Improving Object Extraction With Depth-Based Methods

Fabian Prada, Leandro Cruz, Luiz Velho
Instituto de Matematica Pura e Aplicada - IMPA
VISGRAF Lab
Rio de Janeiro, Brasil
www.impa.br/∼{faprada, lcruz, lvelho}

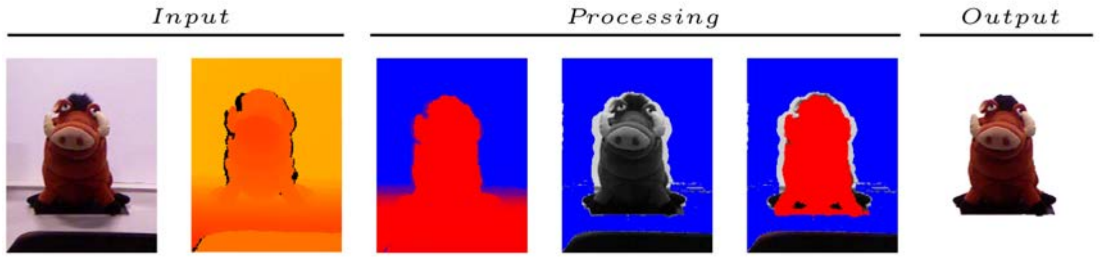


Fig. 1. Our inputs are the color and depth of a RGBD image. In the processing stage, the *Depth Range Estimation (DRE)* of the object, and the identification of *Background Planar Surfaces (BPS)* around it, allow to define accurate seeds. Therefore, a satisfactory segmentation is in general obtained.

***Abstract*—In this work, we introduce a method to do object extraction in RGBD images. Our method consists in a depth-based approach which provides an insight into connectedness, proximity and planarity of the scene. We combine the depth and the color in a GraphCut framework to achieve robustness. Specifically, we propose a depth-based seeding which reduces the uncertainty and limitations of the traditional color based seeding. The results of our depth-based seeding were satisfactory and allowed good segmentation results at indoor environments. An extension of our method to do video segmentation using contour graphs is also discussed.**

***Keywords*-Object Extraction; GraphCut; RGBD.**

I. Introduction

Recent developments of real-time depth sensors have provided a completely new framework to comprehend and interact with indoor environments. Structural informations obtained with depth data have allowed significant enhance in tasks such as human pose recognition [1], indoor scene segmentation [2], object labelling [3], among others. All these works aboard each problem from a machine learning perspective, where the accessibility to high quality databases is fundamental. In this paper we present a method to perform object extraction from raw color and depth data. Thus, our work could support recognition tasks by providing appropiate trainning inputs.

Object extraction is a basic segmentation problem, which can be stated as follows: *Given sets of object and background seeds, identify automatically the set of pixels which belong to the referenced object.* Applications of object extraction can be found in the fields of medical image analysis, background removal in teleconferences, image and video edition at industrial and entertainment applications.

Classical approaches to object extraction are only based in the color data. The main topic of our research is the extension of a color based method to include depth information.

One of the motivations to explore depth-based methods concerns in the increasing of accessibility to low cost and satisfactory quality depth sensors. The RGBD images evaluated in this work were captured using Microsoft Kinect Sensor [4]. Kinect provides good accuracy in depth measure and a frame rate of 50 fps for RGBD video. Currently this sensor have been used in many researches [5].

Because of its good performance and versatility, we chose GraphCut [6] as a reference framework for object extraction. Therefore, a short overview of GraphCut will be presented in Section II.

In Sections III and IV we introduce our object extraction technique, which integrates three depth-based methods using GraphCut. Each one of these methods provides different geometric information about the scene content used to improve the seeds of segmentation.

In Section V we introduce an extension of our object extraction method for videos. In this method we apply the object extraction (as it will be explained to image case) in the first frame. For all other frames, we adapt the previous frame boundary to seed the method.

Section VI is dedicated to explain implementations details and Section VII to discuss our results. Finally, in Section VIII we present the conclusions and future works.

A. Related Work

Object extraction has been widely studied in RGB images [7]. On the other hand, object extraction in RGBD images was hardly explored due to technical limitations on attaining real time and high quality depth maps. Most works on RGBD images combines deterministic methods with appropiate heuristic to deduce structural information from depth data. This approach is precisely followed in this paper.

By deterministic methods, we mean the ones that are able to identify if some input (e.g., depth map) satisfies some property. In this paper, we will show two deterministic methods: one to recognize connected components and other to identify planar surfaces.

Kahler et al. [8], as well as Cobzas and Zhang [9] introduced some deterministic methods to identify planar patches. The first one uses a segmentation model based on GraphCut technique, and it shows robust results by fusion of color and depth. Our technique for extraction of planar patches is quite different, and simpler than that introduced by Kahler et al. [8]. However, we also use a GraphCut model, and we still achieve satisfactory results. The work of Cobzas and Zhang [9] focus on manipulation of depth captures of indoor environments. As in their work, we use intensity information to improve segmentation on regions of noisy depth measures. We follow a different approach since we deal speciffically with objects.

On the other hand, heuristic methods are based on reasonable assumptions (also called *priors*) about the image content. Silberman and Fergus [2] classify objects in 13 possible categories: bed, wall, table, etc., defining certain priors based on location and depth. Our work proposes three priors which have completely different nature. Their priors are designed for a fixed set of large size objects. Ours are designed for medium and small size objects, and are stated in a general way to be object independent. Despite of these differences, both methods exemplifies the importance of depth/location assumptions to get robust results.

The object extraction approach used in this work is an extension of TofCut [10]. In this work the authors evaluate people extraction in video sequences at indoor environments using a GraphCut framework. The authors deal with scenes in which objects and background are non-adjacent. Altogether, their approach has obtained a satisfactory result representing depth models with single Gaussian distributions. The method we propose in this paper takes account the adjacency case which is harder, and requires more structured information than raw depth data.

