Depth-supported real-time video segmentation with the Kinect

Alexey Abramov[1], Karl Pauwels[2], Jeremie Papon[1], Florentin Wörgötter[1], and Babette Dellen[3]
[1]Georg-August University, BCCN Göttingen, III Physikalisches Institut, Göttingen, Germany
{abramov,jpapon,worgott}@physik3.gwdg.de

[2]Computer Architecture and Technology Department, University of Granada, Granada, Spain
kpauwels@atc.ugr.es

[3]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona, Spain
bdellen@iri.upc.edu

Abstract

We present a real-time technique for the spatiotemporal segmentation of color/depth movies. Images are segmented using a parallel Metropolis algorithm implemented on a GPU utilizing both color and depth information, acquired with the Microsoft Kinect. Segments represent the equilibrium states of a Potts model, where tracking of segments is achieved by warping obtained segment labels to the next frame using real-time optical flow, which reduces the number of iterations required for the Metropolis method to encounter the new equilibrium state. By including depth information into the framework, true objects boundaries can be found more easily, improving also the temporal coherency of the method. The algorithm has been tested for videos of medium resolutions showing human manipulations of objects. The framework provides an inexpensive visual front end for visual preprocessing of videos in industrial settings and robot labs which can potentially be used in various applications.

1. Introduction

Video segmentation aims at representing image sequences through homogeneous regions (segments), where the same object part should carry the same unique label along the whole movie. The segmented visual data can be used for higher-level vision tasks which require temporal relations between objects to be established, including object tracking, action recognition, and content-based image retrieval [1, 2, 3]. The major challenges faced in video segmentation are processing time, temporal coherence, and robustness. In this work, we focus on scenarios showing object manipulations in a robot lab scenario, which are suitable for color and depth segmentation, and demonstrate that a coherent and robust video segmentation can be achieved under these conditions in real-time.

In the past, joint segmentation and tracking of segments in videos have been addressed in various works. Many methods for video segmentation are usually performing independent segmentations of each frame and then matching segments for tracking [4, 5, 6], having the disadvantage that segmentations need to be computed from scratch for each frame, affecting the efficiency of the method. Another problem is that the partition of the image may have changed in the new frame, leading to temporal consistency problems between segmentations. To resolve these problems, Grundmann et al. (2010) used a graph-based model to construct a consistent video segmentation from the over-segmented frames [7]. The over-segmentations however are still computed independently of each other, and processing times only reach frame rates of 1 Hz. Wang et al. (2009) formulated a joint object segmentation and tracking problem as a Markov random field energy minimization problem [8]. Observed and hidden variables of objects are defined in the first frame, and then the objects are tracked and segmented through the sequence. The method performs well but has the drawback that energy minimization is computationally very expensive, resulting in large processing times up to minutes per frame.

The described methods are based on color cues alone. In this paper, we show that the inclusion of depth information can greatly improve video segmentation. We extended a recently developed framework for parallel color video segmentation by Abramov et al. (2010) [9] by including depth information to the segmentation kernel. In this method, images are segmented using the method of superparamagnetic clustering of data, which finds the equilibrium states of a Potts model. Consistency of segmentations along the movie is obtained through label transfer from one frame to the next using warping based on real-time optical flow. This

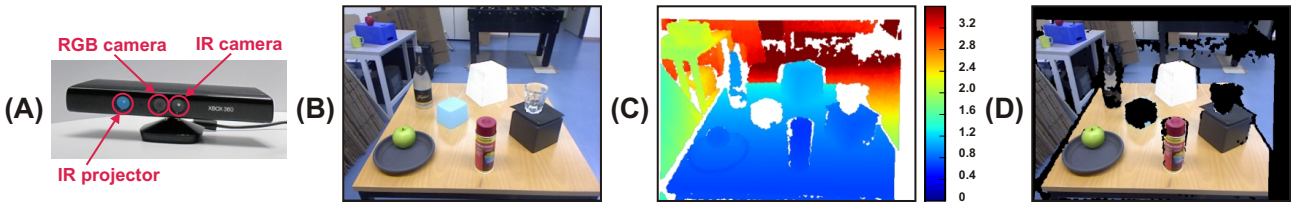

Figure 1. (A) The Kinect device. (B) Original frame acquired by RGB camera. (C) Depth data derived from IR image (in meters). White patches in the image denote pixels for which acquisition of depth information failed. (D) Color pixels having depth values.

way, the segmentation kernel can be initialized with solutions obtained at previous frames, improving the efficiency of the method significantly and allowing a soft tracking of segments to be carried out.

