Active 3D Scene Segmentation using Kinect

CARLOS HERNÁNDEZ MATAS

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2011

Active 3D Scene Segmentation using Kinect

C A R L O S H E R N Á N D E Z M A T A S

Master's Thesis in Computer Science (30 ECTS credits)
at the Systems, Control and Robotics Master's Program
Royal Institute of Technology year 2011
Supervisors at CSC were Niklas Bergström and Carl Henrik Ek
Examiner was Danica Kragic

TRITA-CSC-E 2011:103
ISRN-KTH/CSC/E--11/103--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Abstract

This thesis takes as a starting point a segmentation framework for visual scenes based on Markov Random Fields. The approach integrates several cues and considers the scene in 3D, improving the hypotheses in an iterative manner.

In this thesis, we substitute the hardware previously used for a Kinect, a device capable of creating high quality depth maps of the environment.

The main interest of this work is to study the capabilities that the exploitation of said map can offer to the segmentation framework.

Specifically, we focus on the provision of an alternative attention mechanism to select fixations points for the segmentation, as well as two sets of cues. All of it based exclusively in the depth data retrieved from the Kinect.

The first set of cues is based in the detection of edges in said depth data, while the second set relies on the lack of depth information in specific areas.

Keywords: image segmentation, markov random field, kinect.

Contents

List of Figures

Chapter 1

Introduction

Image segmentation is the task of partitioning an image into consistent regions. It is known to be one of the hardest problems studied in computer vision. The problem is naturally ill posed, since there is no right or wrong partitioning. The accuracy of a segmentation is highly dependent on the context in which the segmentation is being used. This means that there is a need to introduce some sort of bias in the system that makes it prefer certain segments to other.

This bias should be different depending on the purpose of the system. For instance, should a person be covered by one segment, or should different parts like the eyes and the mouth have their own segments? This entirely depends on whether we want to find people in a scene, or locate people's eyes.

General purpose segmentation algorithms, as we will see in Chapter 2 have been developed, and in certain cases they may produce sufficiently good results by themselves.

However, due to the ill posed nature of the segmentation problem, in many cases such general segmentation techniques have to be combined with knowledge of the application domain in order to provide a desirable solution.

Image segmentation has been applied to a large range of different application domains:

For example, in the health field, medical imaging can be used to diagnose tumors or illnesses such as Alzheimer [1].

Similarly, it has been used in the agricultural world to diagnose specific diseases on crops [2] or to determine if a fruit is in its optimal state to be collected [3].

In the automotive sector, useful applications are for example the detection and recognition of traffic lights [4] and road signs [5]. Also, traffic monitoring systems [6] can benefit from this kind of image processing.

Other areas, such as satellite imaging take advantage of image segmentation techniques in order to perform automatic detection of different kinds of terrains [7].

Image processing has been proven to be very useful in manufacturing industries, allowing to quickly perform quality analysis and searching for defects for example on the fabrication of devices [8] or in the industrial painting of surfaces [9].

Image segmentation is also present in some devices that we may use in a daily basis. One of those examples is face recognition in pictures [10], which we can find in most of the commercial cameras sold nowadays. Although in this case, it is not purely segmentation, but also object detection and recognition.

In the context of robotics and manipulation [11], *object* segmentation is of interest. In the case that the robot has no knowledge about the objects in front of it, it still needs to hypothesize about what in the scene constitutes an object in order to manipulate them.

In this case we need to introduce a bias telling the system to prefer segments of whole objects. This is very hard in the case that we have no knowledge of objects and rely solely on data given by an image.

The work presented in this thesis will focus on the problem of segmenting unknown objects in a scene, telling them apart from surfaces and background.

1.1 Report outline

Chapter 2 gives an introduction to the evolution of image segmentation through diverse methods.

Chapter 3 sets the starting point for this work, describing the methodology it is based on, as well as the hardware differences between our approach and the previous work, and how this affects the method.

Chapter 4 presents the innovations that are introduced in this thesis, together with the theoretical motivations behind them and how they are integrated into the existing framework.

Chapter 5 shows the results of the work performed. We present the outcomes of the innovations standalone, combined between them and also in combination with the current methodology.

Chapter 6 discusses the results of the experiments and presents conclusions on the novelties introduced in this thesis.

Chapter 7 hints new lines of research that can be pursued using the work here presented as a starting point.

Chapter 2

Background

The image segmentation problem has been widely studied since a few decades back and is still a hot research topic. Currently a lot is being published on semantic segmentation [12], i.e., partitioning an image into different regions based on their semantic meaning, such as “road”, “sky”, or “building”. These exclusively work with outdoor images, where the scene structure is rather restricted: sky is on the top of the image, road in the bottom and buildings and trees in the middle. In an indoor scene there is typically no such structure, and there is a need to resort to other solutions.

Another popular stream within the segmentation community is interactive segmentation [13]. In this case a user indicates directly in an image where the object is, e.g. by drawing a box around it, or selecting a point in the object. The segmentation is executed, and given the resulting segment, the user has the possibility to further refine the segment. As the name suggests, a human has to be present and directly interact with the image, something that is undesirable in a robotics scenario.

The integration of several different cues in the same system has been progressively gaining more presence in the image segmentation methods, although it has been mostly using cues extracted from color images [14]. Lately the addition of cues extracted from disparity, depth and location data is gaining more relevance [15], adding extra dimensions to the process.

In [16], a method that segments objects in a scene was presented. It gets hypotheses about object locations from an attention mechanism [17], and uses these locations as seeds for the segmentation. By using disparity information in addition to image data, the method has the possibility to segment heterogeneously colored objects. Further, by assuming that the objects are supported by a surface, it has the possibility to distinguish object parts close to the surface.

However, the ability to capture an entire object with aid of disparities can also be a drawback, since the method often captures two objects if they are in close

proximity. Therefore it is important to investigate how other cues can be used in order to disambiguate in these situations.

The disparities in [16] were generated from a stereo camera setup designed to create a disparity map of the scene, but not a high quality map. In recent time we have witnessed quick development in depth and ranging sensors, not only in terms of quality but also in terms of accessibility.

One of the products of this quick development is PrimseSense Kinect [18], an affordable device which provides very dense high quality depth maps in combination with standard color images.

2.1 Image segmentation

During these decades addressing the segmentation problem, many approaches have been used [19]. We will now quickly review a few important segmentations methods, before doing a deeper analysis of Markov Random Fields, which is the method of interest for this thesis.

2.1.1 Methods

Histogram thresholding is one of the methods that appeared first. It is a 2-label approach in which pixels are classified depending on their position in the histogram of the image. The threshold may be chosen by different techniques, like knowing a priori the size of the object, thus, approximately how many pixels it will include, finding the peaks and valleys on the histogram or by applying it recursively to clusters in the image [21].

This method provides a fair result when trying to, for example, segment an object from its background. Although there are requirements for it, such as the colors be clearly differentiated and as plain as possible. Textured objects and backgrounds tend to provide faulty segmentations.

One approach to segmentation is through *edge detection*, a well studied area of image processing. Through a method, such as the Canny edge detector [20], we obtain the edges present due to sharp intensity changes in the image. While this can be useful to segment images in which all the elements have plain colors and no shadows, a scene consisting of textured elements will be challenging.

A more advanced version is *K-means clustering* [22], where the image is segmented into K regions. The process is performed by choosing initial mean values of certain features, such as colors or intensities, for the regions, classifying the pixels in the image according to their closest mean, then recomputing the means of the new classes and reclassifying every pixel, until there is no change in the mean values.

However, histogram thresholding, and to a lesser extent K-means clustering as it depends on the features used, suffer from neglecting spatial coherence. That is,

they do not attend to the dependency between neighboring pixels, which may lead to noise in the segments of the image. Region based segmentation tries to solve this issue.

One way to address this is with *region growing* [23]. This is achieved by building a region adjacency graph (a graph in which costs are associated with both nodes and arcs), and then merging similar neighboring regions with each other. This similarity can be determined by different characteristics, such as mean intensities or statistical distributions.

A different solution is *split-and-merge* [24]. This method takes an image and, if the relevant features are not homogeneous in a specific area, recursively partitions it into four squares. Once finished, similar squares are merged into bigger regions, until all the regions have homogeneous features and every region has different features than its adjacent regions..

But both region merging and split-and-merge lack a mechanism to ensure that the offered solution is globally the best one. That is, the solution obtained may be the best one from a local point of view, but not necessarily a global one.

In the following section we will review a segmentation method that uses a mechanism to ensure that the outcome provided is close to the best one given the segmentation rules used.

2.1.2 Markov Random Fields

A local optimal solution is a solution that is considered the best within a neighboring set of solutions. On the other hand, a global optimal solution is the best among all possible solutions.

In the cases in which we obtain local best solutions, the properties of the method are hard to analyze, as flaws in the solution may be due to limitations in the process. However, the segmentation can be analyzed in a more thorough manner for methods that offer a globally optimal solution, as in this case the segmentation flaws will come from the definition of the cost function.

To apply an appropriate cost function and to obtain that solution that is close to the global optimum, we will look into the application of *graph-theory*, and specifically into *Markov Random Fields* (MRF) [25], which we will explain in more depth than the methods previously commented.

A Markov Random Field is an undirected graph that describes a set of random variables for which future states do not depend on any past state, but just on their present state.

A structure may be represented as a MRF using nodes, which are interconnected between them. If the nodes are to be classified into different labels, a set of rules is needed to be able to perform that classification. Those rules are translated into

energy functions, which assign different weights to the links in a pixel-to-pixel and a pixel-to-label basis.

Then, as we will see now in an example more directly connected to image segmentation, a method to classify the nodes based on the link weights is used.

In a typical image segmentation problem, we want to differentiate important elements on a scene from the rest of said scene. In this example, we use one important element or 'object'. We denominate the rest of the scene as 'background'.

Each node in the MRF can be labeled in one of those two groups. Seed points are used as hard constraints, that is, they indicate which points have to necessarily be part of each label, providing clue to the system on what is supposed to be segmented.

The cost function, which is formulated in terms of boundary and region properties of the segments, is then used as a soft constraint, that is, as a segmentation guideline, to calculate the global optimum of all the possible segmentations that could satisfy the hard constraints.

$$E(A) = \lambda\, R(A) + B(A) \tag{2.1}$$

$$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p) \tag{2.2}$$

$$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}}\, \delta(A_p, A_q) \tag{2.3}$$

In (2.1) we can see a typical cost function. $A = (A_1, ..., A_p, ..., A_{\mathcal{P}})$ is a binary vector whose components A_p specify assignments to pixels (nodes) p in $\mathcal{P}$. The nodes have a neighborhood system represented by a set of $\mathcal{N}$ pairs $\{p, q\}$. In that equation, λ indicates the importance of the region properties $R(A)$ compared to the boundary properties $B(A)$.

The regional properties (2.2) takes for granted that the penalties for assigning a pixel p to either object and background are known, and reflects on how the value of a specific feature or set of features of a pixel p fits into a known model of object and background.

On the other hand the boundary properties (2.3) determine the penalty for discontinuities between pixels p and q (pair-wise term). When their features are similar, the penalty is larger than when they are different. This is due to the fact that neighboring pixels are likely to belong to the same object. This term is used for smoothness in the boundaries, but it is not a requirement for the method.

As shown by Boykov and Jolly, for performing the analysis of the image, it is mapped into an undirected graph (MRF), so that every pixel corresponds to a node. Also, we have a terminal for each possible label, and every terminal is represented by an extra node.

The resulting structure is a graph in which each pixel node is connected to its closest neighbors (*n-links*), but also to each of the terminal nodes (*t-links*).

To finish defining the graph, weights calculated with the energy functions are assigned to every link. In the case of graph cuts [25]., this weights can never be negative. The minimum value will be zero, which is reserved for the value of the t-link connecting a seed point with the corresponding label it was assigned to.

After the cut process is complete, every node will keep only one t-link. Every n-link connecting nodes that are t-linked to different terminals will also be cut. As a result, it will be obtained one subgraph for each label (terminal).

The desired cut is the one that has the minimum cost. And that cost is calculated from the weights of the severed links. So the method always tries to cut the links with the minimum cost, but following the guidelines indicated in the previous paragraph.

