REAL-TIME UNDERSTANDING OF 3D VIDEO ON AN EMBEDDED SYSTEM

Olivier Steiger
ABB Switzerland Inc.
CH-5405 Baden 5 Dättwil
olivier.steiger@ch.abb.com

Stephan Weiss
ETH Zürich
CH-8092 Zürich
stephan.weiss@mavt.ethz.ch

Judith Felder
ETH Zürich
CH-8092 Zürich
judith.felder@alumni.ethz.ch

ABSTRACT

A method is proposed for extracting semantic knowledge from 3D video in real-time on an embedded system. The method consists of foreground segmentation, object segmentation, temporal tracking and classification stages. It assigns relevant objects to different classes (e.g., human, vehicle) and determines their spatial location. These features are used in numerous applications including intrusion detection, people or vehicles counting, incident detection and activity monitoring. On a DSP-based embedded system, people recognition and object localization have been performed at over 10 frames per second. In the process, a time-of-flight range imaging camera has been used for video input. The experiments have also shown that the method is capable of coping with partial occlusion situations when different objects are located on distinct depth planes.

1. INTRODUCTION

Video understanding denotes the extraction of information that is meaningful to a human being (*high-level* or *semantic knowledge*) from video. Common examples of semantic knowledge include the kind of an object, determined by object recognition, and object properties such as location and orientation, as well as narrative information like scene length, motion activity, etc. In this paper, a video understanding solution is proposed that specifically targets real-time operation on an embedded system. In the process, 3D video input from e.g. a stereoscopic system or time-of-flight camera is used. With respect to 2D video, the high discriminative power of depth information allows for robust object segmentation and tracking with substantially lower algorithmic complexity. Additionally, 3D images provide valuable information about the kind and real-world location of objects.

In order to suit the particularities of embedded systems, three issues have to be taken into account: memory usage, computational effort and fixed point arithmetics. To limit memory usage, stored information is kept as concise as possible. Notably, compact descriptors capturing salient object features are used for temporal tracking and classification. Also, iterative methods have been avoided whenever feasible since they require the temporary storage of interim results. The usage of compact descriptors and the avoidance of iterative methods also help lowering computational demands. In fact with descriptors, only a few vectors need to be processed, as opposed to entire regions or contours with region-based or shape-based representations [1]. At last, many embedded systems do not comprise a floating point unit. To avoid costly software emulation, all operations have thus been implemented using fixed-point arithmetics. This has been appreciably simplified through the use of the scale-invariant Mahalanobis distance metric.

The present method partly extends earlier work on 2D video segmentation by one of the authors [2, 3]. It also benefits from numerous publications by fellow researchers. Murakami and Wada describe a real-time system for tracking walking people in 2D video [4]. Their proposal relies on background subtraction within edge images for object segmentation and on the projection of circles centered on objects for tracking. Classification is performed using a very basic color model of the human body. Hansen *et al.* address the fusion of information from intensity and depth images in order to build background models for segmentation [5]. Specifically, they use the joint distribution of intensity and depth information in Mixture of Gaussian and kernel density estimation models. At last, Ghobadi *et al.* describe a system for detection and classification of moving objects using 3D data [6]. Object segmentation is achieved by means of background subtraction of the range image. After a training phase, support vector classification (SVM) is applied to object features extracted through principal components analysis (PCA). Note that the real-time ability of the solutions proposed in [5, 6] has not been discussed.

Our solution goes beyond the above approaches by explicitly exploiting depth information in order to improve video understanding while preserving the real-time capability on embedded systems. Specifically, depth allows us to perform segmentation and tracking of multiple objects in the presence of mutual occlusions without resorting to costly multilevel region-object models [2]. Also, distance information helps us to efficiently classify objects based on their apparent 3D shape and to localize them in space. Finally, the proposed solution is not restricted to a particular kind of objects. Applications are numerous and include intrusion detection, people or vehicles counting, incident detection, activity monitoring, etc. The ability to run on an embedded system further allows local processing on, for instance, a surveillance camera. This also leads to potential system cost reductions due to the possible use of application-tailored hardware and to data redundancy removal at the source.