For depth acquisition, the Microsoft Kinect device, which was first released in the fall of 2010 for the XBox videogame platform [1], is used. The new device has immediately attracted the attention of the computer vision society because of its technical capabilities and its very low cost compared to time-of-flight sensors. The Kinect device features an IR projector for generating infrared images and two cameras: an RGB camera for capturing color images and an IR camera for capturing infrared images under various light conditions (see Fig. 1(A)). The IR camera is based on a monochrome CMOS sensor used in some time-of-flight cameras [10]. For indoor environments the Microsoft Kinect proves to be an inexpensive and suitable device for acquiring depth/color videos.

The structure of the paper is as follows. In Section 2, the calibration of the Kinect is explained. In Section 3, the framework for real-time video segmentation is presented. In Section 4, the algorithm is tested for several indoor scenarios showing a human manipulating objects. In Section 5, the results are discussed and directions for future research are given.

2. Kinect calibration

In a normal stereo setup, images derived from the calibrated cameras are rectified in order to obtain correspondent horizontal lines. In such a system, the relation between disparity and depth is given by $z = b \cdot f/d$, where z is the depth value (in meters), b is the baseline between two cameras (in meters), f is the focal length of the cameras (in pixels) and d is the disparity value (in pixels). Thus in the case of zero disparity values, the rays from both cameras are parallel and depth is infinite. However, the Kinect device returns raw disparity data which is not normalized in this way. So zero disparity values do not correspond to infinite distances. The relation of raw Kinect disparity to a normalized disparity is given by $d = 1/8 \cdot (d_{\text{off}} - kd)$, where d is the normalized disparity, kd is the Kinect disparity and d_{off} is the offset value particular to a given Kinect device. Values for kd and d_{off} are found at the calibration stage. Consequently, the relation between disparity and depth for the Kinect is given by

$$z = \frac{b \cdot f}{1/8 \cdot (d_{\text{off}} - kd)}. \quad (1)$$

In order to relate color and depth images (see Fig. 2(A,B)), pixels of the color image need to be matched to pixels of the depth image. Therefore, a calibration between IR and RGB cameras needs to be performed [2]. In the current work, the OpenNI toolbox [3] was used for the Kinect calibration and mapping of color pixels with range values (see Fig. 2(C,D)).

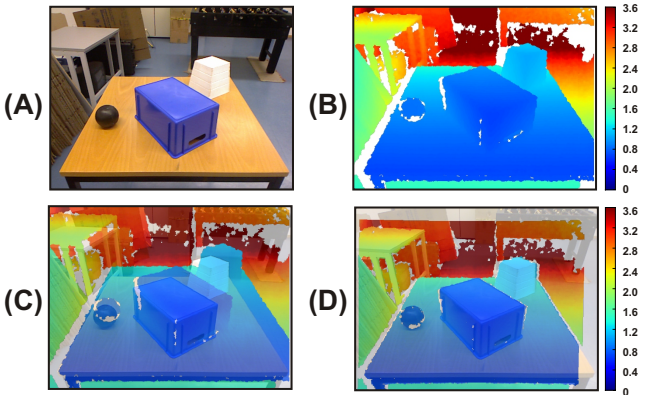

Figure 2. Calibration of the Kinect with OpenNI toolbox. (A) Original frame acquired by RGB camera. (B) Depth data derived from IR image (in meters). (C) Mapping of depth and color pixels without calibration. (D) Mapping of depth and color pixels after calibration.

3. Depth-supported video segmentation

We describe a method for the joint segmentation and tracking of segments in color/depth movies. The segmentation corresponds to the equilibrium state of a Potts model,

[1] Kinect for XBox 360: http://www.xbox.com/en-US/kinect

[2] for more details see http://cv4mar.blogspot.com

[3] available under http://www.openni.org

which is computed using a parallel Metropolis algorithm on the GPU [9]. Consistent video segmentation and segment tracking is achieved by transferring the solutions obtained at the precedent frame to the current frame. The resulting Potts configuration can then be used to find the current equilibrium state requiring only a few Metropolis iterations to be applied. This technique allows us to maintain labels along the sequence and to reduce computation times by recycling of already established solutions at earlier frames. We incorporated depth information into both the Potts model and the label transfer technique in a manner that is consistent with the color information, providing the main cue for segmentation. The inclusion of depth provides important additional information about object boundaries which improves video segmentation.