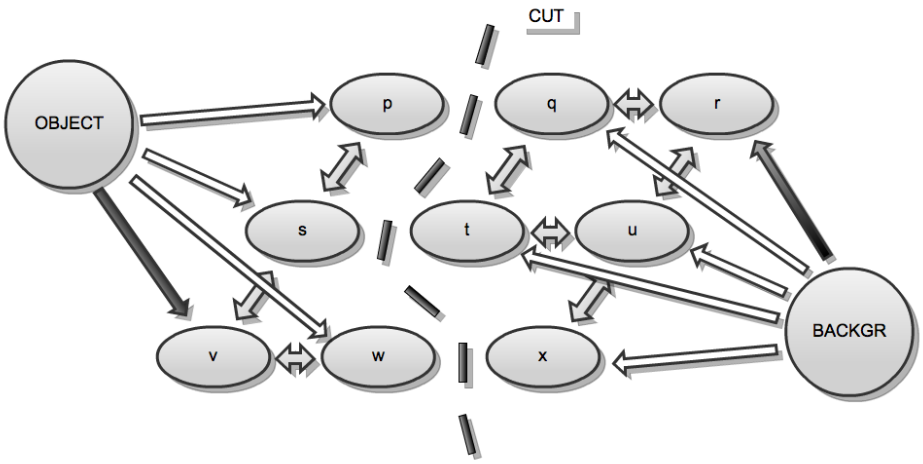


Figure 2.1. Markov Random Field

In Figure 3.1 we can observe a graphical representation of the graph after the segmentation. 'Object' and 'Background' are the terminal nodes, while 'p' – 'x' are the nodes representing the pixels. The grey bidirectional arrows represent the n-links, while the black arrows are the hard constrains of the t-links, and the white links the other t-links.

Due to the proximity to a global optimal solution in the results provided by the application of cost functions (in this case, energy functions) to Markov Random Fields, as well as the efficient adaptability of the method to the interactive addition of new seeds or even labels, we consider that this is the most efficient way to approach the image segmentation problem.

Chapter 3

Setup

One typical failure is when the system over- or under segments objects, i.e., fails to capture the whole object or captures more than the object. This project will investigate different methods for detecting and correcting these situations. In particular it will investigate if and how the higher quality disparity maps generated from a PrimeSense Kinect can be utilized for the problem.

In this chapter, we will describe the starting point of this work, both in terms of hardware and methodology.

3.1 Method

The corner stone of this thesis in terms of methodology is the segmentation framework presented by Björkman and Kragic. In this section we will make a brief description of the method. For further details, please refer to [16].

3.1.1 Segmentation

This is a framework, based on Markov Random Fields, which we explained in Chapter 2.1.2, that separates three kind of elements from a scene:

- foreground (physical object of interest),
- flat surface,
- background (every point that is not in either of the other categories).

For this purpose, it makes use of binocular disparity and color information.

The points in the image are denoted by a set of image measurements $\mathbf{m} = \{m_i\}$, with $m_i = (p_i, c_i)$. $p_i = (x_i, y_i, d_i)$, where x_i and y_i are the coordinates of the point,

and d_i the value of its disparity. $c_i = (h_i, s_i, v_i)$, with h_i, s_i and v_i being the values of the color in the point in HSV space, that is, its hue, saturation and value.

Binocular disparity extracts the depth information of certain elements from a set of 2-dimensional images in which those elements appear. I.e., it is the way the brain is able to reconstruct a 3D scene from the information provided by the eyes.

HSV is used instead of RGB because it makes easier to identify colors. Hue would indicate the color, saturation the amount of color, and value its brightness.

The element of the scene to which a point belongs is represented by a label, and each element has its own model which is based on a set of parameters θ. The sets of parameters are composed by the distributions of the point positions, disparities and colors. They are given by

$$\theta_f = \{p_f, \Delta_f, c_f\}$$
$$\theta_s = \{d_s, \Delta_s, c_s\}$$
$$\theta_b = \{d_b, \Delta_b, c_b\}$$

The goal of the framework is to find the most likely parameter set θ and distribution of labels $\mathbf{l} = \{l_i\}$ given the measurements $\mathbf{m}$. That is, to identify to which element (foreground, surface or background)best models each pixel.

The parameters for the scenes are modeled as follows:

- Foreground: Distributions of the point coordinates and disparity are modeled using a 3D Gaussian. Points with undefined disparities are computed ignoring that dimension. The color distribution is represented using a 2D histogram modeling the hue and saturation.
- Surface: Distribution of the point coordinates is descried by a uniform distribution. Distribution of the disparity is a Gaussian distribution. The color distribution is represented as a 2D histogram created from the hue and saturation.
- Background: Distribution of the point coordinates is a uniform distribution. Distribution of the disparity is a Gaussian distribution. The color distribution is represented as a 2D histogram created from the hue and saturation.

This give us the joint measurement conditionals as

$$p(m_i|\theta_f) = n(p_i; p_f, \Delta_f)H_f(h_i, s_i)$$
$$p(m_i|\theta_s) = N^{-1}n(d_i; \alpha_s x_i + \beta_s y_i + \delta_s, \Delta_s)H_s(h_i, s_i)$$
$$p(m_i|\theta_b) = N^{-1}n(d_i; d_b, \Delta_b)H_b(h_i, s_i)$$

where N is the amount of pixels of the image and N^{-1} represents the uniform distribution of the coordinates. $d = \alpha_s x_i + \beta_s y_i + \delta_s$ represents a plane in (x, y, d).

Also, $n(x; \bar{x}, \Delta)$ is used to reference a Gaussian distribution of a d-dimensional variable x with mean $\bar{x}$ and covariance Δ, as indicated by the following formula

$$n(x; \bar{x}, \Delta) = \frac{1}{\sqrt{(2\pi)^d |\Delta|}} \exp^{-\frac{1}{2}(x-\bar{x}^{\mathrm{T}}\Delta^{-1}(x-\bar{x}))}$$

The model parameters θ are estimated by calculating the maximum likelihood estimation of $p(\mathbf{m}|\theta)$. For that, the labels in $p(\mathbf{m}, \mathbf{l}|\theta)$ are treated like hidden variables, and eliminated via marginalization as in (3.1).

Maximum likelihood estimation is a way to estimate the parameters of a model parameters given a specific statistical model. It is done by selecting values of the model that create a distribution that gives the largest probability to the observed data. In this case, the parameters θ are estimated from the measurements $\mathbf{m}$.

$$p(\mathbf{m}|\theta) = \textstyle\sum_l p(\mathbf{m}, \mathbf{l}|\theta) \tag{3.1}$$

The following functions show the joint probability of $\mathbf{m}$ and $\mathbf{l}$ given θ (3.2). This joint probability is computed through the measurement distribution (3.3) and the prior label probabilities (3.4). In (3.3), I_i^x is 1 if $l_i = l_x$ and 0 if $l_i \neq l_x$.

$$p(\mathbf{m}, \mathbf{l}|\theta) = p(\mathbf{m}|\mathbf{l}, \theta)\, p(\mathbf{l}|\theta) \tag{3.2}$$

$$p(\mathbf{m}|\mathbf{l}, \theta) = \prod_i p(m_i|\theta_f)^{I_i^f} p(m_i|\theta_b)^{I_i^b} p(m_i|\theta_s)^{I_i^s} \tag{3.3}$$

$$p(\mathbf{l}|\theta) = \prod_k p(l_k) \prod_i \prod_{j \in N_i} p(l_i, l_j) \tag{3.4}$$

To calculate the maximum likelihood estimation of $p(\mathbf{m}|\theta)$ for estimating the model parameters θ, the algorithm tries to maximize a function $Q(\theta|\theta')$ that guarantees that given a previous estimate of θ', (3.1) increases.

For that, $Q(\theta|\theta')$ is expressed as the expected value of $\log p(\mathbf{m}, \mathbf{l}|\theta)$ with relation to the conditional distribution $p(\mathbf{l}|\mathbf{m}, \theta')$ under the previous estimation of θ' as in

$$Q(\theta|\theta') = \sum_{\mathbf{l}} p(\mathbf{l}|\mathbf{m}, \theta') \log p(\mathbf{m}, \mathbf{l}|\theta)$$

Then, the model parameters θ are updated through the maximization of $Q(\theta|\theta')$. This is achieved by recomputing $p(l_i|\mathbf{m}, \theta)$ for each label by means of loopy belief propagation [26].

But to be able to do that, the equations have to be expresses as energy functions suitable for belief propagation. Using Bayes' rule, and knowing that m_i depends

uniquely on l_i, then

$$p(\mathbf{l}|\mathbf{m},\theta) = \frac{p(\mathbf{m}|\mathbf{l},\theta)p(\mathbf{l}|\theta)}{p(\mathbf{m},\theta)} \\ = \frac{\prod_k p(m_k|l_k,\theta)p(l_k)}{\prod_k \sum_{l\in L} p(m_k|l_k=l,\theta)} \prod_i \prod_{j\in N_i} p(l_i,l_j) \tag{3.5}$$

The optimization is performed for a fixed amount of iterations unless the result converges. It is calculated by considering the net of image points as a Markov Random Field (MRF) and using belief propagation to minimize the energy for each label in each point. In (3.5), the first factor represents cliques of one point each, and the second the involving pairs of points. This captures spatial continuity, penalizing solutions with discontinuities.

The energy functions are given by the negative logarithms of the factors in (3.5). If we set a constant penalty to label two neighboring pixels differently, and no penalty if they are labeled differently, we can model the joint probabilities of two adjacent points using the Potts model [27], [28].

$$p(l_i,l_j) = \exp^{-V_{i,j}C_i^j} \tag{3.6}$$

were C_i^j equals 1 if $l_i \neq l_j$ and 0 if $l_i = l_j$.

Then, for calculating V, a pairwise penalty based on the difference in luminance between points, similar to the ones presented in [25] and [29] is used

$$V_{i,j} = 50\exp^{-\beta\,(v_i-v_j)^2} \\ \beta = (2\langle(v_i-v_j)^2\rangle)^{-1} \tag{3.7}$$

Where β is a normalization term, with $\langle\cdot\rangle$ denotes the expectation over an image.

As we indicated in Chapter 2.1.2, the pairwise energy function sets the penalty for performing the cut of the image into different labels, translating the characteristics of the segmentation we want to perform into the framework. Due to this, that is one of the parts of the framework to which we will focus our attention.

Furthermore, to take advantage of the consistency over time, instead of calculating the maximum likelihood estimation of $p(\mathbf{m},\theta^t|\theta)$, it is calculated for $p(\mathbf{m},\theta^t|\theta)$, where θ^t are the parameters estimated during the previous frame, which now are considered as measurements. But we will not focus in that, as it is beyond the scope of this work.

3.1.2 Initialization

The system is initialized through a rough segmentation, considering only pixels with valid disparity information. Assuming that the system is fixated in the foreground

object, points lying inside a 3D ball are obtained and assigned to the foreground model.

From the rest of the image, through random sampling, the parameters of a plane $d = \alpha_s x_i + \beta_s y_i + \delta_s$ are calculated, deprecating the planes that are not horizontal enough and selecting the one that includes more image points, which will be assigned to the surface model. The rest of the points will be assigned to the background.

The reasons explained in Chapter 2.1 to choose a technique of this nature, along with the experimental results presented in [16], place Björkman and Kragic's method as the ideal starting point for the work on this thesis.

3.2 Hardware

As indicated in Chapter 2, the hardware used in this work is Kinect [18], an indoor device developed by PrimeSense for Microsoft to use with Microsoft Xbox 360. The device is composed by an RGB camera, 3D depth sensors, a multi-array microphone and a motorized tilt. From those, only the RGB camera and the 3D depth sensors will be used in this current work. The device is depicted in Figure 3.1.

