocentric videos. The proposed structure highlights the importance of decomposing the understanding of the hands in multiple levels, starting with simple questions like the presence of the hands [2,3,5] and finally obtaining more complex variables like the shape of each hand [8], to finally analyze its trajectories and interactions.

Within the Unified Framework proposed by Betancourt [4], the most explored level is the hand-segmentation. Its goal is to extract the shape of the hands by following a background/foreground segmentation at a pixel-by-pixel level. As shown in the literature this pixel-by-pixel approach achieves reliable results; However it has to deal with challenging aspects such as the illumination changes and the significant number of operations required. For instance, the camera of the Google glasses has a resolution of 720 p and records 30 fps, implying 928.800 pixel classifications per frame and a total of 27′864.000 per second of video.

Regarding the illumination changes, this is partially alleviated by using depth sensors or by combining multiple hand-segments, each of them trained to deal with particular light conditions. The former is usually restricted by the use of bigger devices with extra battery requirements; while the training data availability limits the latter [8]. About the computational complexity, a promising strategy is to simplify the frames by using SLIC superpixels [22]. On one side, the superpixel approach reduces the number of classifications tasks per frame and includes the concept of an edge in the segmentation; on the other side, it is necessary to run the SLIC algorithm frame by frame and the classification errors are considerably larger.

In addition to the technical challenges, the traditional background/foreground also carries the conceptual issues in the definition of hands understanding. On one side, the background/foreground approach assumes that both hands are equal and constitute the foreground of the scene, while on the other side, based on human studies, the hands are commonly defined as coupled system centrally coordinated by the brain to achieve a particular goal. Furthermore, there are considerable differences in the motion skills of both hands. In average, 9 out of 10 individuals are right-handed and as a consequence, their upper limb movement skills are asymmetric concerning speed, control, and strength. These differences are significantly larger in patients with upper limb motor problems such as cerebral palsy or upper limb stroke.

This paper proposes an accelerated strategy to segment and identifies the hands of the user when recorded by a wearable camera. The proposed approach

follows the unified framework introduced by [4], but targets, in particular, the segmentation and identification level. The novelties of this paper are three folded:

(i) Formalises a multi-model hand-segmenter based on Random Forest and K-Nearest-Neighbors. The proposed hand-segmenter is inspired by the work of [18].
(ii) Proposes a GPU implementation of the multi-model hand-segmenter. The experimental results show that the accelerated version can process frames 6.2 times faster than a sequential CPU.
(iii) The accelerated hand segmenter is extended with the hand-identification level proposed in [8]. The experimental results show that using an acceptable compression rate it is possible to obtain reliable left/right hand-segmentations in real time.

The remaining of this paper is organized as follows: Sect. 2 introduces our approach. Subsection 2.1 formalizes the hand-segmentation and explains the single and multimodel hand-segmenters. Subsection 2.2 introduces the Random Forest classification algorithm and introduces two GPU kernels. Subsection 2.3 proposes the hand-identification mechanism. Finally, Sect. 3 evaluates the performance of the proposed kernels and the identification model. Section 4 concludes the paper.

2 Our Approach

The goal of this paper is to develop a strategy to delineate the left and right-hand silhouette by exploiting the GPU processing capabilities. Our approach extends the traditional pixel-by-pixel hand-segmentation approach by proposing an additional hand-identification step as suggested by [8]. Figure 1 summarizes the difference between the background/foreground approach and the Left/Right hand-segmentation. This section briefly introduces the pixel-by-pixel hand-segmentation problem, the algorithmic procedure behind random forests and the required improvements to segment each pixel as a separate thread in the GPU. Finally, the identification level is introduced. For more details about the identification step, please refer to [8].

2.1 Hand-Segmentation

It is probably the more explored problem in FPV. The main task is to delineate the silhouette of the hands at a pixel level. The more promising results are reported in [18] and [23] achieving F-scores around 90 % under slightly stable illumination conditions. The general idea behind these methods is to reduce the problem to a pixel binary classification problem, and then construct the frame result as the composition of the individual decisions.