3.1. Image segmentation kernel

The image segmentation kernel proceeds as follows. The original frame is represented by the Potts model where a spin variable σ_k, which can have q discrete values $w_1, w_2, \ldots, w_q$, called spin states, is assigned to each pixel of the image. The parameter q should be chosen as large as possible since the spin states need to serve also as segment labels. In our experiments, we used $q = 256$. It is important to note that this choice of q has no influence on the performance and computation time of the image segmentation kernel itself. The energy of the system is described by

$$E = -\sum_{<ij>} J_{ij}\delta_{ij}, \tag{2}$$

where <i,j> denotes the closest neighborhood of spin i with $||i,j|| \leqslant \ell$, where ℓ is a constant. 2D bonds (i,j) between two pixels with coordinates (x_i, y_i) and (x_j, y_j) are created only if $|(x_i - x_j)| < \ell$ and $|(y_i - y_j)| < \ell$. In the current work we use $\ell = 1$. J_{ij} is an interaction strength or coupling constant and the Kronecker δ_{ij} function is defined as $\delta_{ij} = 1$ if $\sigma_i = \sigma_j$ and zero otherwise, where σ_i and σ_j are the respective spin variables of two neighboring pixels i and j. A coupling constant, determining the interaction strength between two pixels i and j, is given by

$$J_{ij} = 1 - |\mathbf{g_i} - \mathbf{g_j}|/\overline{\Delta}, \tag{3}$$

where $\mathbf{g_i}$ and $\mathbf{g_j}$ are the respective color vectors of the pixels and $\overline{\Delta}$ is the mean distance averaged over all bonds in the image.

Different from the original method, in the presented study coupling constants are computed in the HSV color space instead of the camera input RGB format. In the HSV space a pixel is represented by three values: hue (h), saturation (s) and value (v). Intensity (value) is separated from the color information (hue and saturation) which makes the method more consistent for objects having shadows and changes in lightness. The difference between two pixels i and j in the HSV color space is computed as the difference between two color vectors $\mathbf{g_i} = (s_i v_i \cos h_i, s_i v_i \sin h_i, v_i)$ and $\mathbf{g_j} = (s_j v_j \cos h_j, s_j v_j \sin h_j, v_j)$ using the following metric [11]:

$$\begin{aligned} |\mathbf{g_i} - \mathbf{g_j}| = [&(s_i v_i \cos h_i - s_j v_j \cos h_j)^2 \\ &+(s_i v_i \sin h_i - s_j v_j \sin h_j)^2 \\ &+(v_i - v_j)^2]^{1/2}. \end{aligned} \tag{4}$$

Since the current method uses 8-connectivity of pixels, interaction strengths for one pixel need to be computed in four different directions: vertical, horizontal, left diagonal, right diagonal. The depth data acquired along with the color image (see Fig. 3(A,B)) is used to prevent interactions between pixels having a large range difference. This is done by replacing all color differences having a displacement larger than a predefined threshold τ with the maximum possible value $\Theta = 250$ according to

$$J_{ij} = \begin{cases} J_{ij} & \text{if } |z_i - z_j| \leqslant \tau, \\ \Theta & \text{otherwise.} \end{cases}, \tag{5}$$

where z_i and z_j are range values of pixels i and j, respectively. Matrices containing color differences involved in the formation of segments (see (3)) are shown in Fig. 3(C - F). Excluded interactions, marked by dark red, prevent in most cases neighboring pixels to be assigned to the same segment.

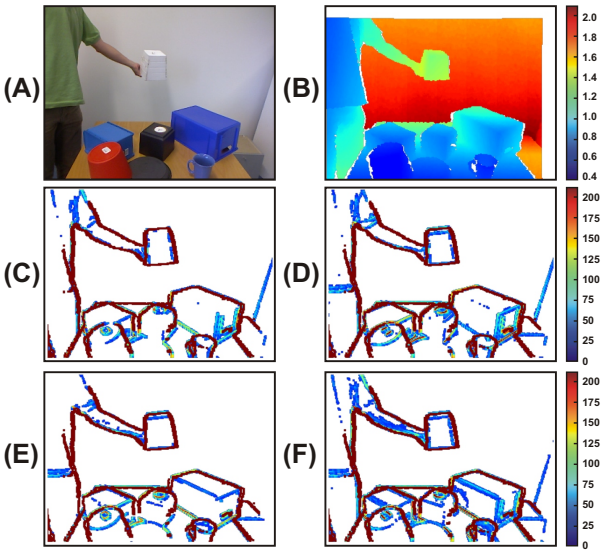

Figure 3. Color differences for the 8-connectivity case in HSV color space. (A) Original frame. (B) Depth data (in meters). (C - F) Matrices with color differences computed for horizontal, left diagonal, vertical and right diagonal directions ($\tau = 30$ cm).