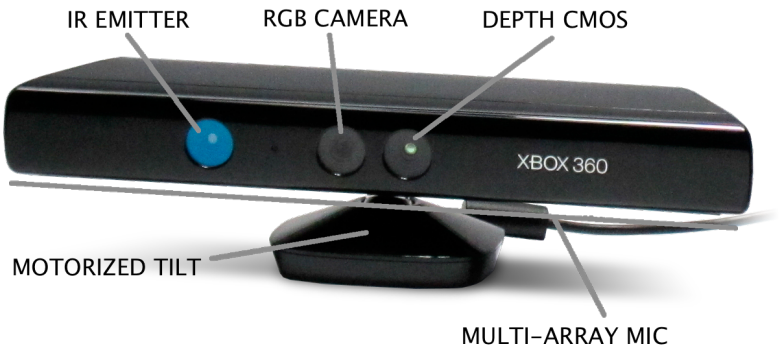


Figure 3.1. Kinect Sensor

3.2.1 Kinect characteristics

The RGB sensor is a regular camera that streams video with 8 bits for every color channel, giving a 24-bit depth. Its color filter array is a Bayer filter mosaic [30].

The depth sensing system is composed by an IR emitter and a standard CMOS Sensor. The IR light source projects structured light, which is then captured by the CMOS image sensor, and decoded to produce the depth image of the scene. It has

an operation range between 0.5 and 6 meters, although best results are obtained in the range that goes from 0.8 to 3.5 meters. Its data output has 12-bit depth.

The depth data retrieved undergoes a *Registration* process, in which the depth pixels are matched with their color equivalents.

The compound system has VGA resolution (640x480 pixels) and a maximal frame rate of 60 Hz, although we will use it with 30 Hz. The field of view is 58^o horizontal, 40^o vertical and 70^o diagonal.

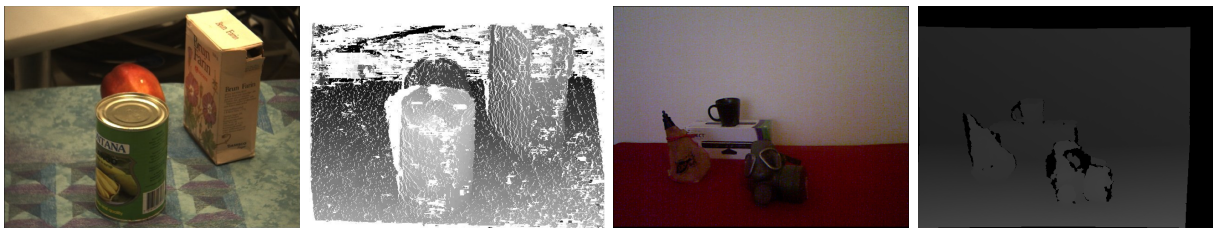
As this device was developed to be used with a gaming console instead of a regular computer, there exists a range of third party open source drivers. For this paper, however, we will use the OpenNI (Open Natural Interface) drivers, developed by PrimeSense alongside with Willow Garage and Side-Kick, and which as of June 2011 are at an alpha stage.

3.2.2 Differences with Björkman and Kragic

The change in the hardware with respect to [16] brings some changes to the overall process.

The experimental platform from Björkman is composed by a 7-joint Armar III robotic head with four Point Grey Dragonfly cameras, two of them being peripheral and the rest foveal. The cameras are distributed so they form two stereo systems. The peripheral view is used to detect the areas of interest in which the foveal system will actually be focused.

A disparity map is then created using Stable Matching [31], method chosen due to its capability to cope with wide disparity ranges and because it focuses on minimizing the number of false positives, instead of on getting the highest possible density.



(A) (B) (C) (D)

Figure 3.2. Disparities. **(A)**: Color image. **(B)**: Stable matching disparity. **(C)**: Color image. **(D)**: Kinect disparity.

In the work presented, using the Kinect the RGB data we obtain a single image with a broad field of view, being more similar to the scene retrieved by the peripheral cameras than to the foveal ones. Due to this, when we focus on a region of

interest inside of the retrieved image, the effective resolution used to perform the segmentation is smaller.

On the other hand, we avoid doing complicated computations in order to calculate a less detailed disparity map from stereo information. Instead from the device we get a high density distance map, which is then quickly translated to a disparity map using the equation (3.8)

$$d = \frac{bf}{z} \tag{3.8}$$

In (3.8), d corresponds to the disparity in pixels. b is the horizontal baseline in meters between the cameras, which in Kinect is 0.075 m. f is the focal length of the IR-camera, 580 pixels. z is the depth value obtained from Kinect.

In Figure 3.2, in **(A)** we can observe a typical scene as observed by a Point Grey Dragonfly foveal camera, and in **(B)** its corresponding disparity map calculated with Stable matching, as shown in [16]. In **(C)** we have another typical scene, this time captured with Kinect, and in **(D)** its disparity map, calculated from the original depth map using (3.8).

This high density disparity map allows us to add new cues based on disparity, which is the main leitmotif behind this thesis. Said additions are described on Chapter 4.

Chapter 4

Cues

The addition of a dense high-quality depth map into the existing framework poses two direct questions. The first one of these questions is 'How can we exploit it?' and the second is 'How effective is that exploitation?'.

This chapter is centered on answering the second question. For that, we are going to study how to integrate disparity-based cues into the existing framework, both for selecting the seed point of the process and for the segmentation per se.

In Chapter 5 and Chapter 6 we will answer the last question, being that the main point of the thesis.

4.1 Symmetry saliency

Symmetry is not fortuitous, hence, symmetrical regions have a higher probability to come from objects than non-symmetrical regions [32]. Although, of course, not every object tends to be symmetric.

We will take the first thought, in combination with the depth map provided by Kinect, as the base to find a suitable seed region for the segmentation to substitute the current method of the framework, which was described in Chapter 3.1.

For this task, we will use the symmetry saliency model presented by Kootstra et al. [33], although with some changes to suit better our task at hand.

Taking the depth data as a starting point, we look for symmetric patterns in pairs of pixels through their intensity gradients in several scales, building a saliency map. Around the global maximum point we select a square patch of points. From this squared patch we take the ones with a depth similar to the one of the global maximum to be used as a seed region. After that, we perform a region growing to find other points that may belong to the same object.

Using that seed region as starting point, we proceed to begin the segmentation process, using the already existing method in the framework by Björkman and

Kragic, together with the cues that will be presented in the following sections of this chapter.

One of the possible problems that first come to mind with this application is that, as the system uses the initial seed as a hard constraint, an erroneous initial calculation of object points will derive into an ill-fated segmentation.

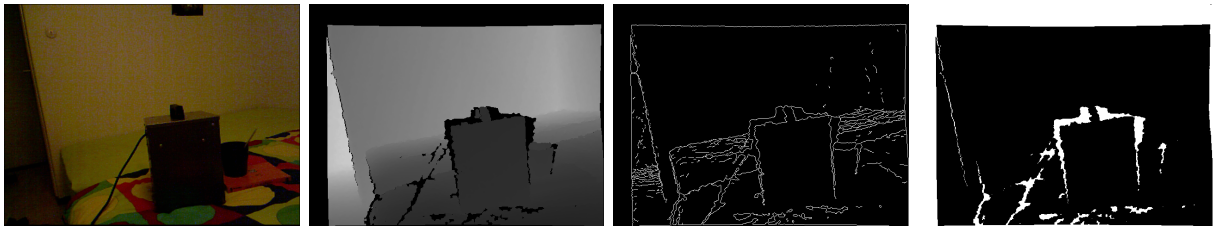
The biggest modification from the original approach by Kootstra et al. is that instead of using color data, we use the depth information retrieved from Kinect. This has the advantage that we will not depend on the colors of the objects, the surface and the background of the image, but only on the spatial distribution of the elements in the scene.

We will maintain the default values provided by the software used, which is part of the Fast and Autonomous Object Detection And Segmentation (ADAS)[1], except for the size of the symmetry kernel, which will be modified by $r_1 = 8$ and $r_2 = 16$.

4.2 Canny edge detector

In Chapter 2.1.1 we discussed the usefulness of an edge detector for data obtained from an RGB camera, as textured elements may cause an edge detector to generate a lot of useless information.

However in this case, as we use the disparity map as input, the edges found will very likely be dependent from the geometric characteristics of the object. Some of those edges will be internal to the object, i.e., looking at a scene with a cube, some of the edges of the cube will separate the object from the rest of the scene, while others will be internal to the object. This information is applied then in the pairwise links of the Markov Random Field explained in Chapter 2.1.2.



(A) (B) (C) (D)

Figure 4.1. Cues. **(A)**: Color image. **(B)**: Depth image. **(C)**: Canny cue. **(D)**: Halo cue.

Despite the fact that our distance data has a depth of 12 bits, we truncate that data, removing the 4 lower bits and keeping the rest. This makes us lose resolution, but as we are looking for really strong edges, it does not affect much our purposes.

[1] http://www.csc.kth.se/~kootstra/index.php?item=703&menu=700

4.2.1 Direct application

A point not marked as an edge, will with high probability have the same label as its neighboring non-edge points. A point that is marked as an edge may represent an internal edge, so it will keep the same label as all its neighbors, or may represent an external edge, thus marking the transition from one label to another.

In our case, we will calculate the Canny edge using a hysteresis procedure. The first threshold will be set to 20 and the second to 30. The kernel size of the Sobel operator will have a size of 3.

After calculating the edges with the Canny filter, we will obtain an image in which pixels that represent an edge have a value of 255, and the rest will be set to 0. An example of a resulting Canny segmentation can be observed in Figure 4.1 **(C)**.

We define the energy function of the pairwise links between the pixels as follows:

$$V_{i,j}^{C} = \lambda^{C} \exp^{-(v_i^C v_j^C)} \tag{4.1}$$

This leaves us the following values for the links:

- Neither i and j are edges: $V_{i,j}^{C} = \lambda^{C} \exp^{0} = \lambda^{C}$.
- Either i or j are edges: $V_{i,j}^{C} = \lambda^{C} \exp^{0} = \lambda^{C}$.
- Both i and j are edges: $V_{i,j}^{C} = \lambda^{H} \exp^{-65025} \approx \lambda^{C}\, 10^{-28240} \approx 0$.

4.2.2 Gaussian blur

As the Canny edge detection produces all or nothing results, that is, all the values will be either 0 or 255, we will also experiment by using the cue after smoothing it with a Gaussian filter.

The aim of this transformation is to check if the smoothing of the cue instead of using such absolutist values, by giving more relevance to the neighboring pixels of the actual edges, can help with the segmentation process.

For this filter, we will use a kernel size of 5x5 and a standard deviation σ of 0.3 both for the x and the y coordinates.

The energy function of the pairwise will remain the same as when we apply this cue directly, so it will be the one defined in (4.1).

4.2.3 Distance transform

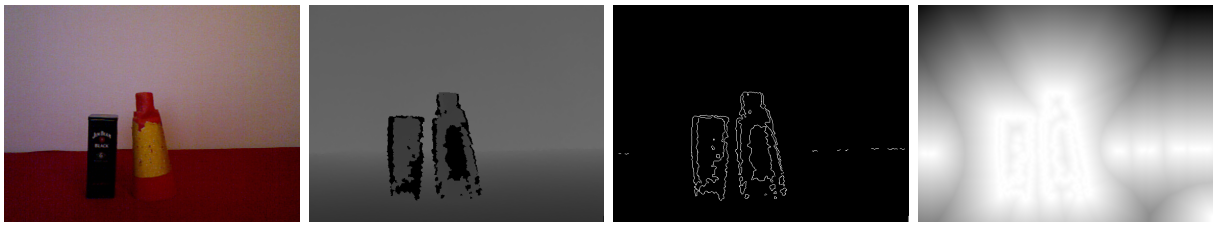
Another way to experiment with giving more relevance to the neighboring pixels of the actual edges is to use the distance transform. This transformation consists on creating a map of the image in which the value of each pixel is the distance of said pixel to the closest pixel with 0 value in the original image [34]. In our case, given the nature of the values in the Canny image, we will have to make several adaptations.

First of all, as in the method by Borgefors the distance transform is calculated using as origin the pixels with value 0, we will switch 255 for 0 and 0 for 255 in every pixel for the original Canny cue. Then we will calculate the distance transform as explained by the method.

Once we have the distance transform, to be able to use it for our purposes, we will have to 'invert' again the values for every pixel. We will make said correction for every pixel using the equation (4.2), where $v_{\max}$ is the value of the maximum distance in the image.

$$v = 255 - \frac{255\, v}{v_{\max}} \tag{4.2}$$

This correction will return normalized values between 0 and 255, with 255 being the value of the detected edges. The resulting image can be observed in Figure 4.2 **(D)**.