Another relevant work in RGBD images was introduced by Lai et al. [11]. In this work the authors propose a method to detect and recognize small to medium size objects using depth and color data. Our work has a different objective: extraction instead of recognition. Our method could be viewed as complementary to Lai's method. Their task of object labeling could be assisted by our technique.

In the field of object extraction in RGB videos we can cite as relevant works the introduced by Bai et al. [12] as well as by Li et al. [13]. Our approach is based in Markov Random Fields in the same way that both of these works. The method proposed by Bai et al. covers the object with several windows defining local color and shape models, and introducing a temporal coherence term in the energy function. Li et al. represent the video as a 3D volume, and propose an energy function that includes a term for adjacent regions in consecutive frames.

These approaches are more elaborated than ours, and they constitute the state of the art in object extraction for RGB video. On the other hand, these works still require a lot of user interaction, not only in the initialization, but making corrections into subsequent frames. Beside this, our data have more structure than theirs. Our aim is not comparing the quality and versatility of our results to these reference works. Instead, we intend to demonstrate how the use of depth information improves the extraction task through reduction of user interaction and preserving quality.

Nowadays, the RGBD images have been used in a lot of research, not only improving techniques related to object extraction. Cruz et al. [5] introduced a survey related to projects using the data provided from Kinect sensor. We believe that this kind of sensor will be more common and with a better quality. In this way, the use of the depth data together the color will be more common in various research areas.

B. Technique Overview

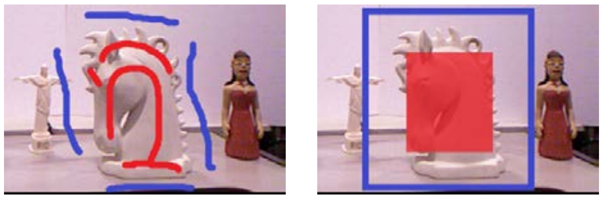
In this article we introduce three depth based methods, each providing different insights to the geometrical composition of the image. The first method introduced, called *Depth Connected Component (DCC)*, is a morphological algorithm exploring the depth neighborhood of a specific region.

The second method, called *Depth Range Estimation (DRE)*, is a clustering algorithm that identifies those pixels which are closest to the camera and are probable object pixels.

The third method, called *Background Planar Surfaces (BPS)*, is a more structured algorithm that estimates the normal at each pixel of the image and use this information to identify the planar surfaces around the object.

We propose a new way to define *Foreground (FG)* and *Background (BG)* models based only on seeds chosen from depth information. It is the main contribution of our work. Traditional ways of choosing seeds from color information [14] are restricted by two limitations: the object should not intersect the boundary of the selection region and the color distribution of *BG* and *FG* should not overlap. These limitations are overcome using the structural information gained from depth data. Consequently, better seeds and a more robust model construction is obtained through our approach.

Once the models are constructed, the next step is defining the energy function to be minimized. Here, we extend the traditional GraphCut energy function to include a *Depth Data Term (DDT)* and *Depth Smoothness Term (DST)*. These new terms conduces to improve the object extraction task as confirmed from the experimental results.


(a)

(b)

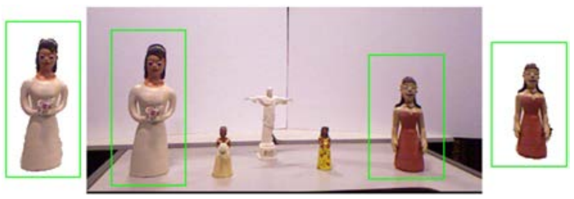

(c)

Fig. 2. (a) Seeds explicitely defined by the user (b) *Rectangular Selection*, *Selection Border* (blue pixels), and *Selection Center* (red pixels). (c) Results from our method using only *Rectangular Selection*.

II. PRELIMINARIES

A. GraphCut

The objective of GraphCut is minimizing an energy function (1) over the set of all possible pixel labeling. To achieve this, GraphCut induces a graph structure on the image. Each pixel is represented by a vertex, and there are two additional vertices associated to the *FG* and *BG* labels. The graph has two kind of edges. Edges between label vertices and pixel vertices are called *t-edges* and weights the likelihood of the pixel to the *FG* and *BG* models. Edges between neighbors pixels are called *n-edges* and weights their similarity. The total weight of *t-edges* and *n-edges* active at a certain cut is measured by the *Data Term* (2) and *Smoothness Term* (3) of the energy function. In the following equations x_p represents the label of pixel p, χ_L is the indicator function of label L, and z_p is the RGB color of pixel p.

$$E(x,z) = \alpha_C U_C(x,z) + \gamma_C V_C(x,z) \tag{1}$$

$$U_C(x,z) = -\sum_p \chi_B(x_p)\log \mathbb{P}(z_p|B) + \chi_F(x_p)\log \mathbb{P}(z_p|F) \tag{2}$$

$$V_C(x,z) = \sum_{p,q\in N} \frac{[x_p \neq x_q]}{|p-q|}\left(e^{-\beta_C||z_p-z_q||^2}\right) \tag{3}$$

The general formulation of GraphCut has led to applications in tasks such as image segmentation, stereo reconstruction and image restoration. This general formulation provided us a fertile terrain to include depth data in object extraction.

B. User Interaction

FG and *BG* models are constructed from special set of pixels called seeds. Seeds can be explicitly defined by the user or be deduced from a set of *priors* applied to a *Rectangular Selection* (Figure 2).

The method for object extraction which is proposed in this article relies exclusively on seed generation from *Rectangular Selection*. This means, once the *Rectangular Selection* is specified, the seeds are generated automatically based on a set of depth based *priors*.