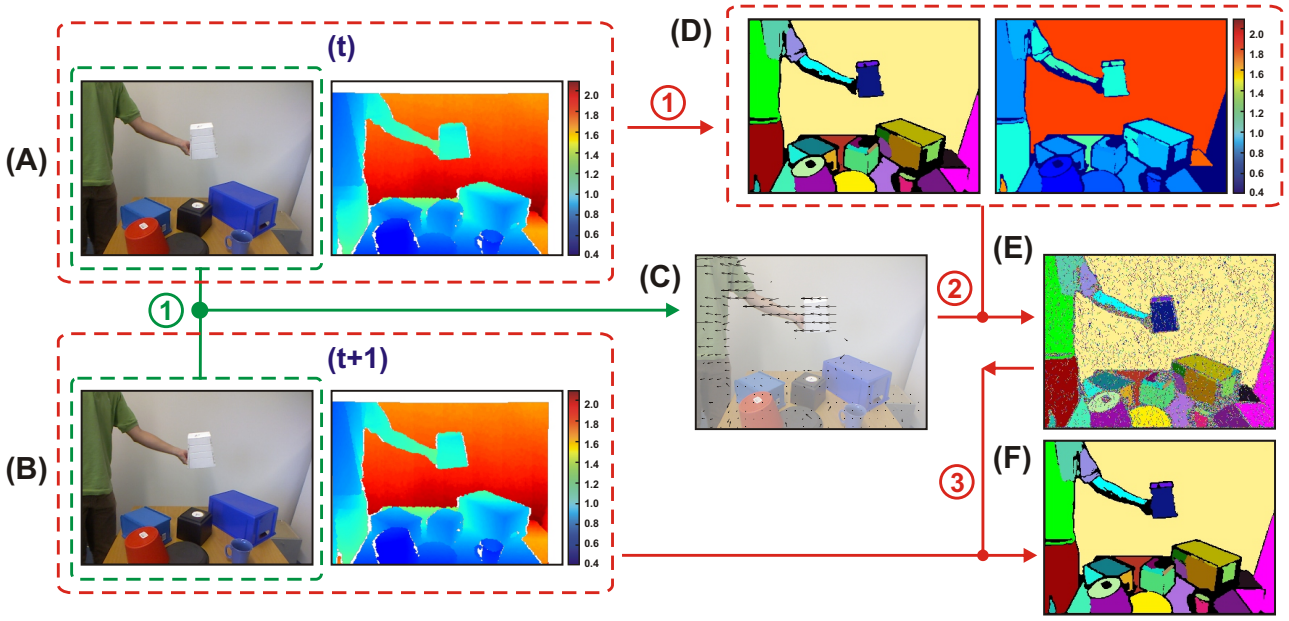

Figure 4. Segmentation of two adjacent frames in a sequence. Numbers at arrows show the sequence of computations. (A,B) Kinect data acquired at time steps t and $t+1$, respectively. (C) Estimated optical flow vector field (sub-sampled 11 times and scaled 10 times) (step 1). (D) Extracted segments S_t with correspondent average range values $\overline{z}$ (step 1). (E) Initialization of frame $t+1$ after the label transfer from frame t (step 2). (F) Extracted segments S_{t+1} (step 3).

Once the interaction strengths are determined, the spin state configuration is iteratively updated by the Metropolis algorithm with annealing choosing the state with the minimum energy at every iteration [12]. The update process runs until no more spin flips towards a lower energy state are being observed. The equilibrium state of the system, achieved after several Metropolis iterations, corresponds to the image partition or segmentation. The computation of coupling constants and the update of spin variables involve only the nearest neighbors of the respective pixels. Therefore the image segmentation kernel with the use of range data is appropriate for implementation on a GPU architecture. The method guarantees that every segment carries a spin variable which is unique within the whole image, thereby the terms *spin* and *label* are equivalent in this work.

3.2. Linking of segments

In the current study only the first frame of the video stream is segmented completely from scratch, i.e., all spin variables are initialized by random values. Consecutive frames are initialized by spin state configurations taken from previous frames considering spatial shifts due to motion. For these frames the image segmentation kernel is used in the relaxation mode, where the initial spin state configuration, warped from the previous frame using optical flow, needs to be adjusted to the current color image. To estimate the motion we use the real-time dense optical flow algorithm proposed by Pauwels et al. (2010) [13]. This algorithm runs on a GPU as well and belongs to the class of phase-based methods, which feature high robustness to changes in contrast, orientation and speed. The algorithm provides a vector field at each pixel indicating its motion

$$\mathbf{u}(x,y) = (u_x(x,y), u_y(x,y)). \tag{6}$$

An estimated optical flow vector field for two adjacent frames t and $t+1$ is shown in Fig. 4(A - C). Having segments with correspondent average range values for a time step t (see Fig. 4(D)) estimated optical flow vector field, labels of segments S_t are transferred to frame $t+1$ (see Fig. 4(E)), excluding transfers between pixels having a range difference larger than a pre-defined threshold τ. We obtain

$$S_{t+1}(x_{t+1}, y_{t+1}) = \begin{cases} S_t(x_t, y_t) & \text{if } \lambda \leqslant \tau, \\ 0 & \text{otherwise.} \end{cases}, \tag{7}$$

$$\begin{aligned} \lambda &= |z_{t+1}(x_{t+1}, y_{t+1}) - z_t(x_t, y_t)|, \\ x_{t+1} &= x_t + u_x(x_t, y_t), \\ y_{t+1} &= y_t + u_y(x_t, y_t), \end{aligned} \tag{8}$$