(A) (B) (C) (D)

Figure 4.2. JimConeFar. **(A)**: Color image. **(B)**: Depth image.**(C)**: Canny cue. **(D)**: Corrected Canny distance transform.

To be able to properly use this cue, we substitute the energy cost of the pairwise links from (4.1), to the ones presented in (4.3).

$$V_{i,j}^{C} = \lambda^{C} \exp^{-\frac{v_i^C v_j^C}{255^2}} \tag{4.3}$$

4.3 Halo effect

While retrieving depth information from Kinect, there may be areas in which we lack depth data. This happens due to two different events:

The first event is the presence of reflective surfaces that deviate some of the IR points to areas that are out of the field of view of the CMOS sensor. Those lost points will be represented in the disparity data after its calculation with $d = -1$.

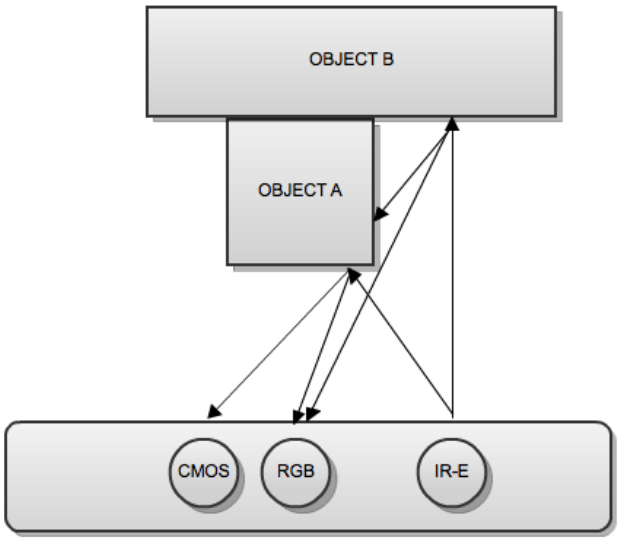


Figure 4.3. Halo effect

The second occurrence is derived from the specific hardware characteristics from Kinect. The IR light source, the CMOS sensor for capturing the depth and the RGB camera are all placed at a horizontal axis in the device. This distribution implies that the projector and the two cameras all have different points of view. This, in conjunction with the *Registration* process performed while retrieving the data, provokes the existence of dark areas produced by objects in the scene occluding some of the IR points.

As an example we can see Figure 4.3. In it, we consider two IR points emitted from Kinect, one being projected in Object A and the other in Object B. Due to the perspective of the two sensors, they capture different information. While the RGB camera is able to see the surfaces in which both points are projected, the CMOS sensor only retrieves the depth information from the point in Object A, while this same object occludes the point projected onto Object B. This is what we have come to define as *Halo effect*.

The two phenomena described here can be observed in Figure 4.1 **(D)**. The sides of the box contain no information due to the Halo effect, and the top of it because

of the reflective surface. Out of this phenomena, the relevant for us is the Halo effect, as that lack of information actually indicates the possible location of objects, or more specifically their boundaries, in the scene with a minimum information processing.

4.3.1 Direct application

Similarly to the case presented in Section 4.2, a raw processing of this data tells us that if a point has depth information, with a high probability it belongs to the same label as its neighboring points with depth information. However, if a point lacks depth information, probably it is in the boundaries of an object, and it could belong with the same probability to the foreground, to the surface of to the background. The difference is that here the presence of internal edges is almost non-existent.

While calculating the Halo image, the points with lack of depth information are marked with 255, and the points with any other information, with the value 0. We define the cost function of the pairwise terms between the pixels as:

$$V_{i,j}^{H} = \lambda^{H} \exp^{-(v_i^H v_j^H)} \tag{4.4}$$

Similarly to the Canny cue, we get the following values for the links:

- Both i and j have depth information: $V_{i,j}^{H} = \lambda^{H} \exp^{0} = \lambda^{H}$.
- Either i or j have depth information: $V_{i,j}^{H} = \lambda^{H} \exp^{0} = \lambda^{H}$.
- Neither i and j have depth information: $V_{i,j}^{H} = \lambda^{H} \exp^{-65025} \approx \lambda^{H}\, 10^{-28240} \approx 0$.

4.3.2 Gaussian blur

As in the case for the Canny cue, with the Halo result all the information is either all or nothing, in some tests we will proceed to smooth it with a Gaussian filter.

The energy function for this case will remain the same as with the direct application of the cue, so we will use the one described in (4.4).

4.3.3 Distance transform

The distance transform operation performed in the Halo cue is exactly the same as we defined it for the Canny cue in Chapter 4.2.3. The resulting image can be observed in Figure 4.4 **(D)**.

Similarly, we have changed the equation for the energy cost of the pairwise links from (4.4) to the one seen in (4.5).

(D) (E) (F) (G)

Figure 4.4. JimConeFar. **(A)**: Color image. **(B)**: Depth image.**(C)**: Halo cue. **(D)**: Corrected Halo distance transform.

$$V_{i,j}^{H} = \lambda^{H} \exp^{-\frac{v_i^H v_j^H}{255^2}} \tag{4.5}$$

4.4 Canny and Halo integration into the framework

While the purpose of the Symmetry saliency is to substitute the attention mechanism from the original framework, both the Canny and Halo cues are designed to be able to work simultaneously with the existing Luminance cue, which is based on the difference in the luminance between points, as described in Chapter 3.1.

If we rewrite the penalty due to the luminance (3.7) as:

$$\begin{aligned} V_{i,j}^{L} &= \lambda^{L} \exp^{-\beta^{L}(v_i^L - v_j^L)^2} \\ \beta^{L} &= (2\langle (v_i^L - v_j^L)^2 \rangle)^{-1} \end{aligned} \tag{4.6}$$

then the newly formed $V_{i,j}$ for the cues if used directly or after the Gaussian blur filter is defined by the equations (4.6), (4.1) and (4.4) as shown in (4.7).

$$\begin{aligned} V_{i,j} &= V_{i,j}^{L} + V_{i,j}^{C} + V_{i,j}^{L} = \\ &= \lambda^{L} \exp^{-\beta^{L}(v_i^L - v_j^L)^2} + \lambda^{C} \exp^{-(v_i^C v_j^C)} + \lambda^{H} \exp^{-(v_i^H v_j^H)} \\ \beta^{L} &= (2\langle (v_i^L - v_j^L)^2 \rangle)^{-1} \end{aligned} \tag{4.7}$$

On the other hand, the resulting $V_{i,j}$ if we use the cues after applying the distance transform is described by the equations (4.6), (4.3) and (4.5) to form the equation (4.8)

$$\begin{aligned} V_{i,j} &= V_{i,j}^{L} + V_{i,j}^{C} + V_{i,j}^{L} = \\ &= \lambda^{L} \exp^{-\beta^{L}(v_i^L - v_j^L)^2} + \lambda^{C} \exp^{-\frac{v_i^C v_j^C}{255^2}} + \lambda^{H} \exp^{-\frac{v_i^H v_j^H}{255^2}} \\ \beta^{L} &= (2\langle (v_i^L - v_j^L)^2 \rangle)^{-1} \end{aligned} \tag{4.8}$$

This will be used in the Potts model (3.6) that models the joint probabilities of two neighboring points.

Chapter 5

Experiments

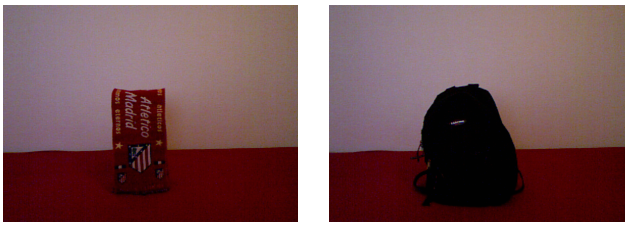
In this chapter, we will proceed to talk about the experiments that were performed. First we will observe the results of the symmetry saliency, the method which purpose is to substitute the original seed region selection of the framework. Then we will talk about energy maps, a graphical representation used to calibrate the λ in each scene.

After that, we will quickly observe the segmentation process over time. Then how the cues individually affect the segmentation. An insight of the cues' performance in several scenes. Finally, we will end with the effects of the Gaussian blur and the distance transform.

5.1 Symmetry saliency

In the beginning, we used the depth map as input for the symmetry saliency almost straight out of the Kinect. However, we noticed that for some specific cases, the seed point was chosen in areas in which apparently there were no points of interest.

We observed that this was happening in scenes in which the depth information of the object was not very symmetric. For example, it was happening in the image Backpack, but not in Scarf. Both images can be seen in Figure 5.1



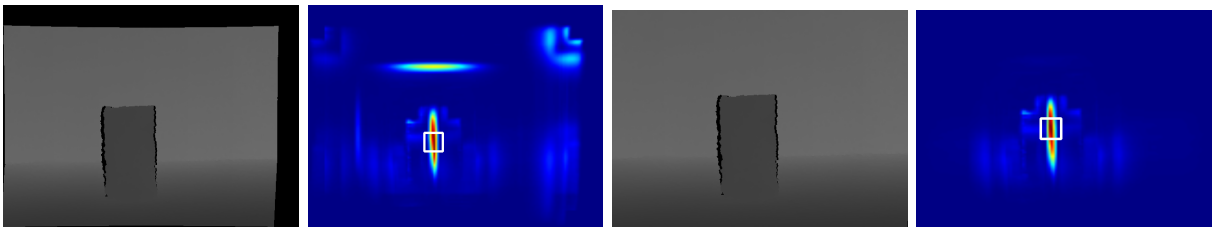
(A) (B)

Figure 5.1. Color images: **(A)**: Scarf. **(B)**: Backpack.

After the *Registration* process performed in the Kinect, we obtain a depth map in which the depth information is in the same coordinates as its corresponding color information. As both the fields of view and the focal length are different, there are two margins that lack depth information on the top and the right of the depth map.

When feeding the depth map directly into the symmetry saliency mechanism, these margins produce false positives in the detection of symmetries. And depending on the scene, these false positives can be stronger than the ones generated by the proper information.

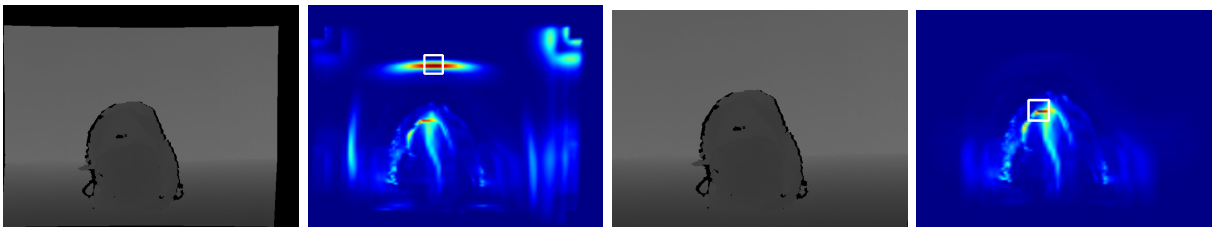
As we stated previously, in Scarf (Figure 5.2) this was not the case. Looking into the symmetry saliency map, we can see that there are several erroneous peaks. Although they are not strong enough to occlude the one provided by the real data. So, in this case we would use the correct seed point anyway.



(A) (B) (C) (D)

Figure 5.2. Scarf symmetry saliency. **(A)** and **(B)**: Original depth map and its symmetry saliency. **(C)** and **(D)**: Cropped depth map and its symmetry saliency.

However, in Backpack (Figure 5.3), given the nature of the object and its depth map, the peak that it produces is not stronger than the incorrect one. The algorithm chooses the wrong point as a seed point and the whole segmentation fails from the very beginning.