The remainder of this paper is organized as follows. In Section 2, the video understanding method is presented and its individual stages, namely foreground segmentation, object segmentation, temporal tracking and classification, are discussed. In Section 3, the method is validated in three different applications: tracking, people recognition and object localization. Finally, conclusions are drawn and some future work directions are outlined in Section 4.

2. VIDEO UNDERSTANDING

The proposed video understanding method is depicted in Figure 1. The 3D video input consists of an intensity image and of the corresponding aligned depth map. These data are typically produced by stereoscopic systems or time-of-flight range imaging cameras [7]. The system outputs are the kind of filmed objects (e.g., human, vehicle, ...) and their spatial location. These object features can be exploited in various applications such as those discussed in the introduction. Alternatively, they might be encoded using high-level descriptors (e.g., MPEG-7) for further processing [3].

Video understanding consists of four interdependent stages. *Foreground segmentation* tells apart the interesting foreground from the background. It produces the *foreground partition* which defines the areas of the frame containing relevant objects. Foreground segmentation results in the classification of pixels into two classes, namely foreground and background. However, no indications are provided yet about the individual objects that are part of the scene. *Object segmentation* therefore further subdivides the foreground into meaningful objects to produce the *object partition*. In the subsequent *temporal tracking* stage, the objects are individually followed in the scene. Finally, *classification* determines the kind of objects and their (absolute) spatial location.

Note that in our earlier work on 2D video analysis [2], a multi-

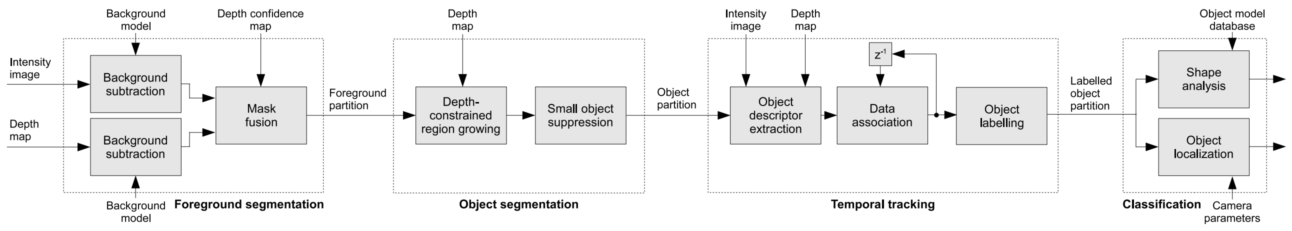


Figure 1: Block diagram of the 3D video understanding method.

level region-object model has been used to segment and track multiple and deforming objects. Here, this has been abandoned in favor of the simpler object-only model in order to fit the particular restrictions of embedded systems. In fact, spatial clustering for region segmentation requires extensive computations and the tracking of numerous regions substantially increases memory demands. Moreover, the main advantage of the multilevel model lies in the efficient handling of mutual occlusions. The latter can instead be achieved by exploiting depth information as discussed in Section 2.2.

2.1 Foreground segmentation

Foreground objects are segmented out by means of background subtraction. This involves calculating a background model, subtracting this model from each new frame and thresholding the result. The Running Gaussian Average approach taken here is based on ideally fitting a Gaussian probability density function (pdf) on the last n pixel's values [8]. Although more accurate methods exist, this has been chosen to maximize speed and limit memory usage.

Specifically at each pixel location (i,j), the running average is computed as

$$\mu_n = M\mu_{n-1} + (1-M)\big(\alpha I_n + (1-\alpha)\mu_{n-1}\big), \tag{1}$$

where I_n is the pixel's current value (intensity or depth) and μ_{n-1} is the previous average. α is an empirical weight chosen as a trade-off between stability and quick update. The binary value M is 1 in correspondence of a foreground value, and 0 otherwise. This prevents unduly updates of the background in the occurrence of foreground values. The standard deviation σ_n is computed similarly. Then at each frame time n, the pixel I_n is marked as foreground if

$$|I_n - \mu_n| > k\sigma_n; \tag{2}$$

otherwise, I_n will be classified as background. Here, k is a threshold which is scaled by the local variance σ_n.