In the most basic form, it is possible to train a single hand-segmenter by using a set of input frames and the ground truth masks (pixels belonging to the hands). This approach can be considered the evolution of the seminal work of

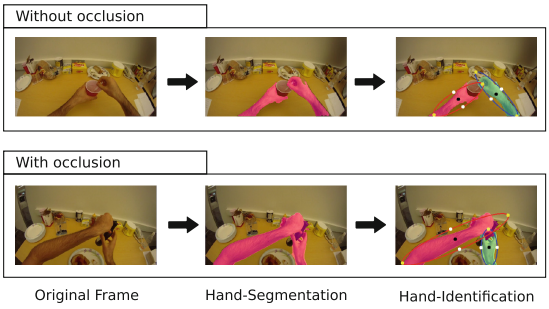


Fig. 1. The difference between hand-segmentation and hand-identification. The hand-to-hand occlusions are captured by following [8].

Jones and Rehg 1999 [16], with some differences in the used colour space and the classification algorithm. Recent works show that by using the *Lab* colour space (Lightness and a/b components) it is possible to alleviate the effect of small illumination changes. Figure 2 illustrates the general procedure to build a single hand-segmenter. In the first columns are the original frames, the training masks and their matrix representation. In the second column the training and testing stage. For illustrative purposes the training stage shows a decision tree; However, in practice, this can be a different type of classifier. Our experiments, as well as the state-of-the-art, are based on Random Forest classifiers.

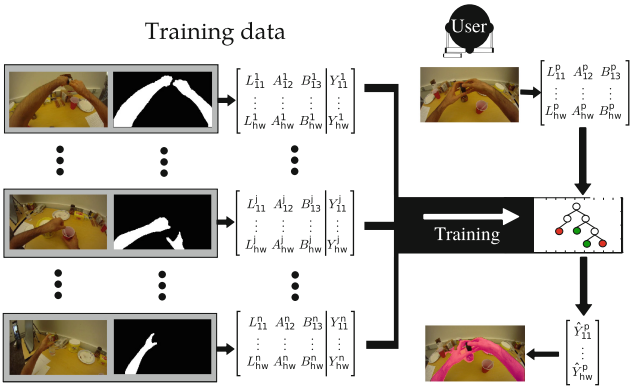


Fig. 2. Single-model hand-segmentation

In general a single hand-segmenter easily fails when applied to videos recorded with slightly changing light conditions. This problem can be partially alleviated by increasing the number of training frames, which in turns, could end up reducing the separability of the color space and producing excessive false-positives and false-negatives. Recent literature propose to train multiple

hand-segmenters, one per frame and switch between them depending of the light conditions of the frame. The latter can be captured by built-in light sensors or estimated indirectly by global features like color histograms or GIST [7]. Figure 3 summarizes our implementation of the multi-model hand-segmenter.

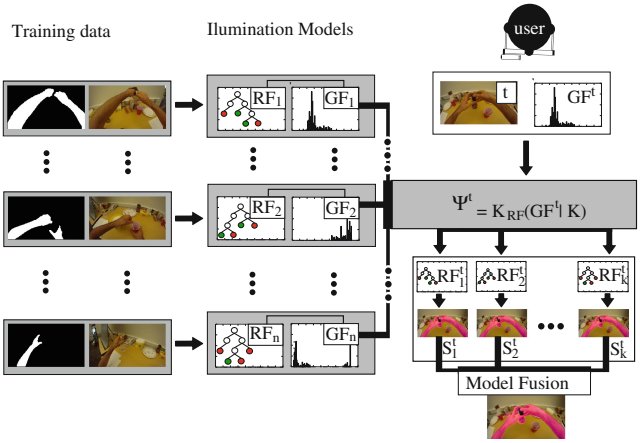


Fig. 3. Multi-model hand segmentation. Source [8]

The first column of the figure contains the manually labelled masks and their corresponding raw frames. Let us denote N as the number of manual labels available in the dataset, and n as the number of training pairs selected to build a multi-model binary hand-segmenter. For each training pair, $i = 1 \ldots n$ a trained binary random forest (RF_i) and its global feature (GF_i) are obtained and stored in a pool of illumination models (second column of the figure). Each RF_i is trained using the LAB values of each pixel in the frame i and as the class their corresponding values in the binary masks. As global feature (GF_i) we use the flatten HSV histogram. The choice of the colour spaces is based on the results reported by [17,21]. Finally, we train a K-Nearest-Neighbor K_{RF} with the global features to switch between the more suitable illumination models.