III. STRUCTURAL INFORMATION FROM DEPTH DATA

A. Depth Connected Component (DCC)

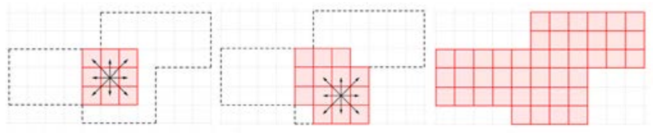
An immediate application of depth data for image segmentation is the identification of *Depth Connected Components*. We say two arbitrary pixels are *depth connected*, if there is a path joining them, where depth difference between consecutive pixels is smaller than a fixed threshold (20mm). *Depth connectedness* induces a partition of the image pixels, and we call each of these components a *DCC*. From this last observation we formulate our first *prior*:

Prior 1: *The object to be extracted is completely contained in a DCC.*

Identifying the *DCC* is a natural idea for object extraction whenever the object is not adjacent to any other element of the image. In such cases the information gained by identifying the *DCC* of any pixel of the object, is almost a perfect object extraction (Figure 3). The discrepancies are due to distortions on the depth data provided by the sensor.

In the adjacency case, the relevant structural information obtained from the *DCC* is reduced, and it is null when the image is a unique *DCC* component. In such cases, a reasonable alternative is to define a second threshold for the depth range, i.e., identify the *DCC* inside of a sphere of certain radius. This alternative depends on the image content and object of interest. So we discarded it from our general-purpose extraction method.

DCC identification can be easily implemented through a Priority Search Algorithm. In this work it is implemented as *Breadth First Search (BFS)* [15].


(a)

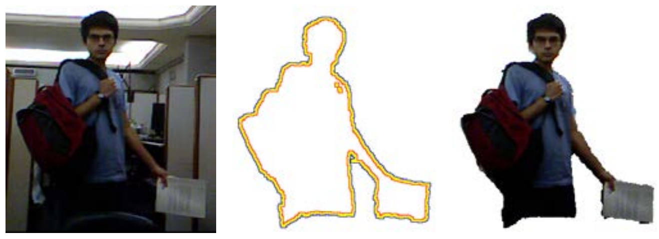

(b) (c) (d)

Fig. 3. a) *BFS*. b) *Rectangular Selection* of the object. (c) *contour graph*. (d) *DCC* of a pixel in the object.

B. Depth Range Estimation (DRE)

The objective of this technique is identifying an interval in the depth histogram that closely contains the object (Figure 4). In order to identify such interval, we adopted the following *prior*:

Prior 2: *Closest pixels in the Selection Center are highly probable object pixels.*

This assumption is supported by two reasonable hypotheses. First, the *Rectangular Selection* is almost centered in the object. Second, the object is not occluded in the *Selection Center*. From the set of pixels in the *Selection Center*, we applied *K-means* to identify n clusters of depth. Clusters associated to the object are identified as follows:

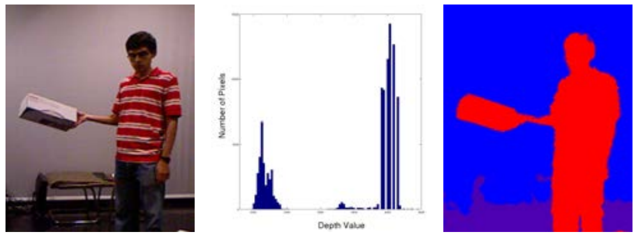
1) Sort the n clusters from closest to farthest. Let $d_{(1,2)}, d_{(2,3)}, ..., d_{(n-1,n)}$ be the distance between the center of consecutive clusters.
2) Label the closest cluster as FG.
3) If $d_{(1,2)} < d_t$ the second cluster is labeled FG. Otherwise finish.
4) For $i \in \{3, 4 \ldots n\}$, if $d_{(i-1,i)} < m_t d_{(i-2,i-1)}$, the i-th cluster is labeled FG. Otherwise finish.

The previous procedure depends on two parameters: d_t and m_t. Parameter d_t defines a threshold between the closest cluster and the second closest. The first cluster is always assumed to belong to the object, so the inclusion of a second cluster depends on its proximity to the first. Since Kinect sensor is designed for environments from 1 to 5 meters depth, it is expected that the object to be extracted is contained in this depth interval. We also should assume that object's depth range is at most 2 meters, which is generally true for any object fully contained in the scene. From these observations and from our experiments, we got as a satisfactory selection for general applications taking 5 clusters and $d_t = 0.5$ meters.

Parameter m_t defines a threshold for the distance between pairs of consecutive clusters. If we have already identified clusters $i-2$ and $i-1$ as belonging to the object, we found that comparing $d_{(i-1,i)}$ to $d_{(i-2,i-1)}$ is a reasonable criterion to decide if we include cluster i in the object. From these observations and experimental results we chose $m_t = 1.5$.

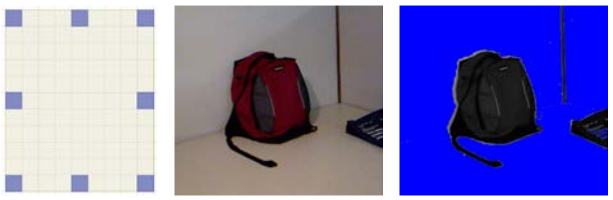
C. Background Planar Surfaces (BPS)

DCC and *DRE* have a problem in common: when there is adjacency between object and background we cannot gain any distinction using just raw depth data. To overcome this difficulty we require more structured information about the image. This is precisely achieved estimating its normal map. The environment around the object in indoor spaces is frequently conformed of planar surfaces. This is the case whenever the object is over the floor, in front of a wall, etc. From this observation we formulate our third *prior*:



(a) (b) (c)

Fig. 4. (a) *Rectangular Selection*. (b) Depth Histogram. (c) *Depth Range Stimation*



(a) (b) (c)