Background subtraction is applied separately to the intensity image and to the depth map. Then, the two resulting foreground masks need to be merged to produce the foreground partition Π_f^n. Mask fusion relies on a confidence measure of the depth input to select the proper foreground mask. At each pixel location, the depth foreground mask value is retained if $C_n \geqslant \tau_c$; otherwise, the intensity mask value is used. The confidence threshold τ_c is often selected experimentally. Depth confidence C_n at frame time n is obtained differently depending on the video acquisition method. For stereo systems, it might for instance be computed using single-view stereo results [9]. For time-of-flight cameras, confidence is a function of the amplitude of the incident modulated signal [10].

Jointly using intensity and depth information appreciably increases the robustness of foreground segmentation. In fact, the depth map is pretty insensitive to photometric changes and therefore robust against illumination variations. On the other hand, depth cannot always be determined accurately. For instance, lack of corresponding points in a stereo image pair impedes disparity estimation. Similarly, light scattering and reflections often disturb depth measurements of time-of-flight cameras. In these cases, the intensity image is used to complete the missing parts.

2.2 Object segmentation

In order to understand video at the object level, the foreground partition must be further subdivided into individual objects. In the process, each *video object* (VO) in the object partition should ideally correspond to a meaningful or *semantic object* in the real world. Here, objects are segmented based on their spatial disparity. That is, objects that are disjoined in the real world are considered to be distinct.

To recognize object disparity, two assumptions are made about the connection between objects in the real world and their reproduction in the foreground partition. The first assumption is that disconnected blobs in Π_f^n always originate from disparate objects in the real world. This assumption only holds when foreground segmentation defines the shape of all objects correctly. When some object pixels are similar to the corresponding background pixels, part of the object might be erroneously identified by a set of pixels which is not connected to the rest of the object, thus invalidating the assumption. However, such errors can often be corrected by temporal tracking, as discussed in Section 3.2.1. The second assumption is that disparate but mutually occluding objects result in a depth discontinuity at their interface.

In practice, the first assumption is taken into account by assigning different object labels to each blob in the foreground partition. The second assumption is considered by further subdividing blobs into regions when sudden depth variations occur within the blob. Specifically, depth-constrained region growing is iteratively applied as follows until all foreground pixels have been labeled:

1. Select the top-left unlabeled pixel as seed point;
2. Check the neighboring pixels and add them to the region if their depth is similar (up to a threshold τ) to the seed;
3. Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added.

The above results in the object partition Π_o^n, where each video object is identified by an individual label. To reduce noise, Π_o^n is regularized by eliminating small connected sets of pixels.

2.3 Temporal tracking

Labels assigned by means of object segmentation are not coherent in time. That is, a particular real-world object is not necessarily given the same label at different frame times. This is the purpose of temporal tracking.

The proposed algorithm tracks the center of mass and the variances of objects. Each object i in the object partition Π_o^n is represented by a descriptor that summarizes object's features:

$$\mathbf{\Phi}_i(n) = \big(\phi_i^1(n), \phi_i^2(n), \ldots, \phi_i^{K_i(n)}(n)\big)^T, \tag{3}$$

where $K_i(n)$ is the number of features in frame n. In our specific implementation, $K_i(n) = K = 9$. In particular, $\big(\phi_i^1(n), \phi_i^2(n), \phi_i^3(n)\big)$ represents the object's center of mass, and $\big(\phi_i^4(n), \phi_i^5(n), \phi_i^6(n)\big)$ its motion vector. The center of mass is determined by averaging the spatial coordinates of all pixels belonging to the object i in Π_o^n. The motion vector is then obtained by computing the displacement of the center of mass between the current frame and the previous frame.