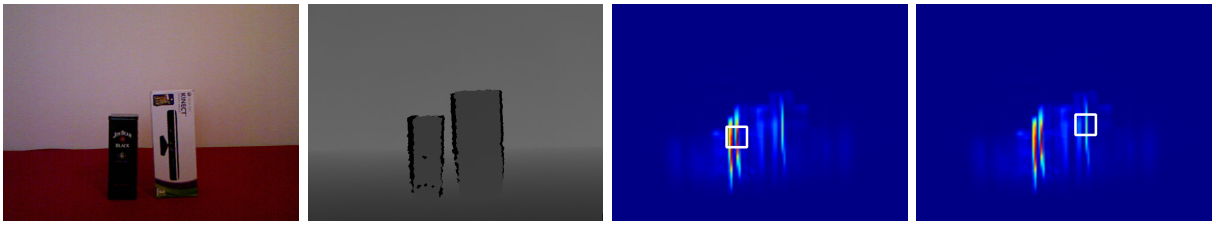
(A) (B) (C) (D)

Figure 5.3. Backpack symmetry saliency. **(A)** and **(B)**: Original depth map and its symmetry saliency. **(C)** and **(D)**: Cropped depth map and its symmetry saliency.

To fix this issue, we process the image by cropping the margins of the depth maps, and also the equivalent points of the rest of the images (color and depth), leaving us with an effective resolution of 580x425 pixels. From this point onwards, all the images shown in this work have been cropped according to this.

As it can be seen in the right half of Figure 5.2 and Figure 5.3, cropping the image fixed the problem, eliminating the errors in the symmetry saliency map and allowing for finding a precise seed point.

We have previously shown that the symmetry saliency works when we have one object, and well defined background and surfaces. But it also works when we have more objects in different arrangements, as shown in Figure 5.4 with two objects close to each other.



(A) (B) (C) (D)

Figure 5.4. JimKinectFar symmetry saliency. **(A)**: Color image. **(B)**: Depth map. **(C)** and **(D)**: Symmetry saliency for the first two peaks.

In this example we see that although the biggest of the two local maxima are in the object on the left, as second object we choose the one in the right. This is performed by ignoring local maxima that have already been labeled as foreground when we look for seed points.

So, for the redundant points to be ignored properly, we have to wait until we have finished one object before adding the next one. If not, we run the risk of setting more than one seed point inside of the same object.

As for the performance of the method with several objects close to each other let's see the examples posed in Figure 5.5 and Figure 5.6.

In Figure 5.5 we can see that the first seed point is within the wineskin, while the second one is in the gas mask. However, while choosing the third one, a point between the top of the wineskin and the mug is chosen, thus, using as a seed point an erroneous one.

In Figure 5.6, the first seed point is within the wineskin, the second one is in the middle of the Kinect box and the third one in the wineskin. In this case, for the fourth point, again we get a wall point that is between the top of the wineskin and the mug.

Also, in **(C)** – **(E)** we see that despite having a big vertical red bar, a point outside of it is chosen. This is because once the saliency map is computed, we look for a single strong point, not taking into account its surroundings. So despite the fact that that vertical area seems a nice starting point, the actual point with most relevance is outside it.

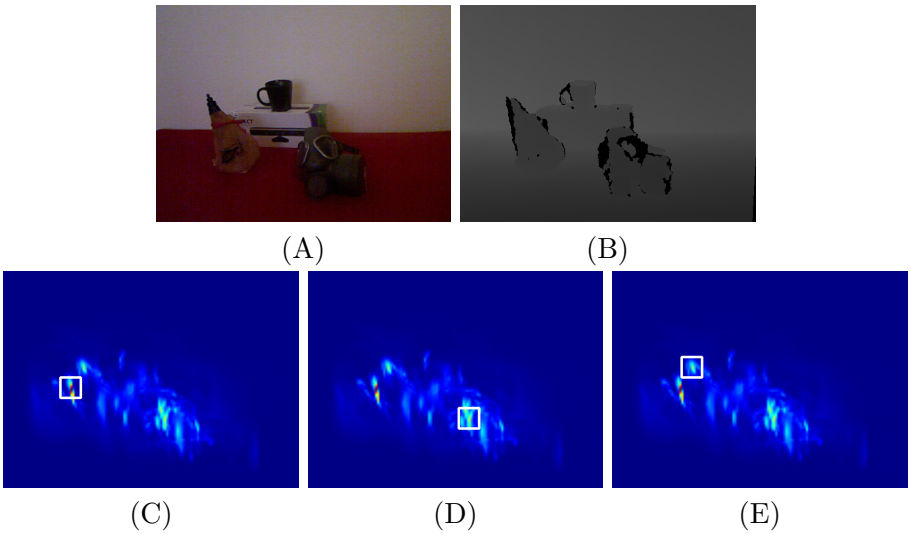


Figure 5.5. Cluttered symmetry saliency. **(A)**: Color image. **(B)**: Depth map. **(C)** – **(E)**: Symmetry saliency for the first three peaks.

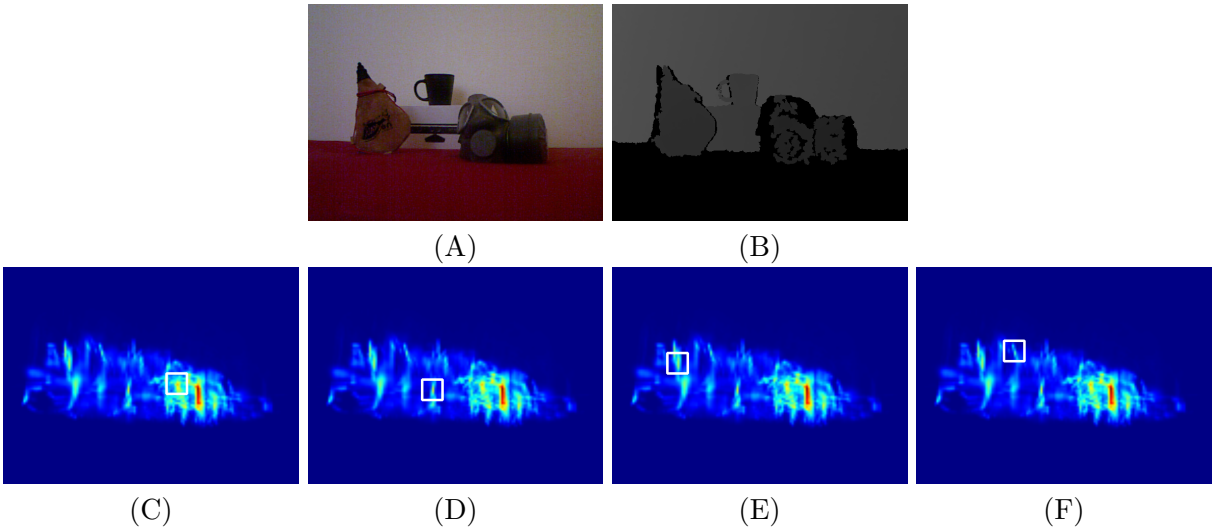


Figure 5.6. Clutteredns symmetry saliency. **(A)**: Color image. **(B)**: Depth map. **(C)** – **(F)**: Symmetry saliency for the first four peaks.

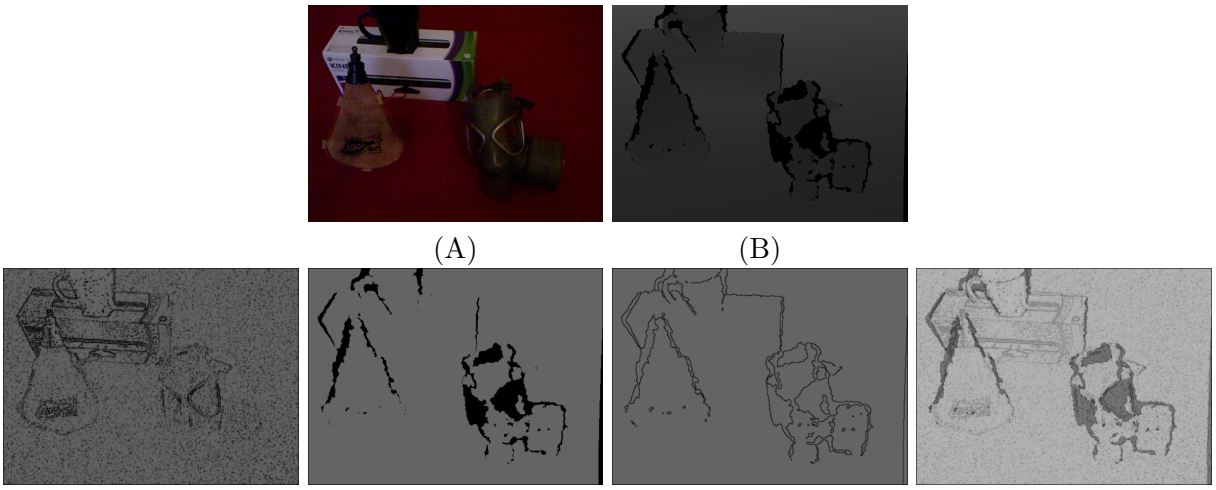
So, if we have clear view of the objects and regular surfaces and backgrounds, we detect the objects without issues, while we may face problems if the objects are occluded or the scene is cluttered.

5.2 Energy map

Once finished with the selection of the seed point, it is time to pass to the proper segmentation process.

As for the tests, in some cases we combined three different cues, to be able to better understand the results of the segmentations due to the effect of said cues, we create an energy map of the scene.

To obtain said energy map, for every pixel we sum the value of each of the four pair-wise terms ($V_{i,j}$ as described by Equation (4.7) in Chapter 4.4), which, in turn is the the sum of the effect of the three cues for that particular pair of pixels.



(A) (B)

(C) $50 - 0 - 0$ (D) $0 - 50 - 0$ (E) $0 - 0 - 50$ (F) $30 - 30 - 30$

Figure 5.7. Clutterednb energy map. **(A)**: Color image. **(B)**: Depth map. **(C)** – **(F)**: Energy maps for different values of $\lambda^L - \lambda^H - \lambda^C$.

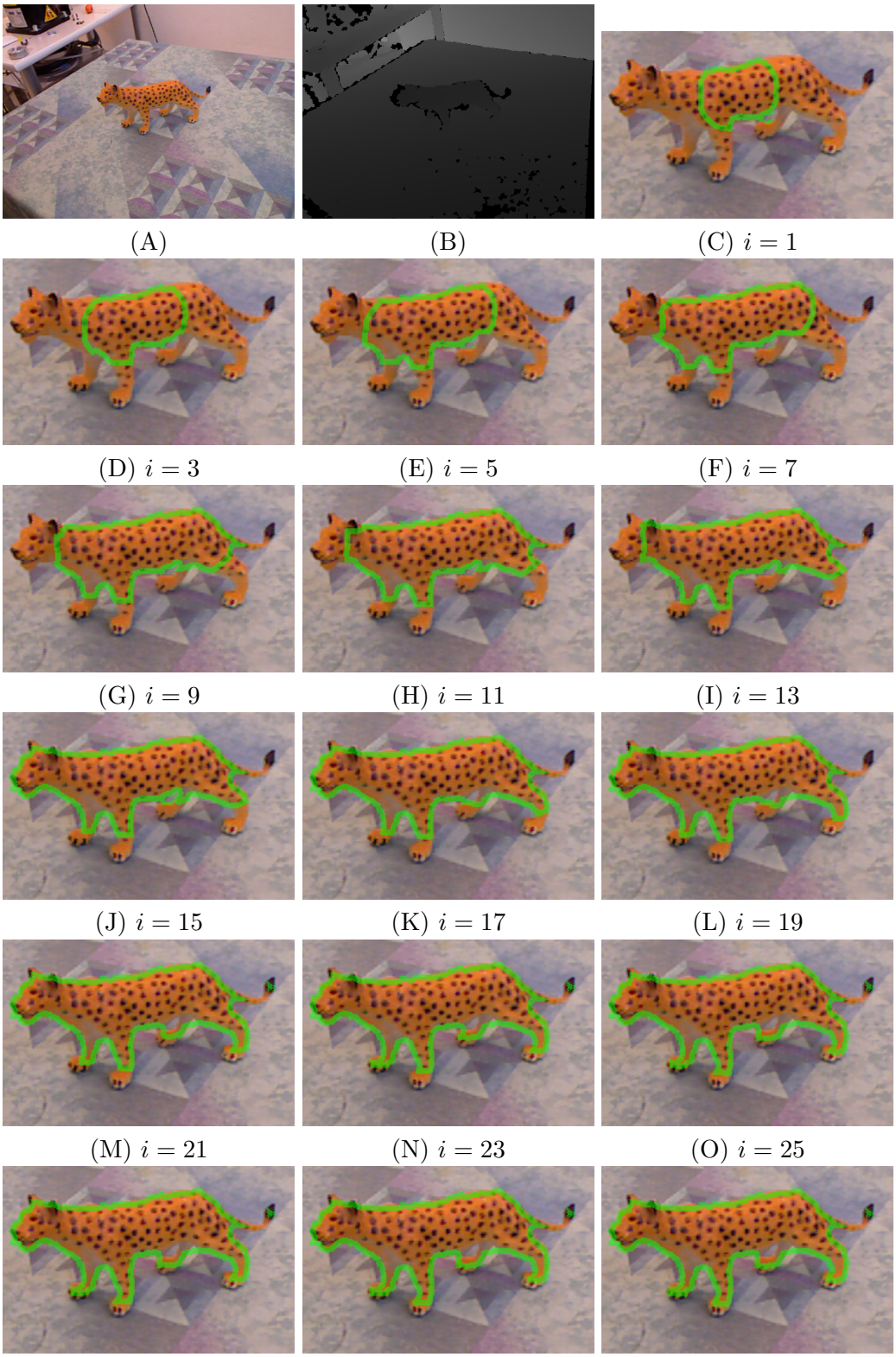
In Figure 5.7 we have an example of such energy maps for the image Clutterednb. They are shown here for illustration, but they will not be necessary for understanding the rest of the examples shown in this chapter.