where z is a range data obtained from the Kinect. Label transfers between segments having large range differences are excluded as well, which yields:

$$S_{t+1}(x_{t+1}, y_{t+1}) = 0 \;\text{ if }\; \xi > \tau, \tag{9}$$

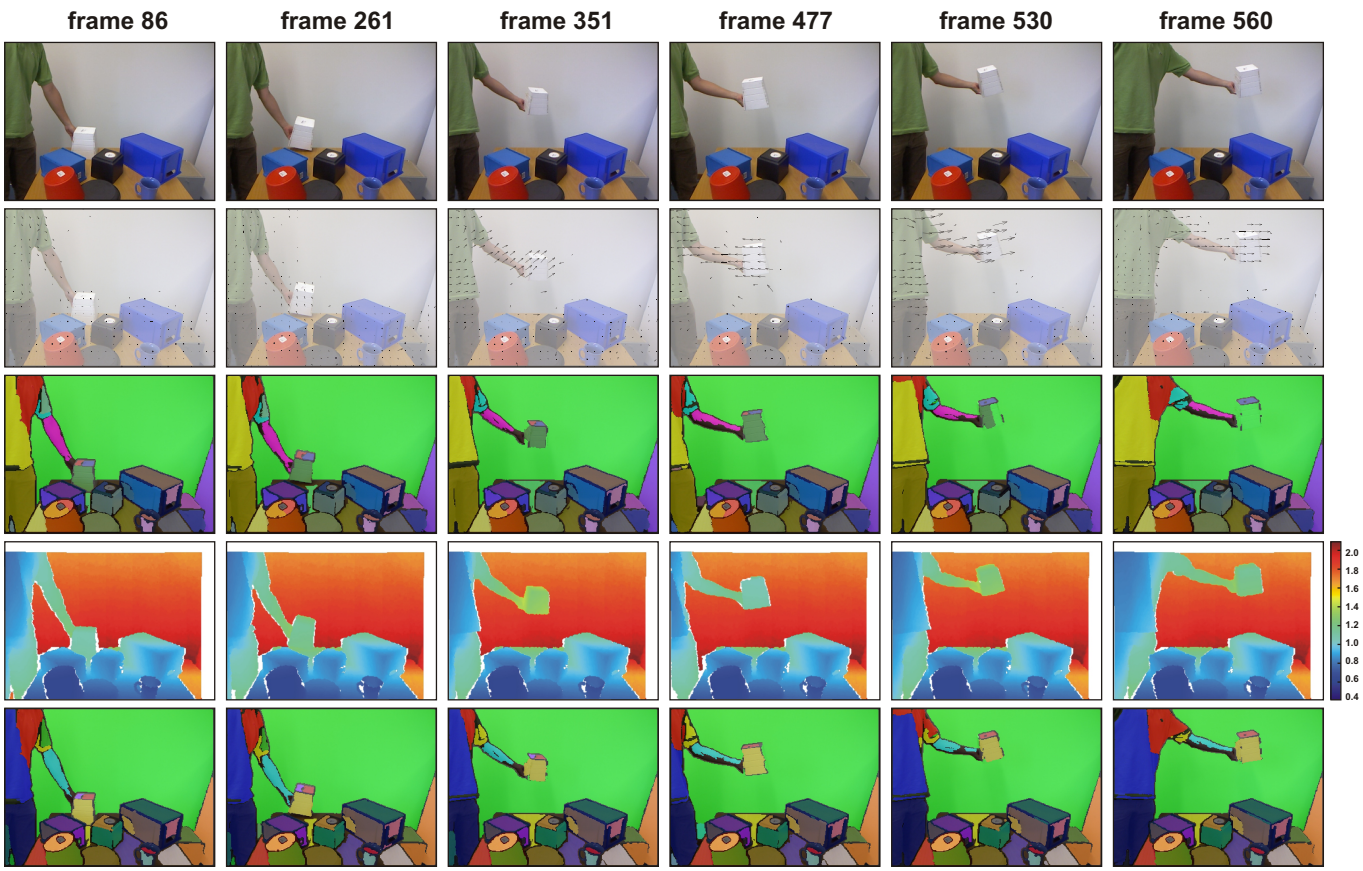

Figure 5. Segmentation of frame sequence "Moving an object". Original frames and estimated optical flow for selected time points are shown in the first and the second rows, respectively. Segmentation results without usage of range data are shown in the third row. The forth row shows depth data obtained from the Kinect. Segmentation results obtained using fusion of image and range data are depicted in the last row.

$$\xi = |z_{t+1}(x_{t+1}, y_{t+1}) - \overline{z}_t(x_t, y_t)|, \qquad (10)$$

and $\overline{z}$ being a matrix containing average range values for each segment (see Fig. 4(D)). Spin variables of pixels without correspondences are initialized with random values (see Fig. 4(E)). Then, the initial labels are adjusted to the data of frame $t + 1$ by equilibrating the system further, resolving erroneous bonds that can take place during the transfer of labels and assignment of randomly initialized labels to found segments (see Fig. 4(F)). Our experiments showed that 20 - 60 Metropolis updating iterations are sufficient to obtain satisfactory segmentation results depending on the complexity of the scene.