Fig. 5. (a) *Component's Generators*. (b) *Rectangular Selection*. (c)*Planar Components*

Prior 3: *Planar pixels near Selection Border are highly probable background pixels.*

This idea is captured in the next procedure:

1) Take a sample of pixels from the *Selection Border* and store them in a queue. These pixels will be called *component's generators* (Figure 5(a)).
2) Fix tolerance parameters e_n for normal and e_z for color.
3) Pick the first pixel p of the queue and identify the *depth connected* set of pixels around p satisfying $||n_p - n_x|| < e_n$ and $||z_p - z_x|| < e_z$. Call this set of pixels $PN(p)$, the planar neighborhood of p.
4) In order to confirm $PN(p)$ as a valid plane the following two conditions must hold:
   - $PN(p)$ is greater than 5% of *Rectangular Selection*.
   - At most 25% of $PN(p)$ belongs to the *Selection Center*.
5) If the previous conditions holds, $PN(p)$ is a *planar component*, and we label its pixels as *planar pixels*.
6) Repeat the process using the next *component's generator* in the queue.

We defined normal (e_n) and color (e_z) thresholds to guarantee the planarity and uniformity of the *planar components*. Since any pixel in a component is compared with its *generator*, this assures that deviation inside a component is at most e_n and e_z respectively.

To validate the planarity of a component, we must verify size and location conditions. Since normal and color variation between neighbor pixels is usually small, always we run the region growing step on a non-planar pixel of the queue, we will still find a small neighborhood around it satisfying the normal and color thresholds. To avoid classify such components as planar surfaces we required that the size of the neighborhoods be at least 5% of the *Rectangular Selection*. This is a reasonable threshold, which is almost satisfied in scenes where planar surfaces are relevant. The location condition sets a threshold for the amount of pixels that a planar component could have in the *Selection Center*. This avoids that planar surfaces belonging to the object, which also intersects the *Selection Border*, be misclassified as background surfaces.

IV. Integration with GraphCut

A. Depth and Color Models Construction

The most popular implementation of GraphCut for the specific task of object extraction in RGB images is GrabCut [7]. GrabCut constructs color models to estimate the color distribution of *FG* and *BG* pixels. The color distributions are approximated by Gaussian Mixture Models.

In a first attempt to extent GraphCut to involve depth data, it seems reasonable to define depth models as Gaussian Mixture Models. This approach was intended by the authors but results obtained were not satisfactory.

In the case of color models, the color variance usually has a similar order in the set of *BG* and *FG* pixels. Instead, in the case of depth data, the common situation is that variance in the set of *BG* pixels is much larger than in *FG* pixels. This problem is slightly relieved by adding more Gaussian components to the *BG* model, but still the variance problem is so pronounced that is difficult to find an energy function which leads to satisfactory results.

The method we propose here is to pick seeds according to the *priors* introduced in the previous section. The seeding process is performed in two steps:

1) *Pixels Scoring*. From methods *DRE* and *BPS* we define functions W_{FG} and W_{BG} that weight the penalty we have to pay to label certain pixel as FG or BG seed.
2) *Seeds Refinement*. The *FG* seeds defined by the previous score function could belong to several *DCC*. Select the largest subset of *FG* seeds that belong to a same *DCC*, and reaffirm these pixels as *FG* seeds. The pixels initially identified as *FG* seeds but not belonging to the chosen *DCC* are then moved to the set of *BG* seeds.

In the first step we define the penalties as follows.

- *Penalties from DRE* : Let m and M be the minimal and maximal depth value in the set of pixels belonging to the *FG* clusters, $\mu = \frac{m+M}{2}$ and $\sigma = M - m$. For all pixel p we set $W^1_{FG}(d_p) = 1$, and,

$$W^1_{BG}(d_p) = \begin{cases} 1.2 & \text{If } d_p \in [m, M] \\ \max(1.2 - 0.2\frac{|d_p-\mu|}{\sigma}, 0) & \text{Otherwise} \end{cases}$$

  This penalty function says FG is preferred to BG for pixels in the depth interval $I_{FG} = [\mu - \sigma, \mu + \sigma]$. As the pixel's depth move aside this interval, the BG penalty decreases to 0 while the FG penalty remains in 1.

- *Penalties from BPS*: Since *planar pixels* are assumed to be non-object pixels, we impose a penalty for the FG label. We define $W^2_{BG}(p) = 0$, for all p, and

$$W^2_{FG}(p) = \begin{cases} 1 & \text{If } p \text{ is a planar pixel} \\ 0 & \text{Otherwise} \end{cases}$$

Total Scores: Set $W_{FG} = W^1_{FG} + W^2_{FG}$ and $W_{BG} = W^1_{BG} + W^2_{BG}$. The seeds are preliminary chosen as follows:

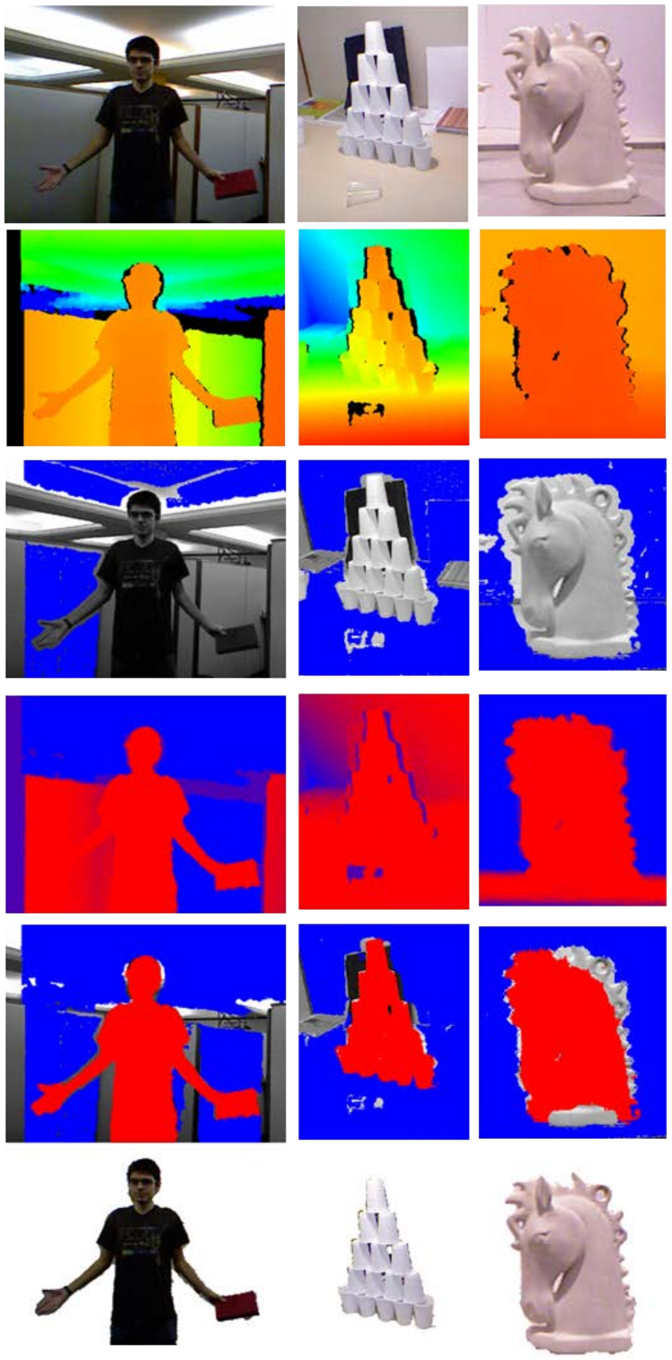


Fig. 6. In Top Down order: Rectangular Selection, Raw Depth, Depth Range Estimation; Background Planar Surfaces; Seeds and Final results.

- If $W_{BG}(p) - W_{FG}(p) > 0$ Then p is a *FG* seed.
- If $W_{FG}(p) - W_{BG}(p) > 0.2$ Then p is a *BG* seed.

The previous selection corresponds to the criteria:

- FG seeds are all non *planar pixels* in I_{FG}.
- BG seeds are all *planar pixels* and pixels in

$$I_{BG} = (-\infty, \mu - 2\sigma] \cup [\mu + 2\sigma, \infty) .$$

B. Energy Function

The energy function we evaluate is a extension of GraphCut energy function (1) that additionally contains a *Depth Data Term (DDT)* and *Depth Smoothness Term (DST)* . The *DDT* is precisely the penalty function defined above:

$$U_D(x,d) = \sum_p \chi_B(x_p) W_{BG}(x_p,d_p) + \chi_F(x_p) W_{FG}(x_p,d_p) \quad (4)$$

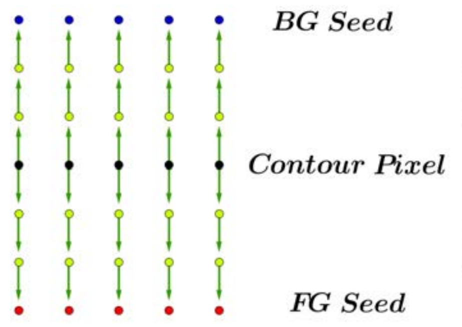
The *DST* is selected to avoid cuts in regions of almost constant depth. Parameter 600 in the following equation penalizes cuts in regions of low depth variation (up to 20mm), and it is not significant for depth discontinuities (above 50mm):

$$V_D(x,d) = \sum_{p,q \in N} \left(1 - \frac{(d_p - d_q)^2}{600 + (d_p - d_q)^2}\right) \quad (5)$$

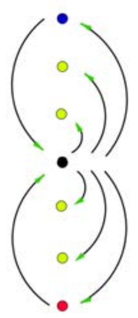
The energy function we implemented for object extraction in RGBD images, in its general formulation, is given by

$$E(x,z,d) = \alpha_C U_C + \gamma_C V_C + \alpha_D U_D + \gamma_D V_D, \quad (6)$$

Here U_C and V_C are the same to (2) and (3) respectively. Observe we have four control parameters $\alpha_C, \gamma_C, \alpha_D$, and γ_D which measures the importance given to each term.

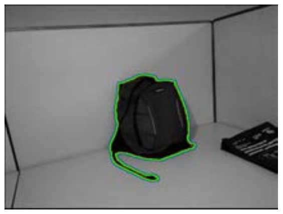


(a)

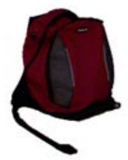


(b)

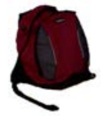
(c)



(d)



(e)



(f)

Fig. 7. (a) *Graph Expansion*. (b) *Local Model Construction* (c)(d) Frames with *countour graphs*. (e)(f) Segmentation results.