Despite the fact that the energy maps were calculated and used to understand the results of each segmentation, they will not be shown further in this report.

5.3 Iterative segmentation

As explained in Chapter 2.1, the process in which this thesis is based is an iterative segmentation.

While in the rest of this chapter we will show only final results for each of the experiments, in this section we show how the result evolves, step by step, from the selection of the seed point till the optimal final result.



(A) (B) (C) $i = 1$

(D) $i = 3$ (E) $i = 5$ (F) $i = 7$

(G) $i = 9$ (H) $i = 11$ (I) $i = 13$

(J) $i = 15$ (K) $i = 17$ (L) $i = 19$

(M) $i = 21$ (N) $i = 23$ (O) $i = 25$

(P) $i = 27$ (Q) $i = 29$ (R) $i = 31$

Figure 5.8. Tiger segmentations timelapse. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(R)**: Segmentation results for different iterations i.

In Figure 5.8 we can observe the original color **(A)** and depth **(B)** images for Tiger. Then, from **(C)** to **(R)** we have the segmentation image obtained every 2 iterations. For better appreciating the evolution of the segmentation, we focus only on the most relevant area of the image.

In this specific case, the segmentation has been performed with $\lambda^L = 0$, $\lambda^H = 50$ and $\lambda^C = 0$.

The evolution of the segmentation is better seen in a video instead of still images. Some segmentation examples can be seen in
http://www.youtube.com/watch?v=NrdCriH6xIA.

5.4 Individual effects of the cues

In this section, we show segmentations obtained by using the cues proposed in Chapter 4.2 and Chapter 4.3. We take a scene and observe how different values of λ affects the final result of the segmentation. We will present the behavior of each cue independently from the others.

Due to all the possible combinations of $\lambda^L - \lambda^H - \lambda^C$ values, it is hard to present a study of how the cues interact with each other depending on the λ values. To this end, several segmentation scenes with specific values of $\lambda^L - \lambda^H - \lambda^C$ both independently and combined will be shown in Chapter 5.5.

In Chapter 5.5.7 we will view a few relevant results for this same scene. Now we present a collection of segmentations with values of 10, 30, 50, 70 and 90 for each of the λ.

To observe the outputs clearer, we only show an area of interest, and the color levels of the images have been modified. The results can be found in Figure 5.9.

First we will focus our attention to how the framework performs while not using any of the three cues, that is, while removing the pairwise links from the process. In **(C)** we can see that the segmentation is very noisy, and expands itself selecting areas that are not even close to the objects. This can be seen very clearly specially on the surface right under the object.

This effect happens as, when we remove those pairwise terms, we only use the hypotheses of the planar surface and disparity measurements, weakening the whole segmentation process.

For the Luminance cue (left column), we see that the segmentation for the two objects has completely opposite fates. While the segmentation of the wineskin improves when we increase λ^L, the result obtained for the gas mask gets progressively worse.

For the Halo (middle column) cue, the effect is similar to the one from Luminance. The difference is that while the degradation in the segmentation of the mask

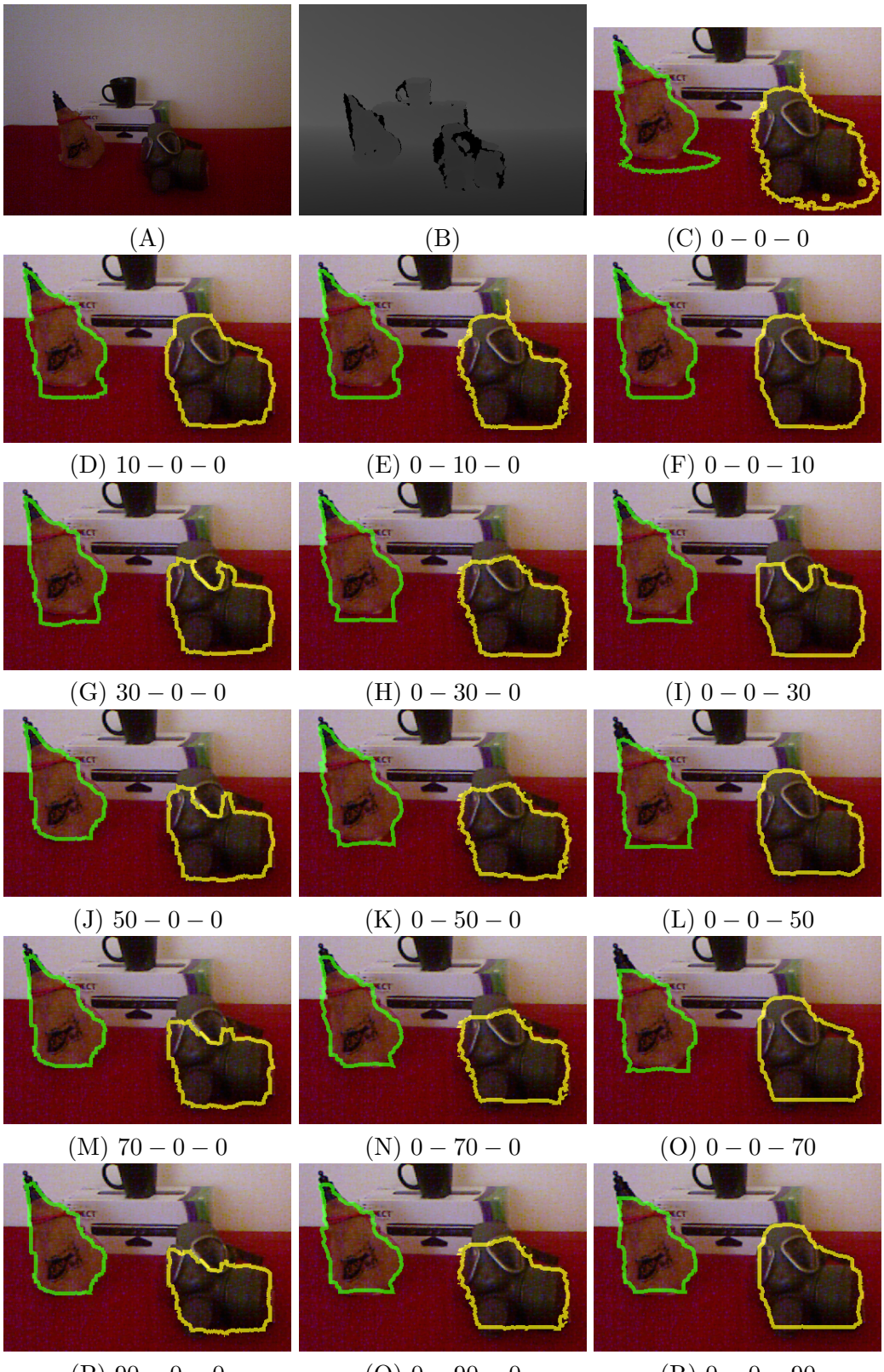


Figure 5.9. Cluttered segmentations range. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(R)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

for Luminance seems to continue as we increase λ^L, in the other case it promptly stops, specifically at $\lambda^H = 30$.

In the Canny (right column) cue, for most of the cases the segmentation of the gas mask is almost perfect, except for $\lambda^C = 30$, where we reach a local best result quite different from the rest of the values. The evolution of the segmentation for the wineskin is similar to the one offered by the other two cues, although getting a faster degradation.

5.5 Segmentation scenes

In this section we will present results obtained by using the cues proposed in Chapter 4.2 and Chapter 4.3, sometimes alone and sometimes in combination between them and also with the original luminance gradient cue from the framework.

All the scenes have different strong points to challenge the segmentation method and its cues in different ways.

In all the tests, when the Luminance cue is used standalone, λ^L is always set to $\lambda^L = 50$, as that is the default value provided by [16].

Where the results of the segmentation could not be clearly appreciated in the original scene, we have zoomed and modified the color levels, to provide clearer results.

5.5.1 Scarf

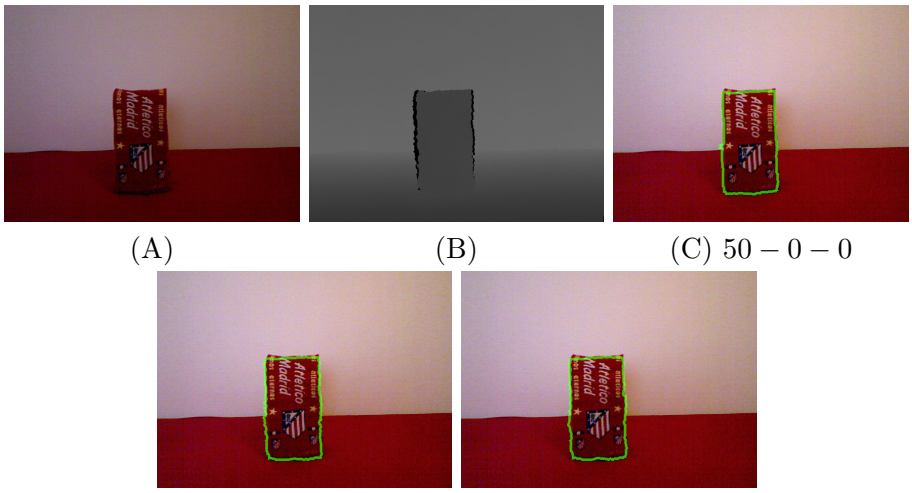
In this subsection we study the result of segmenting a single object with good color and depth information in a scene with clearly differentiated surface and background. Given the similarity between the colors of the surface and the object, this task should be hard for the Luminance cue.

As it can be observed in Figure 5.10, the three cues are able to independently perform a successful segmentation of the object, giving almost the same result each of them.

It is worth nothing that both for Halo **(D)** and Canny **(E)**, values of $\lambda^H = 100$ and $\lambda^C = 100$ were respectively used to achieve this level of accuracy. As we will see in further examples, usually such a high value is not required.

5.5.2 JimConeFar

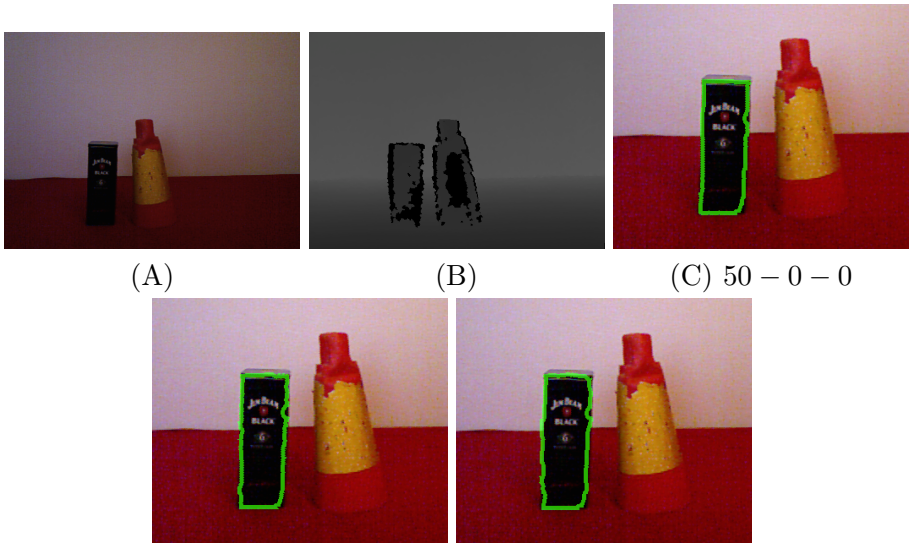
Here we present a scene with two objects that are distant between them. The color information is good, but there are several reflections in the surface of the objects. The surface and the background are clearly differentiated.