The final result after equilibration for the given example is shown in Fig. 4(F). Only segments larger than a pre-defined minimum size are extracted, thereby small segments at borders of the blue cup and at edges of the big blue box formed due to reflections and changes in contrast are excluded (see Fig. 4(D,F)). The use of range data allows us to distinguish between objects having very similar color values like between the white moving object and the wall and between the blue cup and the big blue box (see Fig. 3).

3.3. Detection of new objects

Appearance of new objects is characterized by absence of pixel correspondences with the previous frames. As was mentioned before (see Section 3.2), those regions will be filled by random spin values except already assigned segment labels. After some iterations of the segmentation kernel new homogeneous regions will be covered by new labels. For those regions the kernel in the relaxation mode operates as initial segmentation (see Section 3.1).

3.4. Experimental environment

The proposed method was tested on a PC with Intel(R) core (TM) i7 3.33 GHz CPU (multiple cores) with 11.8 GB RAM using the Nvidia card GTX 295 (with 896 MB device memory) consisting of two GPUs for acceleration of the segmentation kernel and optical flow estimation.

4. Results

We present results of our method obtained for several depth/color movies acquired with the Kinect showing human manipulations of objects. The method is compared with another state-of-the-art video segmentation method. Our approach is evaluated in terms of the quality of the segmentation, coherency of the video segmentation, and computational speed.

4.1. Video segmentation results

Fig. 5 shows video segmentation results for the sequence "Moving an object" obtained without and with support of the depth data. The first and second rows show the original color frames and estimated optical flow for a few selected frames. The third row shows results obtained without usage of the range data. We can see that object tracking fails for fast moving objects. The optical flow method has a limit of 2 pixels per scale, so using 4 scales, the limit is $2^4 = 16$ pixels [13]. For this reason the white wooden object cannot be tracked along the whole movie and some of its parts are initialized improperly in frame 530 by the label taken from the background. It occurs due to the lack of pixel correspondences between adjacent frames. Such erroneous initializations cannot be resolved by the segmentation kernel. Note that both the moving object and the wall have in some frames very similar color values which make the tracking extremely difficult.

Including range data (shown in the fourth row) in the segmentation kernel and on the label transferring stage can resolve such problems (see Sections 3.1 and 3.2). Segmentation results of the same frame sequence derived with range-data support are presented in the last row. Fast moving pixels cannot be initialized by labels of pixels having range differences larger than τ (see (7)). In the current experiment, we used $\tau = 30$ cm. Similar pixels having large range differences do not tend to interact with each other (see (5)). Thereby the segmentation kernel can recover even poorly-initialized segments which makes the tracking of the fast moving white object consistent along the whole sequence.

Next, the segmentation results for a 2 min frame sequence of the sample action "Building a pyramid" are shown in Fig. 6. The first and second rows show original color frames with depth data from the Kinect. The third row shows segmentation results obtained by the proposed depth-supported video segmentation method. Results derived by the hierarchical graph-based video segmentation [7] at 90% and 70% of the highest hierarchy level [1] are shown in the fourth and the fifth row, respectively. Our method provides a temporally coherent video segmentation, in which all segments carry their initially assigned labels along the whole movie. For comparison, we show segmentation results for the same sequence obtained with a recent graph-based video-segmentation method [7]. Depending on the hierarchy level of the graph-based method, a coarser or finer segmentation is obtained. At coarse levels, merging problems leading to under-segmentation are observed, while at finer levels, more segments are formed, leading however to some temporal coherency problems.

To measure the quality of video segmentations we use the *segmentation covering* metric introduced by Arbeláez et al. (2009) [14]. The idea of the metric is to evaluate the covering of a human segmentation S', called also *ground truth segmentation*, by a machine segmentation S. A human segmentation, is a manual annotation of a video sequence showing how humans perceive the scene, whereas a machine segmentation is an output result of the considered video segmentation algorithm. For one frame, the segmentation covering metric is defined as

$$C(S' \to S) = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} O(R, R'), \qquad (11)$$

where N is the total number of pixels in the image, $|R|$ the number of pixels in region R, and $O(R, R')$ is the overlap between regions R and R'. The segmentation covering for the video sequence is computed by averaging the segmentation coverings over all frames in the sequence. Fig. 7 shows the performance of the system for frame sequence "Building a pyramid" as segmentation covering against the current frame number for frames $430 - 630$. As we can see, the color/depth sequence is segmented with high accuracy (the average segmentation covering value is 0.825).

