Active Segmentation in 3D using Kinect Sensor

Zoltan Tomori
Inst. of Experimental Physics
Slovak Academy of Sciences
Watsonova 47
040 01 Kosice, Slovakia
tomori@saske.sk

Radoslav Gargalik
Inst. of Computer Science
P.J. Safarik University
Jesenna 5,
040 01 Kosice, Slovakia
radoslavgargalik@gmail.com

Igor Hrmo
Inst. of Experimental Physics
Slovak Academy of Sciences
Watsonova 47
040 01 Kosice, Slovakia
hrmo@saske.sk

ABSTRACT

The combination of color image and depth map significantly improves the segmentation. The Kinect sensor with pan/tilt motorized movement captures both images and segments them separately by the Grab Cut method. The resulting contours are converted to polar coordinates. After the floor plane detection, corresponding "depth" and "color" contours are combined such that the importance of depth /color information is proportional to the distance from the floor. The segmentation is followed by the extraction of simple scale invariant features like color components and height/width ratio. Subsequently, features are used to train Normal Bayes Classifier. The algorithm was tested on a set of simple objects (mugs) on the table.

Keywords

Active segmentation, Kinect, Depth map, RGBD image

1. INTRODUCTION

Segmentation in 3D is a critical problem in many areas of computer vision. The information about the depth can be obtained by various methods like stereo vision, moving camera or object, defocusing, structured light or comparison with known geometrical model [Mir04]. Kinect - a low-cost 3D sensor for gaming console was launched in November 2010 and achieved big commercial success. Support for programmers appeared shortly after it (Microsoft Kinect SDK, OpenNI, OpenKinect, Freenect etc.). A more comprehensive source of information about Kinect hardware and programming was published only recently - e.g. [Web12].

Kinect provides 3D information which can be easily retrieved (color, depth, points cloud in real distance units). Its depth sensor consists of infrared transmitter/camera system. The transmitter projects small dots (speckles) on the surrounding scene and the IR camera acquires image and compares their position with the reference one. The depth is then calculated from the displacement of the individual speckles. The color + depth images are acquired approximately at the same moment and after their alignment they can be considered as one RGBD image.

The concept of active segmentation was inspired by the biological visual system where the object of interest is segmented with high resolution by the fovea, while the rest of the scene is captured in lower resolution on the periphery of retina [Mis09]. Another important concept is the "contact boundary" introduced in [Mis11] where the boundary pixels touching the surface (floor, desk) are distinguished from the remaining ones. Contact boundary is important in segmentation based on the combination of color image and depth map, both provided by Kinect.

2. ACQUISITION

Mechanical Construction

For testing purposes, we constructed a motorized equipment and we named it KATE (Kinect Active Tracking Equipment) - see Figure 1a). It consists of the Kinect sensor attached to the turntable driven by a precise stepper motor (Intelligent Motion Systems, USA). This allows horizontal movement around its vertical axis (pan) with the resolution 51200 steps per 360 degrees. For the vertical movement (tilt) we exploited the built-in Kinect stepper motor controlled via the same USB port as cameras. This solution is simple but it has a poor resolution in range <-31, 31> degrees. Another problem is that the tilt value is related to the absolute horizontal position measured automatically by Kinect accelerometer. However, this solution is sufficient for testing purposes.

a) **b)**

Figure 1. a) KATE (Kinect Active Tracking Equipment) b) Object as seen by Kinect. Central rectangle represents the region of interest (fovea) where the active segmentation is performed.

Calibration and tracking

Camera calibration based on the pinhole camera model is necessary in applications like objects reconstruction from multiple views. As Kinect consists of 2 cameras (RGB and IR), more complicated model is required [Paj11]. However, initial calibration included in OpenNI library is sufficient for segmentation and tracking purposes. The depth values correlate well with real values, the correspondence between the depth (Z) and XY dimensions can be easily corrected by scale constants.

We developed a supervised calibration function adjusting the scale in both horizontal and vertical directions. KATE watches the given scene displaying live image along with the cross in the image center $p_c(x,y)$ - see Figure 1b). The operator clicks a position $p_1(x,y)$ representing the desired future position of the image center. From p_c and p_1 values in pixels we can find corresponding 3D points $P_c=(0, 0, Z_0)$ and $P_1=(X_1, Y_1, Z_1)$ in real units (meters). Using simple trigonometry we obtain pan angle as $arcsin(X_1/Z_1)$ and tilt as $arcsin(Y_1/Z_1)$.

Calculated angles are converted to the stepper motor units and both motors move the given number of steps. In ideal case, the new central position $p_c(x,y)$ should show the same place of the scene as the $p_1(x,y)$ before the movement. If it is not the case, the scale factor can be adjusted. The calibration function allows manual control of the motors. The number of correcting steps is recalculated to the new scale factors for both pan and tilt values.