$\left(\phi_i^7(n), \phi_i^8(n), \phi_i^9(n)\right)$ represents the object's variances along the spatial axes. Due to the image formation process (*perspective projection* [1]), apparent object dimensions along the x and y axes scale down with distance to the camera. Therefore, the variances $\phi_i^7(n)$, $\phi_i^8(n)$ must be normalized with the factor $1/d^2$, where d denotes the mean distance between the object and the camera as given by the depth map. Note that the use of variances instead of, for example, spatial extension (*bounding box*) to describe object shape substantially reduces the impact of outliers. These are typically caused by erroneous object segmentation.

The subsequent data association stage establishes a correspondence between the objects partition in the current frame and the objects partition in the previous frame. Specifically, the proximity between object descriptors in Π_o^n and in Π_o^{n-1} is computed by measuring the Mahalanobis distance $D_M(\cdot)$ in the feature space:

$$D_M\left(\mathbf{\Phi}_i(n), \widetilde{\mathbf{\Phi}}_j(n-1)\right) = \sqrt{\sum_{s=1}^{K} \frac{\left(\phi_i^s(n) - \widetilde{\phi}_j^s(n-1)\right)^2}{\sigma_s^2}}, \qquad (4)$$

where σ_s^2 is the variance of the s^{th} feature over the entire feature space. In order to account for object motion, descriptors in the previous frame are motion-compensated prior to data association:

$$\begin{cases} \widetilde{\phi}_i^1(n-1) = \phi_i^1(n-1) + \phi_i^4(n-1) \\ \widetilde{\phi}_i^2(n-1) = \phi_i^2(n-1) + \phi_i^5(n-1) \\ \widetilde{\phi}_i^3(n-1) = \phi_i^3(n-1) + \phi_i^6(n-1). \end{cases} \qquad (5)$$

The value of the other features remains unchanged, so that $\widetilde{\mathbf{\Phi}}_i(n-1) = \left(\widetilde{\phi}_i^1(n-1), \ldots, \widetilde{\phi}_i^3(n-1), \phi_i^4(n-1), \ldots, \phi_i^K(n-1)\right)^T$.

The result of the distance computation can be represented as a distance matrix $\mathbf{D} = \{D_{Mp,q}\}$, where each row p corresponds to a region descriptor in frame n, and each column q corresponds to a region descriptor in frame $n-1$. The correspondence between the $\bar{p}^{th}$ region descriptor in frame n and the $\bar{q}^{th}$ region descriptor in frame $n-1$ is then confirmed if

$$d_{\bar{p},\bar{q}} = \min_q(D_{Mp,q}) = \min_p(D_{Mp,q}). \qquad (6)$$

If the condition in Equation (6) is respected, the track is updated. Otherwise, object descriptors are iteratively paired based on their distance. At last, the track of objects in Π_o^n is updated as a consequence of object descriptor tracking.

2.4 Classification

Once meaningful objects have been segmented and tracked, high-level knowledge such as the object's kind and location can be extracted. Here, objects are classified into different kinds by means of shape analysis. Each object's shape is described in terms of its apparent *elongatedness* E_{xy} [1] and depth variance σ_z^2:

$$E_{xy} = \frac{\sigma_x^2}{\sigma_y^2} = \frac{\phi_i^7}{\phi_i^8}; \qquad \sigma_z^2 = \phi_i^9. \qquad (7)$$

Although a variety of more robust contour-based and region-based shape descriptors exist [1], elongatedness and depth variance have the advantage of directly arising from temporal tracking. Therefore, only minimal additional computations are needed for their extraction. To classify the objects into different kinds, all shape descriptor pairs are then compared with reference descriptions that are stored in an object model database. In the process, the Mahalanobis distance is used to do the matching: each object is classified into the category (*kind*) to whose description it has the shortest distance. If the distance to all reference descriptions is longer than a preset classification threshold, the object remains unclassified.