4.2. Time performance

The total processing times, frame rates for various image resolutions are summarized in Table 1. The proposed method runs in real-time for medium image resolutions and can process video sequences of arbitrary length, while the graph-based video segmentation needs about 20 min to process a 40 sec video and only sequences that are not longer than 40 sec (with 25 fps) can be processed in the hierarchical mode [7].

resolution (px)	msec / frame	frame rate (Hz)
128×160	$9 - 17$	$111 - 59$
256×320	$21.5 - 39.5$	$47 - 25$
512×640	$72.5 - 145.5$	$14 - 7$

Table 1. Processing times and frame rates obtained for different image resolutions (for 20 - 60 iterations).

[1] The online version of the hierarchical graph-based video segmentation for 90% and 70% of the highest hierarchy level is available under *http://neumann.cc.gt.atl.ga.us/segmentation/*.

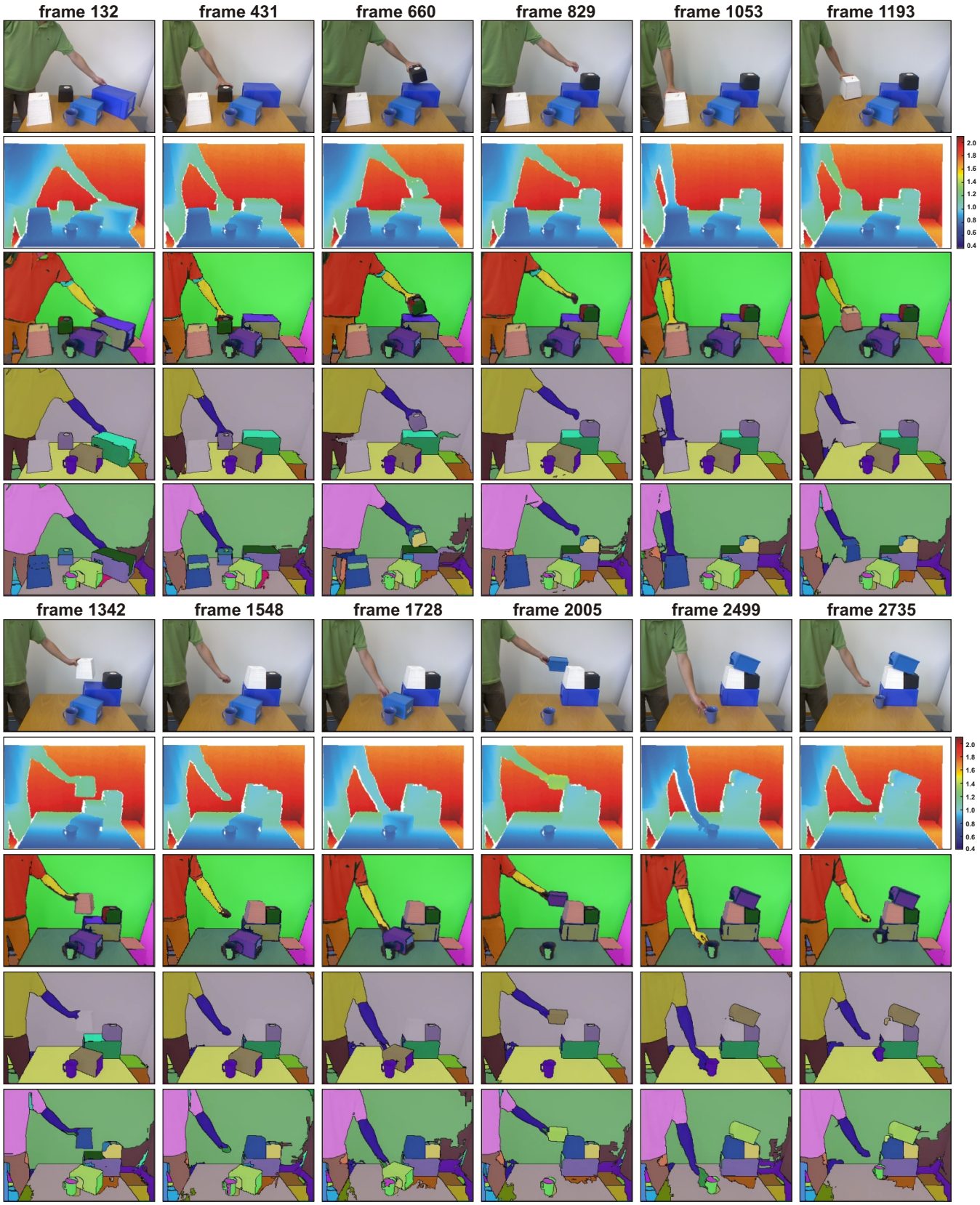

Figure 6. Results for frame sequence "Building a pyramid". Original frames and range data from the Kinect for selected time points are shown in the first two rows. The third row shows the segmentation results of our method. Graph-based video segmentation results obtained at 90% and 70% of the highest hierarchy level are presented in the last two rows, respectively.