We created a face tracking system to test KATE [Sen12]. The new position of motors is not given by the click as described in the previous section, but it is determined by the face bounding rectangle (see Figure 2). Face detector is based on the Haar cascade classifier included in the OpenCV library. If a face is recognized (in the predefined depth range) then the displacement between the face rectangle and the center of the image is calculated. Pan/tilt motors correct the Kinect position trying to keep the face rectangle in the image center.

Figure 2. Face tracking. Kinect detects face by using Haar classifier included in OpenCV library. In the next step, pan/tilt motors adjust Kinect position such that the face rectangle is in the center of image.

Preprocessing

The depth image contains a lot of artifacts resulting from the depth measurement principle. Shadows-like defects appear in places visible from the depth camera but not illuminated by IR projector.

We exploited the "Inpaint" method described in [Tel04], which recovers the missing depth information. Implementation of this method is easy as it is included in OpenCV library.

3. PROCESSING

Processing consists of the following sequence of steps: plane detection, finding volume of interest and segmentation.

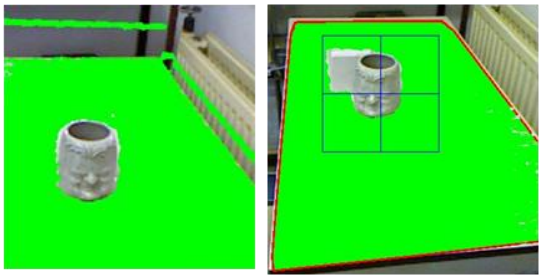
Floor Plane Detection and Region of Interest (ROI)

One of the basic operations in computer vision is the detection of the plane where the segmented objects are standing on (floor, table top). The detection is based on the popular RANSAC algorithm which finds the plane representing the input points cloud. The plane is determined by the equation

$$ax + by + cx + d = 0 \quad \textbf{(1)}$$

where $[a,b,c]$ is the normal vector and d is the distance from the origin.

Although RANSAC eliminates the influence of outliers in principle, it is possible to improve the plane detection by filtering the points that are definitely outliers. There are a lot of RANSAC modifications performing this task [Chu03]. It is possible to iterate plane detection and calculation of the volume of interest (described in the next section). Data for the next plane detection create only voxels from the volume of interest.



a) **b)**

Figure 3. Floor plane detection. a) RANSAC detects not only table-top voxels but also unwanted strips of voxels on the walls b) Region of interest represented by the top of the table (green) outlined by polygon (red).

Figure 3a) shows the detected plane (green pixels). As can be seen there, plane consists of the table top area as well as some undesirable green strips on the walls. We filtered them as follows:

- Erosion filter isolates table top from the connected strips (like this on the top right corner of table).
- Table top contour is found as the largest contour which includes center of image (fixation point).
- Convex hull function eliminates local discontinuities and gives the polygon outlining the table top (red in Figure 3b).

Finding the Volume of Interest (VOI)

The majority of applications focus on a limited space above the floor or the table-top. The bottom of this space is represented by the ROI detected in the previous step. Its top is given by the maximal expected height of our objects.

Limits from sides (walls, furniture) can be found by an algorithm based on a simplified model of the scene assuming that side limits are higher than segmented objects.

1) Project all voxels higher than the top limit to the floor plane.
2) Find the polygon with maximal area that does not include projected points. The number of vertices is given in advance.
3) Side limits of our volume of interest are created by planes perpendicular to the floor crossing the polygon sides.
4) Project voxels lying between the top and the bottom limits. If there are points falling inside the polygon and connected with its sides, algorithm returns to step 2.

Segmentation by Grab Cut

The algorithm exploited in our experiments is based on the idea of active segmentation briefly explained in the introduction. From a lot of possible segmentation methods we focused on these which are based on the classification of pixels inside a region of interest (ROI) and the minimization of an energy function. We assume that the segmented object is placed inside the square frame which is our initial region of interest.

GrabCut [Rot04] is a very popular segmentation algorithm based on the Gaussian Mixture Model and energy minimization. All pixels outside the ROI are labeled as background ones, pixels inside ROI are labeled according to the energy function consisting of regional and boundary terms. Briefly speaking, regional term reflects the likelihood that a given label is appropriate for the given pixel and the boundary term reflects how easily the label can expand to its neighborhood. Assigning a boundary label to a pixel inside a homogeneous region is penalized. The strength of N-link (the link between pixels *m* and *n*) is calculated

$$N(m,n) = \frac{\gamma}{d(m,n)} \exp\left(-\beta \left\| Z_m - Z_n \right\|^2\right) \quad \textbf{(2)}$$

$$\beta = \frac{1}{2\left\langle \left\| Z_m - Z_n \right\|^2 \right\rangle} \quad \textbf{(3)}$$

where Z_m is the color of the pixel *m*, constant γ has recommended value = 50, *d(m,n)* is the unit distance (1 for horizontal and vertical neighbors and *sqrt(2)* for diagonal ones). Value of β is the average inverse difference value calculated in advance.