Another useful information is the spatial location of objects. When both the intrinsic and the extrinsic parameters of the camera(s) are known, the real-world spatial location of any scene point can be computed as follows [1]. Let $\mathbf{X}_w = (x_w, y_w, z_w)^T$ express the scene point X in the world Euclidean coordinate system. The projected point can then be represented in the 2D image plane π in homogeneous coordinates as $\tilde{\mathbf{u}} = (U, V, W)^T$, and its 2D Euclidean counterpart is $\mathbf{u} = (u, v)^T = (U/W, V/W)^T$. Under the pinhole camera model approximation, $\tilde{\mathbf{u}}$ and $\mathbf{X}_w$ are related by the following expression:

$$z_c \tilde{\mathbf{u}} = \mathbf{K}\mathbf{R}(\mathbf{X}_w - \mathbf{t}). \qquad (8)$$

The camera calibration matrix $\mathbf{K}$ collects all intrinsic parameters. The translation vector $\mathbf{t}$ and rotation matrix $\mathbf{R}$ express the unique relation between world and camera coordinate systems (extrinsic parameters). Since both the location of X in the image plane, $\tilde{\mathbf{u}}$, and its depth, z_c, are known, the corresponding real-world location is obtained by solving Equation 8 for $\mathbf{X}_w$.

The above procedure can notably be used to determine the real-world location of object's centers of mass. To do so, the scene point X is substituted with the center of mass: $\mathbf{u} = (u, v)^T = \left(\phi_i^1(n), \phi_i^2(n)\right)^T$ and $z_c = \phi_i^3(n)$. In many applications, it is sufficient to determine only the relative location of objects with respect to the camera. The world Euclidean coordinate system is then identical with the camera Euclidean coordinate system, $\mathbf{X}_w = \mathbf{X}_c$. Thus $\mathbf{R} = \mathbf{I}$ is the identity matrix and $\mathbf{t} = \mathbf{0}$ is the null vector.

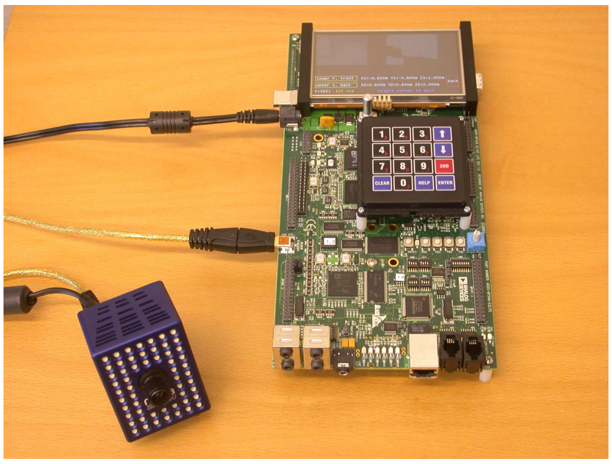


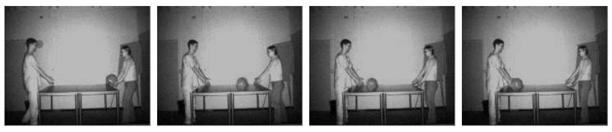
Figure 2: The experimental setup comprises a 3D camera, an embedded computation system and software.

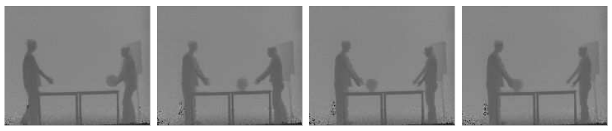
3. EXPERIMENTAL VALIDATION

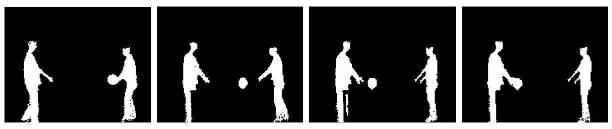
The video understanding method proposed in Section 2 has been implemented on an embedded system and validated in three different applications: tracking, people recognition and object localization. The results are discussed next.