In the testing phase, the K_{RF} is used as a recommender system which, given the global features of a new frame, provides the indexes of the closest K illumination models (RF^t). These models are subsequently used to obtain K possible segmentations (S^t), which are finally fused to get the final binary hand-segmentation (HS^t). The third column of the figure illustrates this part of the procedure. Formally, let's denote the testing frame as t and its HSV-histogram as GF^t, the indexes of the closest K illumination models ordered from closest to furthest based on the Euclidean distance as Eq. (1), their corresponding K random forest as Eq. (2), and their pixel-by-pixel segmentation applied to t as Eq. (3).

$$\begin{aligned}\Psi^t &= K_{RF}(GF^t|K) \\ &= \{\psi_1^t, \ldots, \psi_K^t\} \qquad (1)\\ RF^t &= \{RF_{\psi_1^t}, \ldots, RF_{\psi_K^t}\} \qquad (2)\\ S^t &= \{RF_{\psi_1^t}(t), \ldots, RF_{\psi_K^t}(t)\} \\ &= \{S_1^t, \ldots, S_K^t\} \qquad (3)\end{aligned}$$

The binary hand-segmentation of the frame is the normalised weighted average of the individual segmentations in S^t, defined by Eq. (4). Where λ is a decaying weight factor, selected as 0.9.

$$HS^t = \frac{\sum_{j=1}^{K} \lambda^j \cdot S_j^t}{\sum_{j=1}^{K} \lambda^j} \qquad (4)$$

2.2 Random Forest and Decision Trees

The egocentric literature points to Random Forest (RF) as the most suitable classifier for the hand-segmentation problem. They offer a fast classification, a reliable skin detection and are less prone to overfitting. As shown in [8,23], under a proper training, the use of multiple random forests could provide a robust hand segmentations even under changing light conditions.

On its general definition, a Random Forests is an ensemble method that fuses the result of multiple Decision Trees, each of which is in turn on a subset of the training data [15]. In this way, the main classification workload is carried out by its decision trees. A Decision Tree, respectively, is an algorithmic strategy to divide the feature space in such a way that the proposed division fit the output variable [19].

Without loosing generality lets assume a set of N observations, each of them containing containing p features and a response: that is (x_i, y_i) for $i = 1, 2, \ldots, N$, with $x_i = (x_{i1}, x_{i2}, \ldots . x_{ip})$. Lets define an arbitrary partition of the input space X^p into M regions as $R_1, R_2, \ldots, R_M$, and the response of the model in each region as the constant c_m. In this way, the goal of the decision tree is to find, by using the input data, an appropriate partition and response constants to map the input space to the output space.

Given an space partition, the values of c_m can be obtained by defining an error function $J(c_m) = J(f(y_i, x_i, cm)|(x_i, y_i) \in R_m)$ and finding the constants that minimize it for each region (6). Finding the best partition of the space is computationally expensive for highly dimensional input spaces; However, it is possible to design recursive binary partitions in an efficient way.

$$\hat{c}_m = \underset{cm}{\operatorname{argmin}}\, J(c_m) \qquad \forall m \in 1, \ldots, M \qquad (5)$$

$$= \underset{cm}{\operatorname{argmin}}\, J(f(y_i, x_i, c_m)|(x_i, y_i) \in R_m) \qquad \forall m \in 1, \ldots, M \qquad (6)$$

Starting with all the inputs consider a splitting feature j and value s to divide the feature space in two half-planes, namely R_1 and R_2 defined in (7). The best

pair of (j, s) is the one that minimize the overall error of R_1 and R_2 as shown in (8). The same splitting procedure can be applied recursively on each half-plane until a stop criteria reached. Different stop criteria can be used to finish the procedure, such as the maximum tree depth or the minimum error required to consider valid a partition. Other options include the construction of large trees and then prune the unnecessary branches. The resulting regions and the model responses are defined by the leafs of the tree and their model responses respectively. The obtained tree can be applied to process new observations by following the rules captured by the sequences of (j, s) and returning the constant assigned to the final leaf.