Combined Segmentation

Kinect generates two images reflecting the same scene - color image and the depth map. Figure 4 shows the problem of color image segmentation if the other object with similar color is behind our object of interest. On the other hand, the depth map segmentation has troubles with parts near the floor where the depth of object and the background are almost the same.

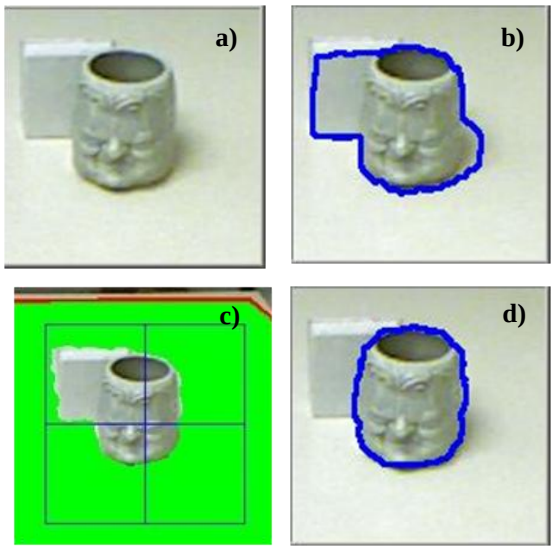


Figure 4. a) Overlapping objects with shadow on the right side. b) Segmentation by the GrabCut method exploiting only the color image. c) Floor plane detection (green pixels) d) Segmentation of depth image masked by the floor plane.

Above mentioned problems can be solved combining information from both - the depth map and color image. Several authors [Kar10], [Mut10] combine color and depth information. In [Sil11] a model was proposed which modifies computation of N-link as follows:

$$N(m,n) = kN_C(m,n) + (1-k)N_D(m,n) \quad \textbf{(4)}$$

where N_C and N_D are N-links from color image and depth image respectively and k controls the influence of both links to the final N-link calculation (e.g. 80% color and 20% depth). Modified N-link calculation is incorporated into the energy function optimized by graph cut method.

Distance Dependent Segmentation

Our approach is based on the assumption that the value of k from **(4)** is not constant but depends on the distance from the floor. In standard situations, the object standing on the floor is captured by Kinect under tilt angle. The top part of the object has a much higher contrast of the depth image than its bottom (Figure 5b). We combine the results of contours segmented from both color and depth images.

The initial step is a conversion of both contours from Cartesian to polar coordinates using the center of the image as the origin. The x-axis corresponds to the angle in the clockwise direction starting from the position "3" on the clock, y-axis is magnitude. The blue curve in Figure 5c corresponds to the contour from the color image and the green one to the depth image, the resulted red curve is their combination. After its conversion back to the Cartesian coordinates we can obtain a contour shown in Figure 5d.

Several alternatives exist how to change the color/depth influence, depending on the type and configuration of the objects on the scene.

a) Piecewise combination of segmented curves is the simplest method. The depth image contour representing the contact boundary is replaced by the corresponding part of the color image contour in a pre-selected interval. For instance, an interval <45,135> degrees represents the bottom of an object where a low contrast of depth image is expected. Selection of the end points of the interval should respect the continuity of resulted curve. Good candidates are intersections of the both curves.

b) Derivation of k on the distance from the plane using normal vector (1). As Kinect gives *(x,y,z)* values in the camera coordinate system, the calculation of the distance from the plane is straightforward.

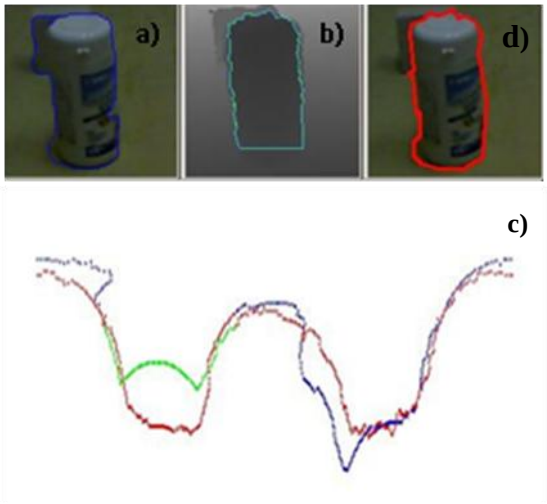


Figure 5. Segmentation by combined GrabCut method. a) color image b) depth map c) contours in polar coordinates d) result of segmentation

Classification of segmented objects

The successful segmentation is often the crucial step in computer vision (e.g. in objects recognition based on machine learning principles). We tested our system to recognize several simple objects (like mugs on the table). We exploited supervised learning based on the Normal Bayes Classifier. System segmented each object on the table and found the bounding rectangle of its contour as well as the average color. Feature vector consisting of 4 components (R, G, B color components and height/width ratio of bounding

rectangle) was used in the supervised learning stage taking cca 5 seconds. After the learning, the system was able to recognize object and display its label (Figure 6).