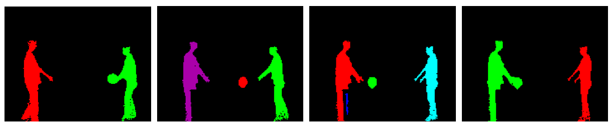
3.1 Setup

The experimental setup shown in Figure 2 comprises a 3D camera, an embedded computation system and software. The SwissRanger SR-3000 range imaging camera uses time-of-flight (TOF) technology [7] to produces aligned intensity and depth images with a resolution of 176×144 pixels (QCIF). The rated distance resolution is 1% of range and the typical frame rate is 25 fps. The ADSP-BF548 EZ Kit Lite by Analog Devices features a fixed-point digital signal processor (DSP) and a wide range of I/O peripherals, storage devices and interfaces. Notably, 64 MB SDRAM and 40 GB hard disc memory are available. The processor core is clocked at 533 MHz and provides up to 1066 MMACS performance. Also, an LCD touch screen and a small numeric keypad serve as human interfaces. The camera is connected to the computation system over USB.


(a)


(b)


(c)


(d)

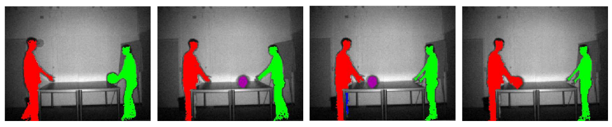

(e)

Figure 3: Tracking results for the sequence *Ping-pong*: frames 52, 58, 63 and 65. (a) Intensity image; (b) depth map; (c) foreground partition; (d) object partition; (e) labeled object partition superimposed on intensity image. Each label is represented by a distinct color.

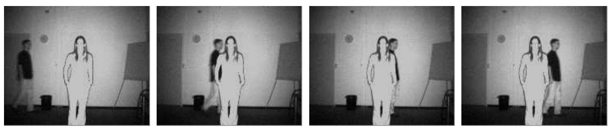
The application software has been written in ANSI-C and runs on top of the uClinux operating system. The code handles the graphical user interface (GUI), video understanding and data transfers over USB. The following empirical parameter settings have been used in all experiments: foreground update weight $\alpha = 0.05$, local variance threshold $k = 3$, depth confidence threshold $\tau_c = 0.6$, object segmentation threshold $\tau = 0.3$ meter.

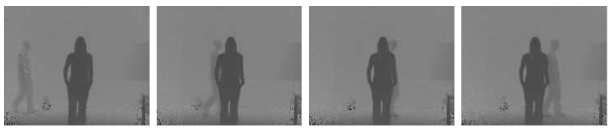
3.2 Results

In this section, results and the computational performance of the proposed method are discussed.

3.2.1 Tracking

Object segmentation and tracking have been evaluated with two different test sequences: *Ping-pong* and *Occlusion*. The first sequence represents two people playing with a ball; the second sequence displays a person passing behind another person. Four sample frames from *Ping-pong* and the corresponding foreground partition, object partition and labeled object partition are depicted in Figure 3. This sequence illustrates several strengths and weaknesses of the proposed approach. In Figure 3 (c)-(e) one realizes that, up to some noise, the shape of all objects is well segmented. Also both people are tracked reliably. In frame 52, the ball and the right person belong to the same blob in the foreground partition. Since they are also both located at the same distance from the camera, the two objects cannot be differentiated based upon their depth. Therefore, they are given the same label in the object partition. In frame 58, the ball is now separate from both people. Thus, a new object track


(a)


(b)

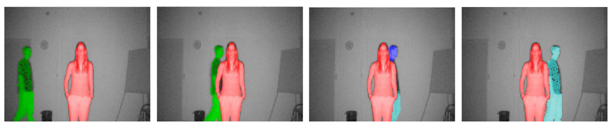

(c)

Figure 4: Tracking results for the sequence *Occlusion*: frames 95, 107, 114 and 118. (a) Intensity image; (b) depth map; (c) labeled object partition superimposed on intensity image.

is initiated and an individual label is assigned to the ball. In frame 63, the left leg of the left person is not completely segmented in the foreground partition, resulting in a gap between the person's and the leg's blobs. Hence a new track is erroneously initiated for the leg. Such behavior can be avoided by assigning identical labels to objects whose descriptors are "close enough". In frame 65, the ball and the left person are assigned the same label. However, the ball's track could be recovered once it re-detaches from the person by extending the data association stage to m past frames (instead of 1) in Equations 4 and 6. This will be the object of future work.