V. Object Extraction in RGBD videos

Once the object has been extracted from an initial image, we now face the problem of keeping track of it in subsequent frames. Kinect provides real-time RGBD videos that allow enhance this task using depth information. The method we propose for object extraction in videos follows these steps:

- *Initialization:* The object is extracted from a single frame. This is done through the method described for RGBD images.
- *Graph Expansion:* From the current segmentation, call *contour pixels* those having both object and non-object neighbors. Initialize a queue with the *contour pixels* and perform a 4-neighborhood *BFS*, this is, traverse the queue adding the unvisited 4-neighbors of the current pixel to the end of the queue. This procedure let us explore a band of certain radius around the *contour pixels*. This band will be called the *contour graph* (Figure 7.a).
- *Seeds Locations: BFS* makes each pixel in the *contour graph* to be descendant of one and only one *contour pixel*. For each *contour pixel* we are going to construct a local model based on its descendants in the boundary of the *contour graph*. If a descendant is in the *contour graph's* boundary and belongs to the object it is marked as *FG seed*. Similarly, if a descendant is in the *contour graph's* boundary but does not belong to object it is marked as *BG seed*.
- *Graph Mask:* The *contour graph*, *contour pixels*, and *seeds* obtained on the already segmented frame, are all translated to identical positions in the next frame. Now, we will refer as *contour graph*, *contour pixels*, and *seeds* to the respective pixels in the non segmented frame.
- *Local Model Construction:* We are ready to construct the local models based in the information of the non segmented frame. For each *contour pixel* the *FG* color and depth model are given by the mean color (z_{FG}) and mean depth (d_{FG}) of its *FG seeds*. *BG* color model and depth model are defined analogously. [1]
- *Energy Function - Color and Depth Terms:* The likelihood of a pixel to belonging to the object is evaluated from the local model of its *contour pixel* parent. The *CDT* is given by:

$$U_C(x,z) = \sum_p \sum_{L \in \{BG,FG\}} \left(\frac{\chi_L(x_p)|z_p - z_L| MC(p)}{|z_p - z_{FG}| + |z_p - z_{BG}|} \right),$$

where $MC(p)$ is a constant that measures the magnitude of the color difference in the model associated to p:

$$MC(p) = \frac{|z_{FG} - z_{BG}|}{|z_{FG} + z_{BG}|}.$$

Similarly, the *DDT* is:

$$U_D(x,d) = \sum_p \sum_{L \in \{BG,FG\}} \left(\frac{\chi_L(x_p)|d_p - d_L| MD(p)}{|d_p - d_{FG}| + |d_p - d_{BG}|} \right),$$

$$MD(p) = \frac{|d_{FG} - d_{BG}|}{d_{FG} + d_{BG}}.$$

[1]*Graph Expansion* by *BFS* does not guarantee that all *contour pixels* will have *FG* and *BG* seeds associated to it. In such cases the *contour pixel* points to the models of a close neighbor.

The *CST* and *DST* are defined as (3) and (5) respectively.

- *Energy Function - Temporal Term:* To make extraction robust it is necessary take account temporal coherency. Our Temporal Terms penalizes changes in the label of a pixel, when difference of color and depth in consecutive frames are below very small thresholds ϵ_C and ϵ_D.

Let $z_{p,t-1}$ and $z_{p,t}$ be the color of pixel at position p in frames $t-1$ and t. Suppose segmentation in frame $t-1$ was already done, i.e., $x_{p,t-1}$ is known for all p. The *Temporal Color Coherency Term (TCCT)* is given by:

$$T_C(x_t) = \sum_p \mathbf{1}((x_{p,t-1} \neq x_{p,t}) \wedge (|z_{p,t-1} - z_{p,t}| < \epsilon_c)).$$

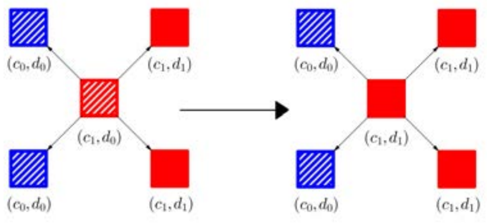
Similarly, the *Temporal Depth Coherency Term (TDCT)* is given by:

$$T_D(x_t) = \sum_p \mathbf{1}((x_{p,t-1} \neq x_{p,t}) \wedge (|d_{p,t-1} - d_{p,t}| < \epsilon_d)).$$

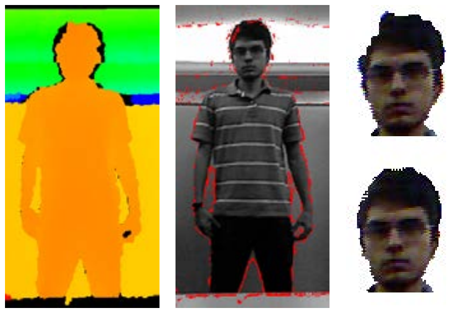
Our energy function consists of six terms: *Data, Smoothness* and *Temporal Coherence* for both color and depth. To each term we associate a control parameter:

$$\begin{aligned} E(x_t, z_t, d_t | x_{t-1}, z_{t-1}, d_{t-1}) = \alpha_C U_C + \gamma_C V_C + \delta_C T_C \\ + \alpha_D U_D + \gamma_D V_D + \delta_C T_D \end{aligned}$$

- *Segmentation:* The minimization of the energy function produce a partition of the *contour graph* in *FG* and *BG* pixels. From this partition we update the global segmentation to the current frame. Take the global segmentation of the previous frame as basis, and label the *FG* pixels of the *contour graph* as object. Label the *BG* pixels of the *contour graph* as non-object. Return to the *Graph Expansion* step to process the next frame.



(a)



(b) (c) (d)

Fig. 8. (a) Filter scheme (b) Depth data with pronounced shadows (blank spaces). (c) Pixels where depth value was updated. (d) Detail of a segmentation using *DCC*, without the filter (top) and with the filter (below).

VI. Implementation

A. Preprocessing

Since Kinects depth sensor and camera are located at distinct positions, shadows problems are quite common. This is specially observed at object edges.

Figure 8 shows a way to overcome the shadows problem. Our implemented approach filters the depth using information of the color data. The idea is to fill the large blank spaces, and correct erroneous measures using color similarity of the image. The proposed filter algorithm works as follows:

1) Fix thresholds for depth (e_d) and color (e_c). Here we used $e_d = 10mm$ and $e_c = 40$.
2) For each pixel p in the image, compare its depth value to the four pixels located 2 positions away in diagonal directions.
3) If there is a high variation between d_p and at least two diagonal pixels (i.e. $|d_p - d_q| > e_d$) , we proceed to compare the color values of p and the four diagonal pixels.
4) The depth value of p is updated if the diagonal pixels satisfy $|d_p - d_q| > e_d \iff |z_p - z_q| < e_c$. In this case, the we set d_p as the mean depth value of the similar color diagonal pixels (i.e those satisfying $|z_p - z_q| < e_c$).