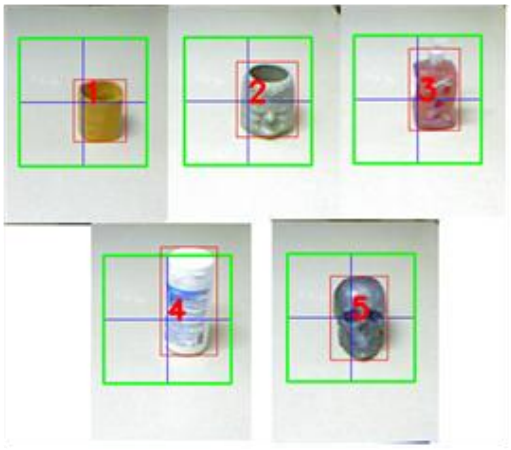


Figure 6. Classification of segmented objects using Normal Bayes Classifier based on the feature vector created by RGB color components and height/width ratio.

4. SUMMARY

Motorized pan/tilt Kinect system was constructed for the acquisition of color and depth images. This system tracks the object of interest keeping it in the center of image (tested with face tracking based on the Haar cascade classifier included in OpenCV library).

We applied Grab Cut method to segment both color and depth images using the image center as the fixation point. We transformed contours into the polar coordinates and combined them. The weights controlling the importance of color/depth edges was dependent on the distance from the floor detected by RANSAC method. This approach significantly improved segmentation near the floor as well as in partially overlapping objects. Segmented contours were used for the features extraction (R, G, B color components and height/width ratio). We used this features vectors for supervised training of Normal Bayes Classifier and for the classification of simple objects like mugs on the table.

Future work

This communication paper reflects our experiments with Kinect as the initial stage of the project oriented to application of Natural User Interface. We plan to exploit RGBD images for wider group of problems like active segmentation, tracking and control of specific devices. Growing number of papers combining color + depth along with the progress in sensors hardware make this research area very promising.

5. ACKNOWLEDGMENTS

This work was supported by the Slovak research grant agencies APVV (Project No. 0526-11) and VEGA (Project No. 2/0191/11).

6. REFERENCES

[Chu03] Chum, O. Matas, J. and Kittler, J. Locally Optimized RANSAC, Lecture Notes in Computer Sciences, 2781, pp. 236-243, 2003.

[Kar10] Karthikeyan, V. Anil, A. and Ebroul, I. GrabcutD: improved grabcut using depth information. Proc. ACM workshop on Surreal media and virtual cloning, Firenze, Italy, pp. 57-62 , 2010.

[Mir04] Mirzabaki, M. Depth Detection Through Interpolation Functions: A New Method. Proc. WSCG, Plzen, Czech Rep., pp. 105-108, 2004.

[Mis11] Mishra, A. and Aloimonos, Y. Visual Segmentation of "Simple" Objects for Robots. Proc. Robotics Science and Systems conference (RSS), Los Angeles, June 27 - July 1, 2011. http://www.umiacs.umd.edu/~mishraka

[Mis12] Mishra, A. K. Aloimonos, Y. Cheong, L. F. and Kassim, A. A. Active Visual Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34, pp. 639-653, 2012.

[Mut10] Mutto, C. D. Zanuttigh, P. and Cortelazzo, G. M. Scene Segmentation by Color and Depth Information and its Applications. Proc. Streaming Day, Udine, 2010.

[Paj11] Pajdla, T. Smisek, J. and Jancosek, M. 3D with Kinect. Proc. of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, Barcelona, Spain, pp. 1154-1160, 2011.

[Rot04] Rother, C. Kolmogorov, V. and Blake, A. "GrabCut" - Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics, 23, pp. 309-314, 2004.

[Sen12] Senaj, M. OpenCV library in computer vision applications in robotics, Master's thesis, FEI, Technical University of Kosice, 2012.

[Sil11] Silberman N. and Fergus, R. Indoor Scene Segmentation using a Structured Light Sensor.Proc. Int. Conf. on Computer Vision - Workshop on 3D Representation and Recognition, Barcelona, 2011.

[Tel04] Telea, A. An image inpainting technique based on the fast marching method. Journal of Graphics Tools, 9, pp. 23-34, 2004.

[Web12] Webb, J. and Ashley, J. Beginning Kinect Programming with the Microsoft Kinect SDK. Apress, (ISBN 978-1-4302-4104-1), 2012.