Whereas all objects in the *Ping-pong* sequence are separated by about the same distance from the camera, the *Occlusion* sequence depicted in Figure 4 contains useful depth information for the segmentation of overlapping objects. This is visible in frames 107, 114 and 118, where the two persons are differentiated in spite of belonging to the same blob in the foreground partition. Note also how a new track is initialized after the total occlusion occurring between frames 107 and 114; this could again be avoided by considering descriptors from several past frames in the data association stage.

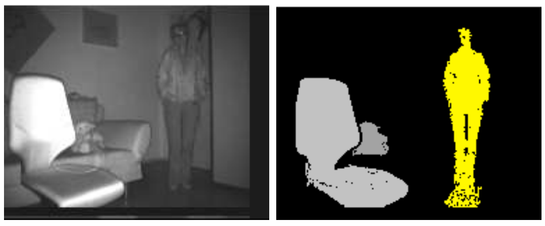
3.2.2 People recognition

One common kind of video understanding is people recognition. This is widely used in surveillance applications such as intrusion detection and people counting. The example depicted in Figure 5 shows a living-room where three objects have been successively brought in: an office chair, a soft toy and a person. The objects are classified based on their apparent elongatedness E_{xy} and depth variance σ_z^2, as described in Section 2.4.

To obtain the reference description, a typical person has been crudely modeled as a rectangular parallelepiped of aspect ratio 1:4 and apparent depth 0.2 m. This leads to $E_{xy,\text{ref}} = 0.06$ and $\sigma^2_{z,\text{ref}} = 3.4 \cdot 10^{-3}$. More refined people models could instead be derived from ground truth data. Using measurements from the example, the Mahalanobis distance D_M between the objects and the reference is then computed:

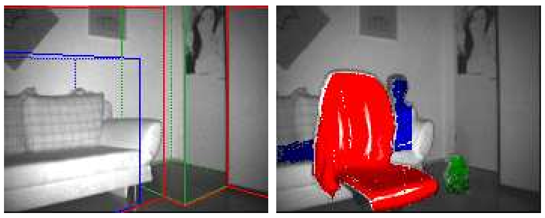
- **Person:** $E_{xy} = 0.07$, $\sigma_z^2 = 2.2 \cdot 10^{-3} \Rightarrow D_M = 0.56 \cdot 10^{-3}$
- **Soft toy:** $E_{xy} = 1.17$, $\sigma_z^2 = 3.1 \cdot 10^{-3} \Rightarrow D_M = 11.1 \cdot 10^{-3}$
- **Office chair:** $E_{xy} = 0.75$, $\sigma_z^2 = 21 \cdot 10^{-3} \Rightarrow D_M = 10.6 \cdot 10^{-3}$.

The Mahalanobis distance is one order of magnitude smaller for the person than for the other objects, making it easily classifiable.



(a)

Figure 5: People recognition example. (a) Intensity image; (b) object partition. Objects classified as people are highlighted in yellow.



(a)

Figure 6: Object localization example. (a) Definition of non-overlapping volumes; (b) localization of three objects with respect to the volumes indicated by object colorization.

3.2.3 *Object localization*

Object localization is notably needed to analyze object trajectories or to monitor particular areas for presence. The latter was the goal of the example illustrated in Figure 6. Here, the filmed space has been subdivided into three non-overlapping volumes. In order to avoid determining the camera calibration matrix, the volumes have been directly defined in the intensity image by means of the GUI. Since, unlike 2D video, each image point is uniquely located thanks to the depth information, apparent object surfaces can be labeled according to the volume they belong to.