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j > s\} \tag{7}$$

$$\begin{aligned} min_{j,s}\left[J(\hat{c}_1) + J(\hat{c}_2)\right] = min_{j,s}[&J(f(y_i, x_i, \hat{c}_1)|(x_i, y_i) \in R_1(j,s)) \\ &+ J(f(y_i, x_i, \hat{c}_2)|(x_i, y_i) \in R_2(j,s))] \end{aligned} \tag{8}$$

Once obtained the decision tree it can be represented by using 5 arrays: The decision variables J, the splitting values S, the position of the left and right nodes L and R respectively, and the response constants C. Each of these vectors has as many elements as nodes in the trained decision tree. For a Random Forest, the arrays of all its Decision Trees can be merged by keeping control of the position of the first node of each decision tree. These positions can be stored in a sixth array F. Algorithm 1 shows the Random Forest decision procedure. To obtain the Decision Tree pseudocode it is necessary to modify lines 3, 7 and 9, or define $F = [0]$ in the Random Forest procedure.

Algorithm 1. Random Forest Pseudocode.

```
1: function SegmentPixel(pixel)
2:     result = 0
3:     for i ∈ F do
4:         node = F[i]
5:         while L[node]! = −1 do
6:             if pixel[J[node]] ≤ S[node] then
7:                 node = L[node] + F[i]
8:             else
9:                 node = R[node] + F[i]
10:            end if
11:        end while
12:        result+ = C[node]
13:    end for
14:    return result
15: end function
```

Existent hand-segmenters process each single pixel sequentially in the CPU of the device. This approach highly restricts the number of pixels being process at every time instance and creates long queues of pixels to be processed. Additionally, by analysing Algorithm 1, it is clear that the traversing algorithm is the result of a sequential access to the decision arrays (J, S, L, R) comparing the input features with (S) to return the average value (C) finally. This description

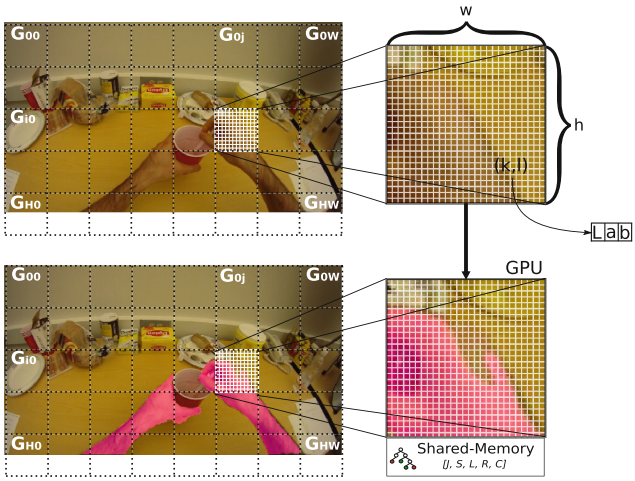


Fig. 4. Summary of the GPU hand-segmenter, including the block and grid dimensions and the information stored in the shared memory.

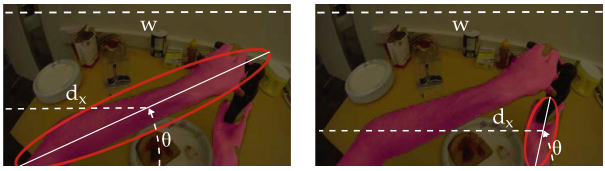
and easily fits the single instruction multiple data paradigm (SIMD) and point to promising speed improvements if processed by GPUs [11].

Figure 4 summarizes the intuition behind our GPU implementation. In practice we store the decision vectors in the shared memory of the GPU to guarantee that they can be accessed quickly by all the threads. This is done only once after the training phase is finished. In addition to the decision arrays the processing kernel must be uploaded to processing units. For comparative purposes we propose two kernels; The first kernel (GPU-DT) only parallelize the decisions of each Decision Tree, while the second kernel (GPU-RF) also includes the average of the decision trees. The pseudocode, as the final kernels, assume that J, S, C, L, R, and F are stored in the shared memory of the device. These arrays does not change and can be submitted to the shared memory immediately after the training phase.