(A) (B) (C) $50-0-0$

(D) $0-100-0$ (E) $0-0-100$

Figure 5.10. Scarf segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(E)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.



(A) (B) (C) $50-0-0$

(D) $0-30-0$ (E) $0-0-30$

Figure 5.11. JimConeFar segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(E)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

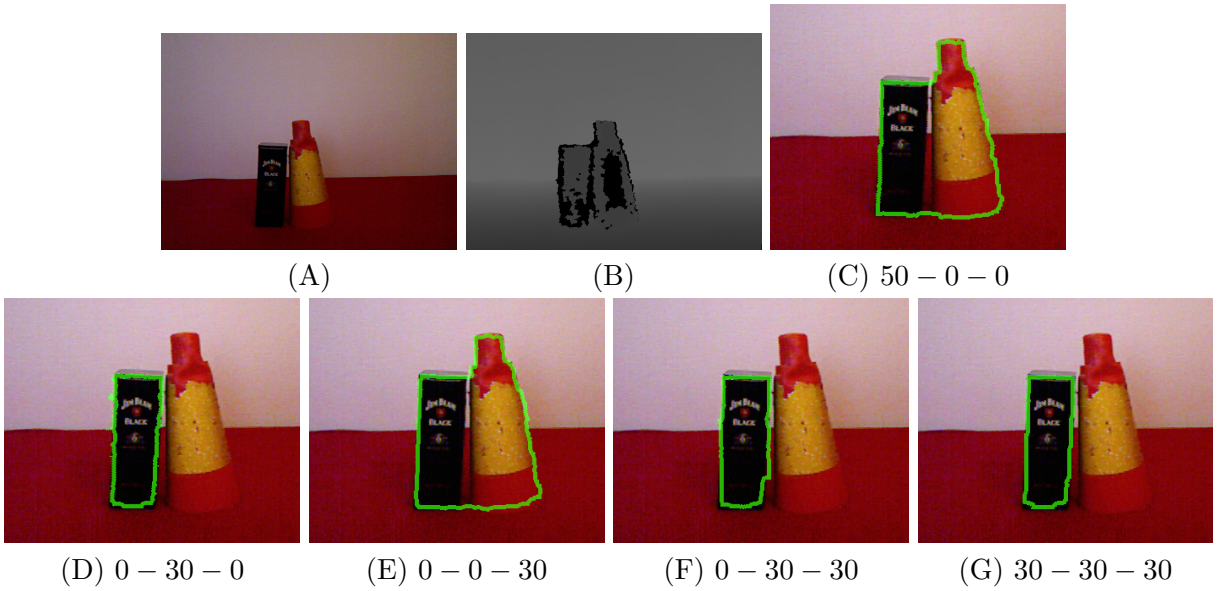
In Figure 5.11, in the Luminance cue **(C)** is able to almost perfectly capture the shape of the object, as well as the Halo cue **(D)**. In this case the Canny cue **(E)** presents a slightly worse performance that the rest.

For Halo **(D)** and Canny **(E)** we use values of $\lambda^H = 30$ and $\lambda^C = 30$ respectively. This values are much smaller than the ones used in the previous scene, and generally

during the experiments presented here, usually proved to provide the best results for this cues.

5.5.3 JimConeClose

Now we will use a scene similar to the one in Chapter 5.5.2, but placing the objects so close to each other that they are almost touching.



(A) (B) (C) $50 - 0 - 0$

(D) $0 - 30 - 0$ (E) $0 - 0 - 30$ (F) $0 - 30 - 30$ (G) $30 - 30 - 30$

Figure 5.12. JimConeClose segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

In Figure 5.12 we can appreciate some differences with respect to the previous scene. While Luminance **(C)** and Canny **(E)** cues fail to capture one object and end up labeling the two objects as the same, Halo **(D)** is able to differentiate one from the other, with good accuracy.

Combining Halo and Canny **(F)** we are able to capture only the desired object, and actually the performance in the top, bottom and left part is better than Halo **(D)**. If we combine the three cues **(E)**, we obtain the best result for this scene.

5.5.4 Leopard

In this subsection, we show an image similar to the ones used in [16]. Specifically, the same tablecloth and one of the same objects is used.

In Figure 5.13 we can observe that by just using the Luminance cue **(C)** we get a result in which the area between the legs of the toy is incorrectly mistaken as figure. The tail is also not correctly identified, but as we will see, that is common

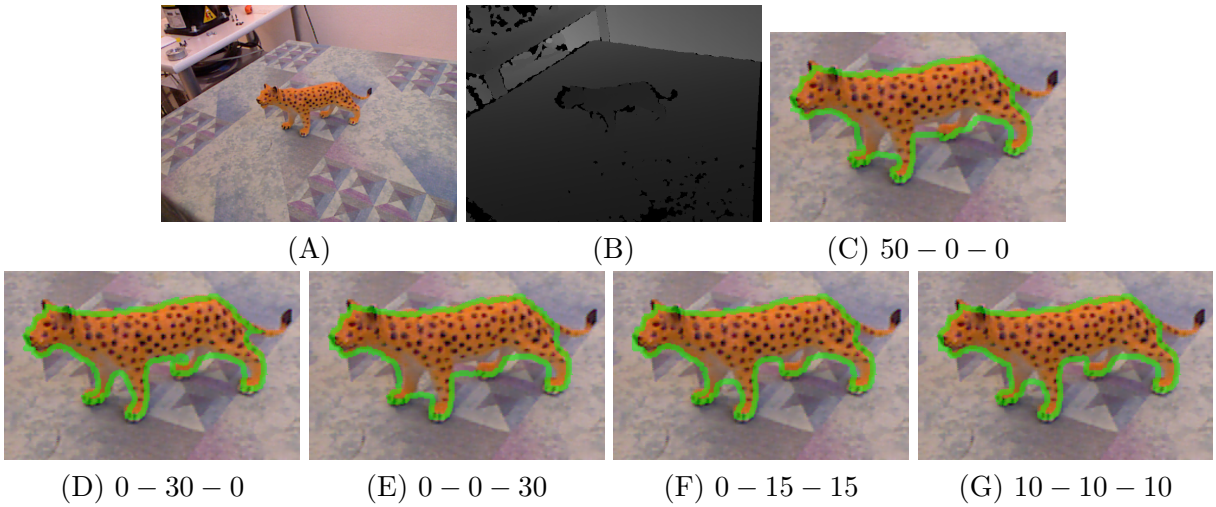


Figure 5.13. Tiger segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** − **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

to every segmentation attempt we performed. This happens due to the second term in (3.5), which is a smoothing term that penalizes discontinuities.

Canny **(E)** provides a solution of similar characteristics to the one performed by Luminance **(C)**.

Out of the three cues, the best result is gotten with Halo **(D)**. It gives the best result. Both the head and the space between the legs are adequately segmented.

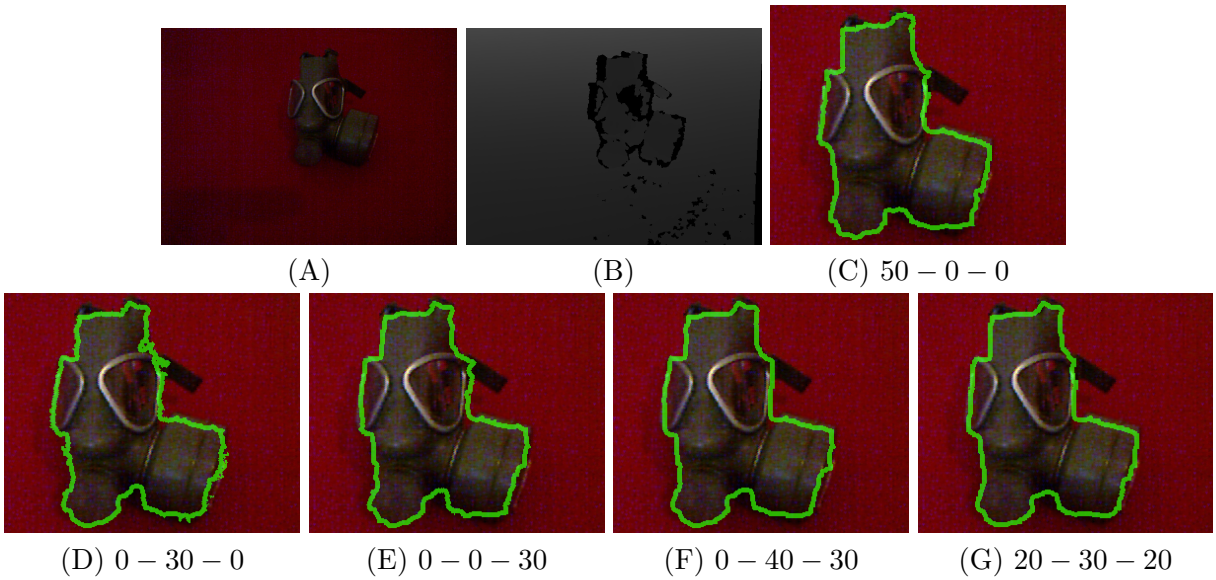
While combining the cues, as in **(F)** and **(G)**, we see that although the result is better than Luminance **(C)** and Canny **(E)** standalone, it is not as good as using Halo **(D)** only.

5.5.5 GasmaskNB

This scene presents an asymmetrical object with shiny surface and transparencies. It is placed in an environment in which we only have one plane, in contrast to the previous scenes in which we had both a surface and a wall. Further, there is a presence of heavy shadows.

The original images in question are in Figure 5.14. The Luminance cue **(C)** does a good job finding the boundaries of the gas mask, although it confuses part of the shadow between the two lower modules with the object, and fails to completely retrieve the left glass and the rightmost part of the filter.

The Halo cue **(D)** does a similar job to **(C)**. However, while it is not confused by the shadow and captures slightly better the left lens and the filter, the result is noisier.



(A) (B) (C) $50 - 0 - 0$

(D) $0 - 30 - 0$ (E) $0 - 0 - 30$ (F) $0 - 40 - 30$ (G) $20 - 30 - 20$

Figure 5.14. GasmaskNB segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(G)**: Relevant segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

The result of the Canny **(E)** cue is the best of the three, slightly improving the performance of Luminance **(C)** by retrieving better the shape of the object with a steadier segmentation. In **(F)** the combination of Halo and Canny produces a result that slightly outperforms Canny **(E)**.

Lastly, the combination of the three cues **(G)** gives the best result, capturing better the top right of the mask, as well as the left lens, while not falling so much for the shadow between the modules. The filter is not so well retrieved. But we can also observe that the overall segmentation is less noisier than any of the previous results, with more solid boundaries.

5.5.6 GasmaskNS

Here we have again the same asymmetrical object as in Chapter 5.5.5, but from a different perspective. If we pay attention to the depth map **(B)** in Figure 5.15, we see that we do not have depth data for the surface. Also, the depth data from the sides of the object has some big missing segments.

Of course, the Luminance cue **(C)** is unaffected by this situation, providing again a very good segmentation.

By studying the depth map, we can guess that the cues depending on it will not be performing well. And the results from the Halo cue **(D)** confirms that. Although it more or less captures the object, the segmentation provided is not very accurate and presents a lot of noise.