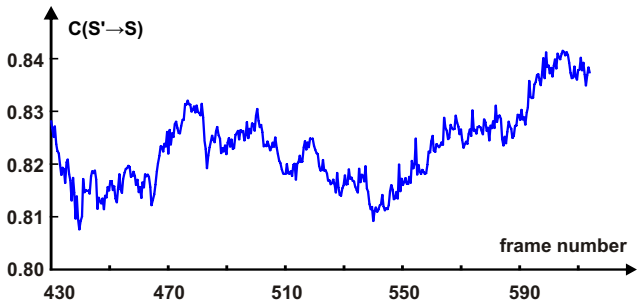

Figure 7. Segmentation covering for video "Building a pyramid".

5. Conclusion

In the current study we presented a novel real-time technique for the spatiotemporal segmentation of depth/color videos. The proposed method performs a homogeneous video segmentation, i.e. all objects visible in the scene carry the same unique labels along the whole video sequence. A Kinect device was used as a hardware setup for simultaneous real-time acquisition of color images and correspondent range data. The used image segmentation kernel based on the superparamagnetic clustering of data [9] exploits color/range data for consistent video segmentation. Usage of depth data makes it possible to track relatively fast moving objects by preventing interactions between pixels having significant range differences. Our method can be considered to be at match with the graph-based method [7] in terms of segmentation quality for the types of movies considered. However, some differences exist, which would have to be evaluated in more detail in the future. In terms of computational speed, we passed the graph-based method, which works at lower frames rates than ours. However, for complex actions and scenes, the coherency of the segmentation may be impaired due to the following problems: (i) objects are getting partly or completely occluded during the action, (ii) objects are getting joint/disjoint, (iii) objects move extremely fast, causing optical flow to fail. In the future, we aim to improve performance of the proposed method under these circumstances.

6. Acknowledgement

This research has received funding by the EU GARNICS project FP7-247947 and the EU IntellAct project FP7-269959. B.Dellen acknowledges support from the Spanish Ministry for Science and Innovation via a Ramon y Cajal fellowship. K. Pauwels acknowledges support from CEI BioTIC GENIL (CEB09-0010) of the MICINN CEI program. We thank Tomas Kulvicius for valuable discussions.

References

[1] H. Kjellström, J. Romero, and D. Kragic, "Visual object-action recognition : Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81–90, 2011. 457

[2] D. Rao, Q. V. Le, T. Phoka, M. Quigley, A. Sudsang, and A. Y. Ng, "Grasping novel objects with depth segmentation," in *IROS*, 2010. 457

[3] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research (IJRR), Special Issue on 'Semantic Perception for Robots in Indoor Environments'*, no. 30, pp. 1229–1249, 2011. 457

[4] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 800–810, 2001. 457

[5] I. Patras, R. L. Lagendijk, and E. A. Hendriks, "Video segmentation by map labeling of watershed segments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 326–332, 2001. 457

[6] D. Wang, "Unsupervised video segmentation based on watershed and temporal tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 539–546, 1998. 457

[7] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa, "Efficient hierarchical graph-based video segmentation," in *CVPR*, pp. 2141–2148, 2010. 457, 462, 464

[8] C. Wang, M. de La Gorce, and N. Paragios, "Segmentation, ordering and multi-object tracking using graphical models," in *ICCV*, pp. 747–754, 2009. 457

[9] A. Abramov, E. E. Aksoy, J. Dörr, K. Pauwels, F. Wörgötter, and B. Dellen, "3D semantic representation of actions from efficient stereo-image-sequence segmentation on GPUs," in *3DPVT*, 2010. 457, 459, 464

[10] J. Zhu, L. Wang, R. Yang, J. E. Davis, and Z. Pan, "Reliability fusion of time-of-flight depth and stereo geometry for high quality depth maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, pp. 1400–1414, 2011. 458

[11] A. Foley, J. D.and van Dam, S. K. Feiner, and J. F. Hughes, *Computer graphics: principles and practice (2nd ed. in C)*. Addison-Wesley, 1996. 459

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. of Chem. Phys.*, vol. 21, no. 11, pp. 1087–1091, 1953. 460

[13] K. Pauwels, N. Krüger, M. Lappe, F. Wörgötter, and M. Van Hulle, "A cortical architecture on parallel hardware for motion processing in real time.," *Journal of Vision*, vol. 10, no. 10, 2010. 460, 462

[14] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *CVPR*, pp. 2294–2301, 2009. 462