2.3 Hand-Identification

Once obtained the hand-segmentation it is possible to fit an ellipse to the contours of the hands. A quick analysis of egocentric videos of daily activities easily points to the angle of the hands with respect to the lower frame border (θ), and the normalized horizontal distance to the left border (x) as two discriminative variables to build our L/R hand-identification model. Figure 5 illustrates these variables.

For the identification level, we use the Maxwell model proposed in [8]; where the identity of the hand is decided according to the result of a likehood ratio test between two maxwell distributions. The reasons behind the choice of the Maxwell distribution are two: (i) It is positive defined (ii) It allows to include



(a) Geometric problem of the left hand-segment (b) Geometric problem of the right hand-segment

Fig. 5. Input variables for the L/R hand-identification model.

an asymmetry factor in our formulation. The mathematical formulation for the left hand (p_l) and the right hand (p_r) is given by Eqs. (9) and (10) respectively, where p_x is the Maxwell distribution with parameters $\Theta = [d, a]$. The values of x and θ are defined in the interval $[0, 1]$ and $[0, \pi]$ respectively. In general d controls the displacement of the distribution (with respect to the origin) and a controls its amplitude.

$$p_l(x, \theta|\Theta_l^x, \Theta_l^\theta) = p(x|\Theta_l^x)p(\theta|\Theta_l^\theta) \tag{9}$$

$$p_r(x, \theta|\Theta_r^x, \Theta_r^\theta) = p(1 - x|\Theta_r^x)p(\pi - \theta|\Theta_r^\theta) \tag{10}$$

$$p(x|\Theta) = p(x|d, a) = \sqrt{\frac{2}{\pi}} \frac{(x-d)^2}{a^3} e^{-\frac{(x-d)^2}{2a^2}} \tag{11}$$

In total this formulation contains 8 parameters summarized in Eq. (12). As notation, the subindex of Θ refers to the left (l) or right (r) parameters, and the super-index refer to the horizontal distance (x) or the anti-clockwise angle (θ). The parameters of the model are selected by fitting the model to the angles observed in the Kitchen dataset of subject 1. The final values are given by Eq. (13). For more details about the fitting procedure and the motivation behind the Maxwell formulation please refer to [8].

$$\begin{bmatrix} \Theta_l^x & \Theta_l^\theta \\ \Theta_r^x & \Theta_r^\theta \end{bmatrix} = \begin{bmatrix} d_l^x & a_l^x & d_l^\theta & a_l^\theta \\ d_r^x & a_r^x & d_r^\theta & a_r^\theta \end{bmatrix} \tag{12}$$

$$= \begin{bmatrix} -0.05 & 0.24 & -0.63 & 0.94 \\ -0.08 & 0.21 & -0.91 & 1.10 \end{bmatrix} \tag{13}$$

To compare the goodness of fit of the L/R hand-identification models given by Eqs. (9) and (10) we perform a Likelihood ratio test on the post-processed hand-like segments. The Likelihood ratio test is given by Eq. (14).

$$\Lambda(x, \theta) = \frac{L_l(\Theta_l^x, \Theta_l^\theta|x, \theta)}{L_r(\Theta_r^x, \Theta_r^\theta|x, \theta)} = \frac{p_l(x, \theta|\Theta_l^x, \Theta_l^\theta)}{p_r(x, \theta|\Theta_r^x, \Theta_r^\theta)}, \tag{14}$$

Relying only on the likelihood ratio, could lead to cases where two hand-like segments are assigned the same label (left or right). To avoid this cases, and given that a frame cannot have two left nor two right hands, we follow a competitive

rule in the following way. Lets assume two hands-like segments in the frame described by $z_1 = (x_1, \theta_1)$ and $z_2 = (x_2, \theta_2)$ as explained in Fig. 5, and their respective likelihood ratios given by $\Lambda(x_1, \theta_1)$ and $\Lambda(x_2, \theta_2)$. The competitive ids are assigned by Eq. (15).