B. Acquisition and Processing

The color data was stored and processed in 24 bit RGB image. The depth data was processed as 16 bit unsigned integer, allowing millimetrical precision in a range from 1 to 5 meters.

The code for *DCC, DRE* and *BPS,* as well as the code for models, energies, and graphs was written in MATLAB.

The energy function minimization was done using Olga Veskler et al. code [16][17][18]. This package provides a MATLAB mex file to implement α-expansion algorithm.

C. Parameters Adjustment

- *Image Object Extraction:* The energy function as defined in (6) mixes a *Color Data Term* (2) and a *Depth Data Term* (4) whose ranges have completely different scales. In order to have both *CDT* and *DDT* in a similar scale, we normalized the *CDT* to sum one at each pixel. The *CDT* we implemented is given by:

$$U_C(x) = -\sum_p \frac{\chi_B(x_p) \log \mathbb{P}(z_p|B) + \chi_F(x_p) \log \mathbb{P}(z_p|F)}{|\log \mathbb{P}(z_p|B) + \log \mathbb{P}(z_p|F)|}$$

After this normalization the range of *CDT* is in [0,1]. This is similar to the range of *DDT*, which is [0,2]. Both *CST* (3) and *DST* (5) also belong to the range [0,1].

To our experiments, the set of parameters $\alpha_C = 2$, $\alpha_D = 1$, $\gamma_C = 0.25$, $\gamma_D = 0.02$ gave satisfactory results in multiple contexts (Figure 6).

- *Video Object Extraction:* To perform *Graph Expansion* step we must specify a radius for the *contour graph* around the *contour pixels*. For general applications we chose a radius of nine pixels. For the segmentation task a satisfactory set is $\alpha_C = 1$, $\alpha_D = 2$, $\gamma_C = 0.1$, $\gamma_D = 0.01$, $\delta_C = 1$, $\delta_D = 1$.

VII. Results and Discussion

A. Depth Based Methods

The combination of the three depth based methods, introduced in Section III, provided the required information to define appropriate seeds. In many cases, the *BG* and *FG* seeds almost corresponded to the desired segmentation.

Depth Connected Component was the first approach to segmentation from depth data in the non-adjacency case. Using *DCC* as a direct method of segmentation, lead to satisfactory results (Figure 3(d)), but this depends on having accurate depth information. Kinect produces noisy depth values and large blank spaces, so any segmentation method based only in depth data is not dependable.

Depth Range Estimation is a method that explores depth statistics to deduce the object size and location. Dividing the set of depth values in a fixed number of clusters and choosing some of them based in proximity criteria, gave us good results in many contexts. However, identifying the object from the depth histogram is a difficult problem, and our method was not so accurate at some other situations. When the object have a non-uniform depth range (i.e., there are sudden changes in the separation of consecutive depth clusters), the method does not work well, and it discards some clusters that belong the object. An attempt to fix this problem is to change the number of clusters or the proximity criteria. Instead, we recommend to define a threshold for the amount of pixels that the identified *FG* clusters must have. For instance, defining that the *FG* clusters must contain at least 30% of the *Selection Center* could improve results.

Background Planar Surfaces gave robust results in almost all the experiments done. We obtained very good segmentation of object when it was surrounded of planar surfaces. In the case of non-adjacency, *BPS* had some difficulties to completely identify the planar surfaces around the object. This happened when the planar surface was not adjacent to the selection boundary, when the planar surface did not satisfy the required size threshold, or when depth noise introduced errors in normal calculation. Despite of these limitations, *BPS* still contributed with valuable information.

B. Object Extraction

Satisfactory segmentation results were obtained in the evaluated cases. The quality of segmentation specially depended on having appropriate seeds and energy function.

Seeds selection was characterized by the accuracy of *DRE* and *BPS*. For some images they provided complementary information, and in others, one of them provided all the required information to define good seeds. Cases of inaccurate seeding occurred when *DRE* was unable to identify the farthest depth clusters of the object.

The energy function we proposed depends on four control parameters. Defining a set of parameters that works well in different scenarios is a hard task. Results in Figure 6 used the set of parameters specified in Section VI. As observed from these results, we obtained satisfactory segmentations in scenarios with adjacency, non-adjacency and overlapping of color distributions, using the same set of parameters. Figure 9 shows a comparisson between a color-based segmentation (b) and our approach (c) using also depth data.

Since the seeding process is done using depth data, we recommend assigning larger weight to color terms in the energy function. So, the segmentation is not biased only by depth, and color information allow to correct error produced by depth noise. We followed this criterion for the parameter set selection, and this led the segmentation process to be more robust as a whole.

The selection of better parameters for specific scenes is a tradeoff process. Giving more weight to certain terms of the energy function improves the segmentation at some regions of the image while affecting others. We identified two main aspects that must be taken into account to deal with specific cases: scene content and depth noise.

The scene content criterion considers the proximity between object and background, as well as the similarity of color distributions. According to the image structure, depth could provide more valuable information for segmentation than color, and *vice versa*. On the other hand, the depth noise criterion looks for correctly classify the regions with blanked depth data using color information. Emphasizing the *CST* improved segmentation results in depth noisy regions.