3.2.4 *Performance analysis*

A major goal of the proposed method was to allow for real-time operation on an embedded system. In order to coarsely assess the performance of the algorithm, execution times of individual processing steps have been measured throughout the experiments. The results are summarized in Table 1. With an empty scene (no objects), only GUI handling, rendering and foreground segmentation are carried out. This leads to the minimal processing time of 40 ms, which corresponds to the maximum frame rate of 24 fps. With a typical scene involving three average-sized objects (e.g., the *Ping-pong* sequence in Figure 3), the frame rate drops to 11 fps. This is still considered acceptable for real-time video processing. Note that the performance loss is mainly due to object segmentation, where a time-consuming iterative region growing algorithm is used (Section 2.2). Since the effort increases with the number of pixels to be segmented, the image portion occupied by foreground objects must be kept small in order to maintain an acceptable frame rate.

4. CONCLUSIONS

A method for real-time video understanding on an embedded system has been proposed and validated in three different applications. The results have shown how distance information can be exploited to segment objects individually even in the presence of partial occlusion. Also, 3D video has enabled us to locate objects in space effectively. Finally, performance analysis has shown that with reasonably complex scenes, the algorithm runs at over 10 fps on a processor clocked at 533 MHz.

Step	**Min. (empty scene)**	**Typ. (3 objects)**
GUI handling & rendering	15.3	15.3
Foreground segmentation	25.1	25.1
Object segmentation	-	32.9
Temporal tracking	-	0.8
Shape analysis	-	2.3
Object localization	-	13.1
Total	**40.4** (24 fps)	**89.5** (11 fps)

Table 1: Execution time of individual processing steps (in ms).

These encouraging results allow us to outline some future work directions. So far, objects cannot be followed individually once they merge. This could be achieved by further subdividing objects into regions based on the motion projected objects or regions from past frames. This approach has been successfully applied to the 2D case by the authors [2]. One could also extend data association to multiple past frames in order to recover objects after total occlusions or temporary disappearance. Finally, parallelization could be exploited in order to increase the frame rate, notably by performing independent tasks (e.g., tracking, shape analysis and object localization) simultaneously.

REFERENCES

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, PWS Publishing, 1998.

[2] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Tracking video objects in cluttered background," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 4, pp. 575–584, April 2005.

[3] O. Steiger, A. Cavallaro, and T. Ebrahimi, "Real-time generation of annotated video for surveillance," in *Proc. of IEE Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'05*, April 2005.

[4] S. Murakami and A. Wada, "An automatic extraction and display method of walking persons' trajectories," in *Proc. of 15th Intl. Conf. on Pattern Recognition*, September 2000, vol. 4, pp. 611–614.

[5] D.W. Hansen *et al.*, "Cluster tracking with time-of-flight cameras," in *Computer Vision and Pattern Recognition Workshops, CVPRW '08*, June 2008, pp. 1–6.

[6] S.E. Ghobadi *et al.*, "Detection and classification of moving objects-stereo or time-of-flight images," in *Proc. of Intl. Conf. on Computational Intelligence and Security,*, November 2006, vol. 1, pp. 11–16.

[7] E. Stoykova *et al.*, "3-d time-varying scene capture technologies - a survey," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1568–1586, November 2007.

[8] M. Piccardi, "Background subtraction techniques: a review," in *Proc. of IEEE Intl. Conf. on Systems, Man and Cybernetics*, October 2004, vol. 4, pp. 3099 – 3104.

[9] G. Egnal, M. Mintz, and R.P. Wildes, "A stereo confidence metric using single view imagery," *Image and Vision Computing*, vol. 22, no. 12, pp. 943–957, October 2002.

[10] O. Steiger, J. Felder, and S. Weiss, "Calibration of time-of-flight range imaging cameras," in *Proc. of IEEE Conference on Image Processing*, October 2008, pp. 1968–1971.