$$id_{z_1}, id_{z_2} = \begin{cases} \Lambda(x_1,\theta_1) > \Lambda(x_2,\theta_2) \rightarrow & id_{z_1} = l \\ & id_{z_2} = r \\ \Lambda(x_1,\theta_1) \leq \Lambda(x_2,\theta_2) \rightarrow & id_{z_1} = r \\ & id_{z_2} = l \end{cases} \tag{15}$$

3 Results

To measure the speed improvement of the accelerated hand-segmenter we use the subject 1 sequences of the kitchen dataset and two different GPU hand-segmentation kernels: The first kernel (DT_{GPU}) performs each Decision Tree as separate task in the GPU[1] and fuses the results in the CPU[2]. The second kernel (RF_{GPU}) evaluates the full Random Forest including the average directly in the GPU. Finally, the CPU sequential implementation of the algorithm is used as the baseline.[3] For comparative purposes the segmentation is performed at different compression levels. The results reported use 20 illumination models and fuses the closest 5 on each frame.

Figure 6 shows the time in seconds required by each hand-segmenter to process each frame at a particular compression width. It is noteworthy from the figure that the largest benefits, as expected, are obtained when the full image is segmented where the GPU_{RF} and GPU_{DT} kernels process in average 6.2 and 4.8 times faster than the CPU counterpart. The GPU_{RF} takes 0.052 s per frame while the CPU takes 0.32 s. In the worst case scenario, the CPU implementation takes 0.41 s, and the GPU_{RF} takes 0.058 s, which means that a throughput of 17.24 full resolution frames per second.

If a detailed segmentation is not required a common practice is to compress the frames before segmentation. If compressed to $180px$ width and assuming the worst scenario the GPU_{RF} and CPU could segment 141 and 44 fps respectively. It is noteworthy that this compression rate strongly compromises the detail of the segmentation but could lead to real-time analysis of the motion patterns of the user. The dashed and dotted lines in the figure shows the processing speed required to process video streams of 40 and 60 fps respectively.

To evaluate the segmentation quality we use the coffee sequence for training and the remaining ones for testing (i.e. CofHoney, HotDog, Tea, Pealette). Table 2 shows the performance of the Left/Right segmentation for each video sequence and the overall performance in the last columns. For this table the

[1] NVIDIA Corporation GF116 [GeForce GT 640 OEM].
[2] Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz.
[3] For the CPU baseline we use the Cython procedure available in sklearn.

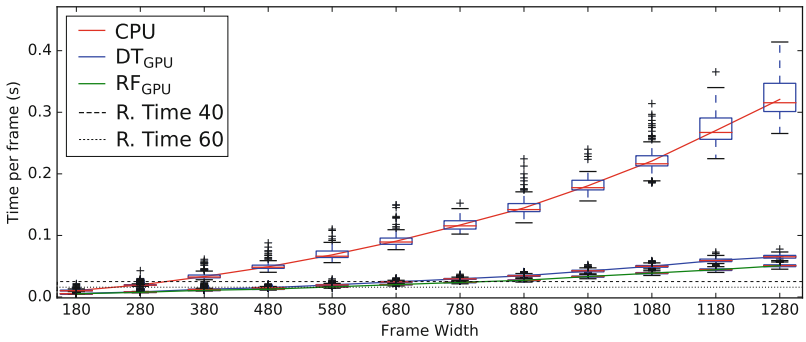


Fig. 6. GPU vs. CPU comparison.

Table 1. L/R hand-segmentation confusion matrix. This table uses the "Coffe" video sequence for training [8].

	CofHoney			Hotdog			Tea			Pealette			Total		
	No-hands	Left	Right	No-hands	Left	Right	No-hands	Left	Right	No-hands	Left	Right	No-hands	Left	Right
No-hands	0.990	0.003	0.007	0.989	0.005	0.006	0.996	0.002	0.002	0.991	0.006	0.003	**0.992**	**0.004**	**0.004**
Left	0.064	0.932	0.004	0.040	0.958	0.002	0.056	0.943	0.001	0.120	0.871	0.009	**0.073**	**0.923**	**0.004**
Right	0.092	0.002	0.906	0.136	0.001	0.864	0.082	0.000	0.918	0.112	0.002	0.886	**0.096**	**0.001**	**0.903**

hand-to-hand occlusions were disambiguated by using the algorithm proposed in [8] (Table 1).