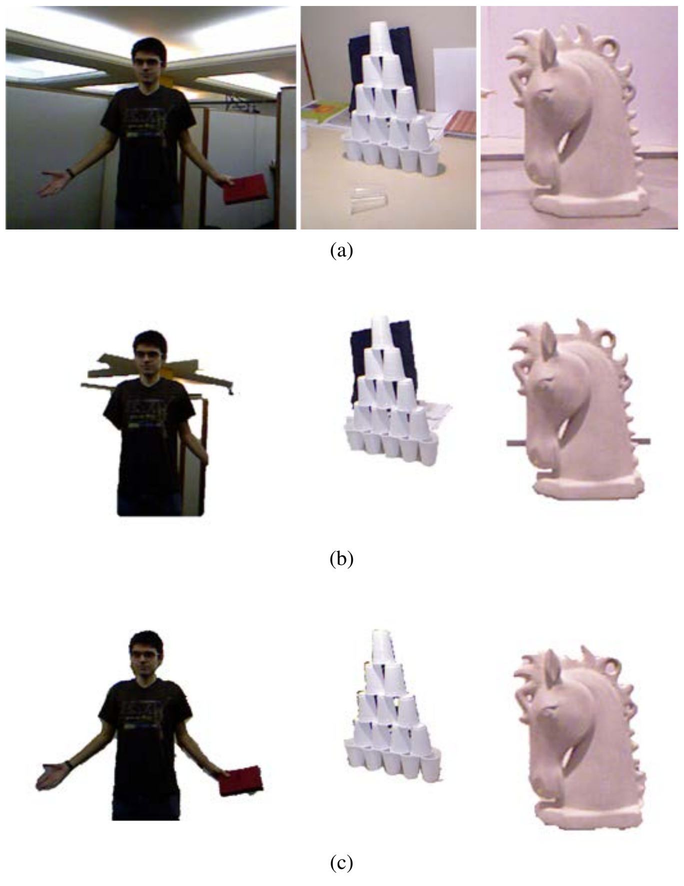


Fig. 9. (a) Input Images (b) Result obtained only applying GraphCut in RGB channels. (c) Result using our method for RGBD channels.

C. Video Extraction

The method we propose for video extraction lead to good results in the cases of width objects and slow camera-object movements. In such cases fair sequences of frames where correctly segmented with low error rates (300 frames in the case Figure 7). Our method still have limitations and it is specially sensitive to error propagation. Despite of this, it is computationally simple and avoid excessive user interaction.

Video extraction also has to deal with parameter adjustment. Again, setting good parameters for general cases is a difficulty task. We proposed a set of parameters that worked fine for some evaluated scenes, but certainly, better results could be obtained by replacing them according to the specific case. In general, we recommend giving to the *DDT* the greatest value to effectively track the object movement, and also give a significant weight to the *CST* to get sharper contours for the object. The temporal coherence terms are also valuable since they avoid visual artifacts such as sudden pixel twinkling.

VIII. Conclusions

Depth information is a valuable tool to improve the object extraction task. Through depth based methods we can easily discover aspects of the scene geometry that are hardly attainable from color data. In this article we combined three depth methods in a GraphCut framework. They allowed designing a more robust seeds selection process, and defining a more flexible energy function. We stand out the efficiency of *BPS* on the task of identifying planar surfaces around the object.

Seeding process was robust and segmentation results were satisfactory. Regarding to the energy function, there is still work to do on parameter fitting. Some strategies that we consider worth for a future work are learning the parameters from an overview of the scene structure, and allowing the parameters to change locally.

The method of object extraction in video produced fair results under certain conditions. It is not as competitive as [12] or [13], but it is favored by its simplicity and reduced user interaction. The novelty of our method consist on using depth information to improve the tracking. Our method could be improved by adaptively changing the width of the *contour graph* and defining a restart strategy to deal with error propagation. Classical techniques like SIFT descriptors, shape models, or optical flows could also be adapted. All of this is an interesting subject for future research.

Acknowledgment

We would like to thank the CNPq, by funding the first and second authors of this article, and the VISGRAF Lab, at IMPA, for providing an environment for a fruitful research.

References

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition*, 2011.

[2] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *3DRR Workshop, ICCV*, 2011.

[3] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3d scenes," in *ICRA*, 2012.

[4] "Microsoft kinect." [Online]. Available: http://www.xbox.com/en-us/kinect

[5] L. Cruz, D. Lucio, and L. Velho, "Kinect and rgbd images: Challenges and applications," in *Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012*, 2012.

[6] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," in *International Conference on Computer Vision*, 2001.

[7] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," in *SIGGRAPH*, 2004.

[8] O. Kahler, E. Rodner, and J. Denzler, "On fusion of range and intensity information using graph-cut for planar patch segmentation," in *Intelligent Systems Technologies and Applications*, 2008.

[9] D. Cobzas and H. Zhang, "Planar patch extraction with noisy depth data," in *International Conference on 3-D Digital Imaging and Modeling*, 2001.

[10] L. Wang, C. Zhang, R. Yang, and C. Zhangand, "Tofcut: Towards robust real-time foreground extraction using a time-of-flight camera," in *Fifth International Symposium on 3D Data Processing, Visualization and Transmission*, 2010.

[11] K. Lai, L. Bo, X. Ren, and D. Fox., "A large-scale hierarchical multi-view rgb-d object dataset," in *Proc. of International Conference on Robotics and Automation*, 2011.

[12] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video snapcut: robust video object cutout using localized classifiers," in *ACM SIGGRAPH*, 2009.

[13] Y. Li, J. Sun, and H. yeung Shum, "Video object cut and paste," in *ACM Transactions on Graphics*, 2005.

[14] C. Rother, V. Kolmogorov, Y. Boykov, and A. Blake, "Interactive foreground extraction using graph cut," in *Technical Report*.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms." The MIT Press, 2001.

[16] O. Veskler, "Max-flow/min-cut v3.01." [Online]. Available: http://vision.csd.uwo.ca/code/

[17] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[18] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.