Finally, Table 2 compares the multi-model hand-segmenter with previous works. Compared with a single pixel by pixel classifier of [18], our approach achieves improvements between 3 and 5 $F1$ score points. After the post-processing and identification process our method achieves a total $F1$ improvement of 9, 12 and 14 $F1$ points on the Coffee, Tea and Peanut video sequences respectively. In comparison to the shape aware hand-segmenter proposed by [23], our implementation performs better in all the video sequences. In particular, the "Tea" video sequence is improved by 10 $F1$ points.

Table 2. Hand-Segmenter state of the art comparison [8].

	Coffee	Tea	Peanut
1999 - Single pixel color [16]	0.83	0.80	0.73
2011 - stabilization + gPb + superpixel + CRF [12]	0.71	0.82	0.72
2013 - Li 1×1 window [18]	0.85	0.82	0.74
2013 - Li 9×9 window [18]	0.88	0.88	0.76
2014 - Shape Aware Forest (post-process) [23]	0.90	0.84	0.84
2016 - Ours (k = 20, m = 50)	0.88	0.87	0.77
2016 - Ours (k = 20, m = 50) + Hand-Id Post Process	**0.94**	**0.94**	**0.88**

4 Conclusions and Future Research

This paper contributes to 3 of the more challenging aspects of hand-segmentation methods in egocentric videos. The contributions of this paper are three folded:

(i) It proposes a hand-segmenter that fuse multiple hand-segmenter to alleviate illumination changes.
(ii) The proposed hand-segmenter is accelerated by using two different GPU kernels. The GPU hand-segmenter process 6.2 times faster that the sequential version.
(iii) A probabilistic hand-identification framework is introduced. This identification level seek to reduce the conceptual differences of traditional background/foreground hand segmenters and the human understanding of the hands as two cooperative entities. The proposed method properly identifies 99 % of the left and right hands.

As future research we highlight two possible extensions of this work: The first one is to migrate the implementation to embedded hardware. This would require additional work in the hardware side. (ii) Another interesting research line is to extend the identification level with tracker systems as proposed in the Unified Framework of Betancourt 2015 [4]. This is a promising research line for example to understand the motion patterns in patients of upper limb motor diseases.

Acknowledgement. This work was partially supported by the Erasmus Mundus joint Doctorate in Interactive and Cognitive Environments, which is funded by the EACEA, Agency of the European Commission under EMJD ICE.

References

1. Baraldi, L., Paci, F., Serra, G., Benini, L., Cucchiara, R.: Gesture Recognition using wearable vision sensors to enhance visitors' museum experiences. IEEE Sens. J. **15**(5), 1 (2015). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7058423
2. Betancourt, A., Lopez, M., Regazzoni, C., Rauterberg, M.: A sequential classifier for hand detection in the framework of egocentric vision. In: Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 600–605. IEEE, Columbus, June 2014. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6910041
3. Betancourt, A., Morerio, P., Barakova, E.I., Marcenaro, L., Rauterberg, M., Regazzoni, C.S.: A dynamic approach and a new dataset for hand-detection in first person vision. In: Azzopardi, G., Petkov, N., Yamagiwa, S. (eds.) CAIP 2015. LNCS, vol. 9256, pp. 274–287. Springer, Heidelberg (2015). doi:10.1007/978-3-319-23192-1_23
4. Betancourt, A., Morerio, P., Marcenaro, L., Barakova, E., Rauterberg, M., Regazzoni, C.: Towards a unified framework for hand-based methods in first person vision. In: IEEE International Conference on Multimedia and Expo (Workshops). IEEE, Turin (2015)
5. Betancourt, A., Morerio, P., Marcenaro, L., Rauterberg, M., Regazzoni, C.: Filtering SVM frame-by-frame binary classification in a detection framework. In: International Conference on Image Processing. IEEE, Quebec (2015)

6. Betancourt, A., Morerio, P., Regazzoni, C., Rauterberg, M.: The evolution of first person vision methods: a survey. IEEE Trans. Circuits Syst. Video Technol. **25**(5), 744–760 (2015). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7055926
7. Betancourt, A., Díaz-Rodríguez, N., Barakova, E., Marcenaro, L., Rauterberg, M., Regazzoni, C.: Unsupervised understanding of location and illumination changes in egocentric videos (2016). arXiv preprint: http://arxiv.org/abs/1603.09200
8. Betancourt, A., Morerio, P., Marcenaro, L., Barakova, E., Rauterberg, M., Regazzoni, C.: Left/Right Hand Segmentation in Egocentric Videos. ArXiv e-prints Under Revi (2016)
9. Buso, V., Benois-Pineau, J., Domenger, J.P.: Geometrical cues in visual saliency models for active object recognition in egocentric videos. In: Proceedings of the 1st International Workshop on Perception Inspired Video Processing - PIVP 2014, pp. 9–14. ACM Press, New York (2014). http://dl.acm.org/citation.cfm?id=2662996.2663007
10. Coudert, F., Butin, D., Le Métayer, D.: Body-worn cameras for police accountability: opportunities and risks. Comput. Law Secur. Rev. **31**(6), 749–762 (2015). http://dx.doi.org/10.1016/j.clsr.2015.09.002
11. Duncan, R.: A survey of parallel computer architectures. Computer **23**(2), 5–16 (1990)
12. Fathi, A., Ren, X., Rehg, J.M.: Learning to recognize objects in egocentric activities. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3281–3288. IEEE, Providence, June 2011. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5995444
13. Feng, S., Caire, R., Cortazar, B., Turan, M., Wong, A., Ozcan, A.: Immunochromatographic diagnostic test analysis using google glass. ACS Nano **8**(3), 3069–3079 (2014). http://pubs.acs.org/doi/abs/10.1021/nn500614k
14. Harvey, M., Langheinrich, M., Ward, G.: Remembering through lifelogging: a survey of human memory augmentation. Pervasive Mobile Comput. **27**, 14–26 (2016). http://www.sciencedirect.com/science/article/pii/S157411921500214X
15. Hastie, T., Tibshirani, R.J., Friedman, J.: The Elements of Statistical Learning, vol. 1, 10th edn. Springer, Heidelberg (2009). http://www.springerlink.com/index/D7X7KX6772HQ2135.pdf
16. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. Int. J. Comput. Vis. **46**, 81–96 (2002). IEEE Computer Society, Fort Collins, CO
17. Li, C., Kitani, K.: Model recommendation with virtual probes for egocentric hand detection. In: 2013 IEEE International Conference on Computer Vision, pp. 2624–2631. IEEE Computer Society, Sydney (2013). http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6751437
18. Li, C., Kitani, K.: Pixel-level hand detection in ego-centric videos. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3570–3577. IEEE, June 2013. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6619302
19. MathSoft: Classification and regression trees. Guide to Statistics 1, 369–401, February 1999
20. Matsuo, K., Yamada, K., Ueno, S., Naito, S.: An attention-based activity recognition for egocentric video. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 565–570. IEEE, June 2014. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6910036

21. Morerio, P., Marcenaro, L., Regazzoni, C.: Hand detection in first person vision. In: Fusion, p. 6. University of Genoa, Istanbul (2013). http://www.isip40.it/resources/papers/2013/fpv_FUSION2013.pdf
22. Singhai, S., Satsangi, C.: Hand segmentation for hand gesture recognition. In: Workshop on Interactive Multimedia on Mobile & Portable Devices, vol. 1, pp. 48–52. ACM Press, New York (2014). http://dl.acm.org/citation.cfm?id=2505490
23. Zhu, X., Jia, X., Wong, K.-Y.K.: Pixel-level hand detection with shape-aware structured forests. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014, Part IV. LNCS, vol. 9006, pp. 64–78. Springer, Heidelberg (2015). doi:10.1007/978-3-319-16817-3_5