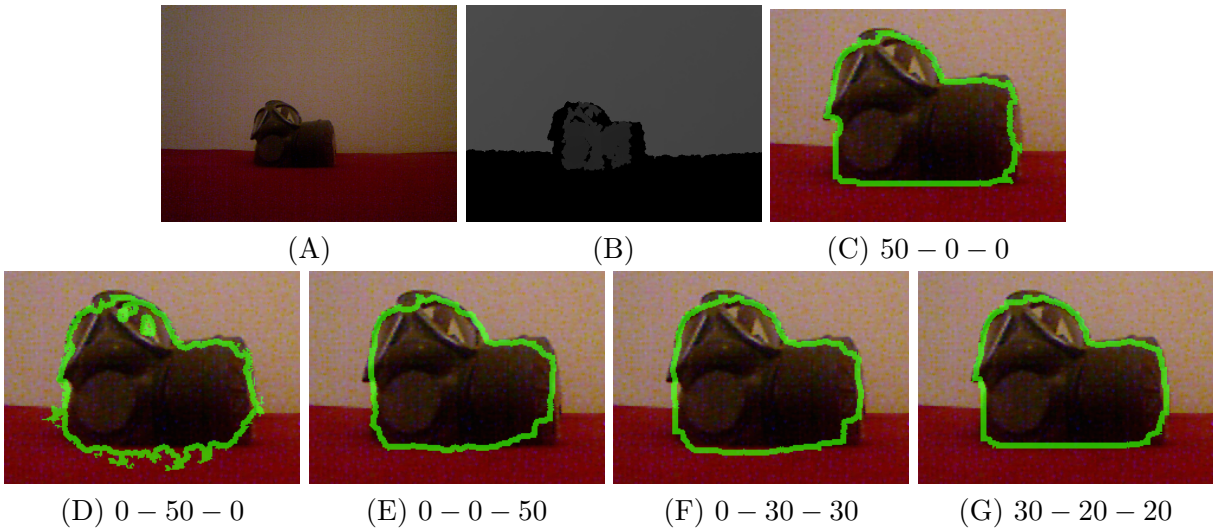


Figure 5.15. GasmaskNS segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** − **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

For Canny **(D)**, however, the bad quality of the depth map does not affect the segmentation as much as it did with Halo **(D)**, although it does not reach the performance level of Luminance **(C)**

The combination of both the Halo and Canny cues **(F)**, although improving the independent results, is still not on par with the performance of Luminance standalone **(C)**.

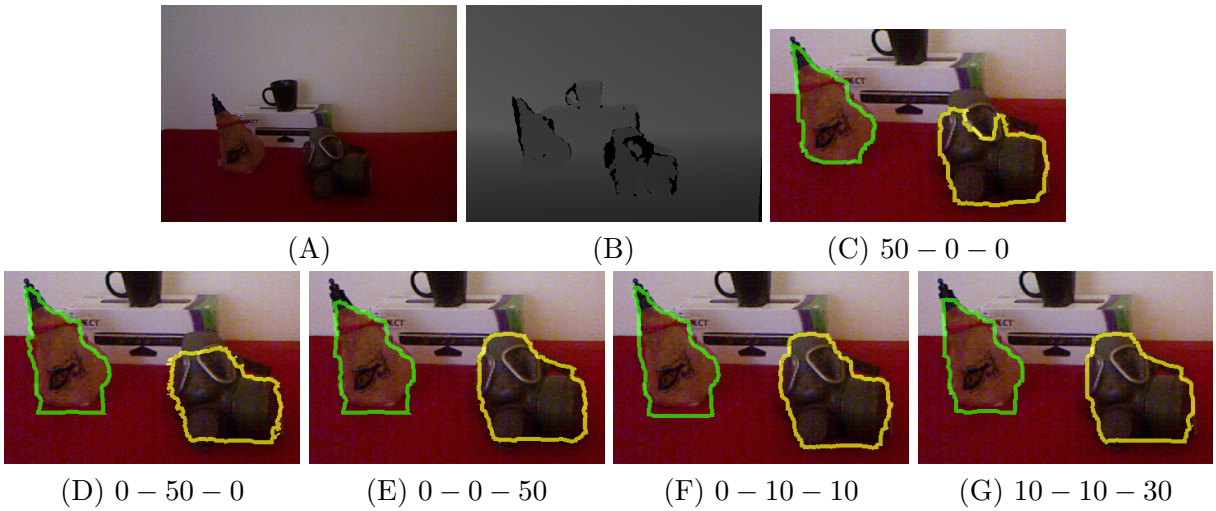
However, using the three cues together **(G)** we obtain the best result overall. Comparing it with **(C)** we see that although the arch of the left side is not as precise, the segmentation is better both on the top and the right of the mask, covering the object more accurately.

5.5.7 Cluttered

Now we turn our attention to segmentation of more than one object. In this scene we have four objects, although we will only try to segment the wineskin and the gas mask, as both attention methods used fail to find the Kinect box and the mug, due to the distribution on objects in the scene, as we commented in Chapter 5.1.

The scene shown in Figure 5.16 has clearly defined surface and background, and as in the two previous scenes, one of the objects is very asymmetrical and includes reflections and transparencies.

The result provided by the Luminance cue **(C)** is a little bit mixed. While it captures the wineskin almost perfectly, it fails to correctly segment the upper part of the gas mask.



(A) (B) (C) $50 - 0 - 0$

(D) $0 - 50 - 0$ (E) $0 - 0 - 50$ (F) $0 - 10 - 10$ (G) $10 - 10 - 30$

Figure 5.16. Cluttered segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

The Halo cue **(D)** fails to properly find the boundary between the wineskin and the surface, while the rest is properly segmented. For the gas mask, the result is quite similar to the one found by the Luminance **(C)**, although it also manages to cover the right lens of the mask.

The Canny cue **(E)** provides similar a worse result than the other two cues for the wineskin. However, for the gas mask it provides a perfect segmentation.

Combining the depth based cues **(F)**, we get almost the best of both worlds, improving the gas mask segmentation with relation to Halo **(D)** and the performance for the wineskin compared to Canny **(E)**.

Finally, the result that merges the three cues **(G)**, for the gas mask is again perfect, and for the wineskin just slightly worse than in Luminance **(C)**.

More segmentations for this scene can be found in Chapter 5.4.

5.5.8 ClutteredNB

This scene is similar to the one presented in Chapter 5.5.7, but with the same perspective as Chapter 5.5.5. This provides us with a scene full of objects in different arrangements in relation with each other.

As we can observe in Figure 5.17, one of the objects is highly asymmetrical, having also reflections and transparencies. There is only one surface, and no background. In Figure 5.18 we have zoomed into an area of interest, and fixed the color levels of the image, to better reflect the effect of the cues.

The Luminance cue **(C)** provides a good segmentation for the wineskin, but recognizes the Kinect box and the mug as if they were entirely the same object. For

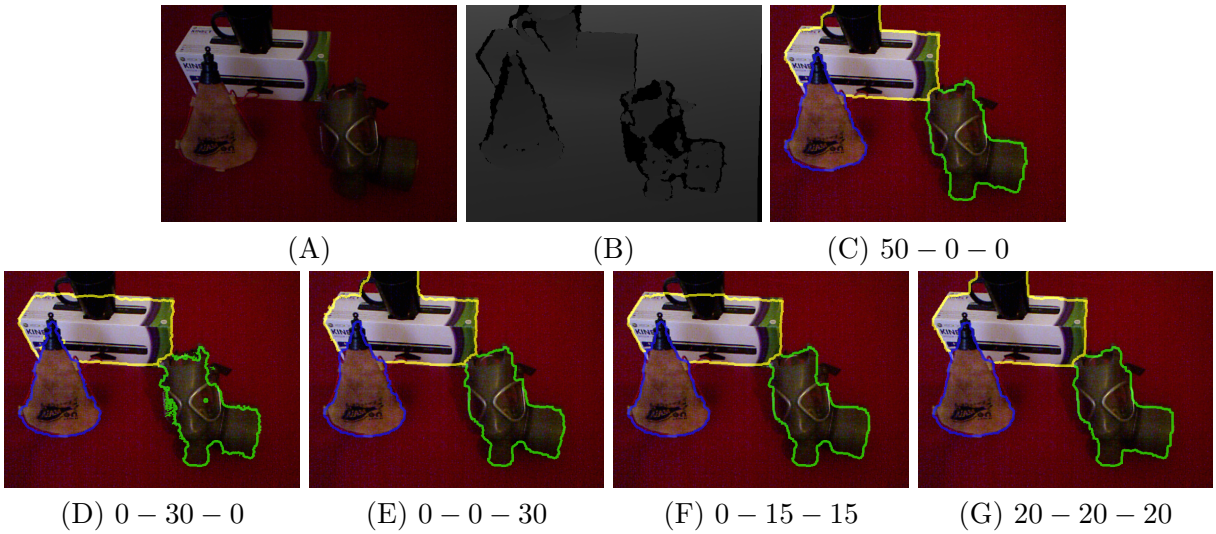


Figure 5.17. ClutteredNB segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** − **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

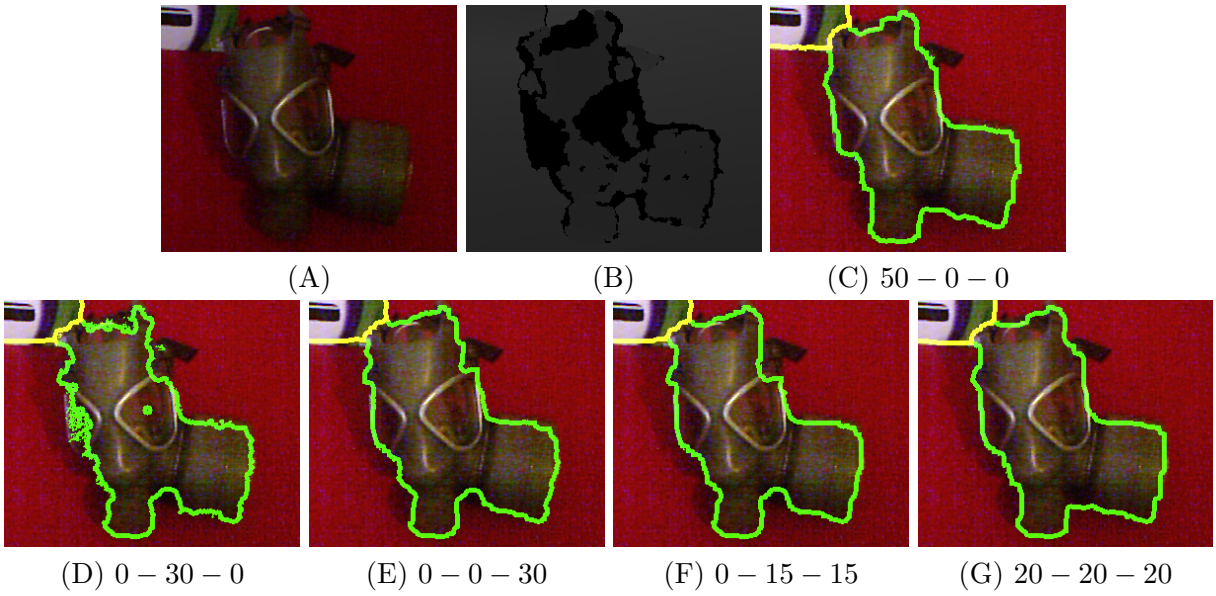


Figure 5.18. ClutteredNB segmentations detail. **(A)**: Color image. **(B)**: Depth image. **(C)** − **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

the the gas mask it does a very good job, although it fails to differentiate between its two lower modules and the shadow from the surface.

The Halo cue **(D)** does a fine job for the wineskin and is able to partially distinguish between the box and the mug. But in the gas mask it fails to do a precise segmentation.

The performance of the Canny cue **(E)** for the whole scene offers a performance similar to that of the Luminance cue **(C)**.

The result of Halo and Canny combined **(F)** only improves Canny **(E)** in partially differentiating between the box and the mug.

In the image with the three cues combined the result is quite similar to Luminance standalone **(C)**. However, the left lens of the gas mask is better segmented in here.

5.5.9 ClutteredNS

In this last scene, we have the same objects and distribution as in Chapter 5.5.7, but with the point of view we used in Chapter 5.5.6. That means that, as we can see in Figure 5.19 **(B)** the depth data of the surface is gone. Also, it is worth commenting that the depth data for the gas mask is even worse than when we studied the scene in Chapter 5.5.6.

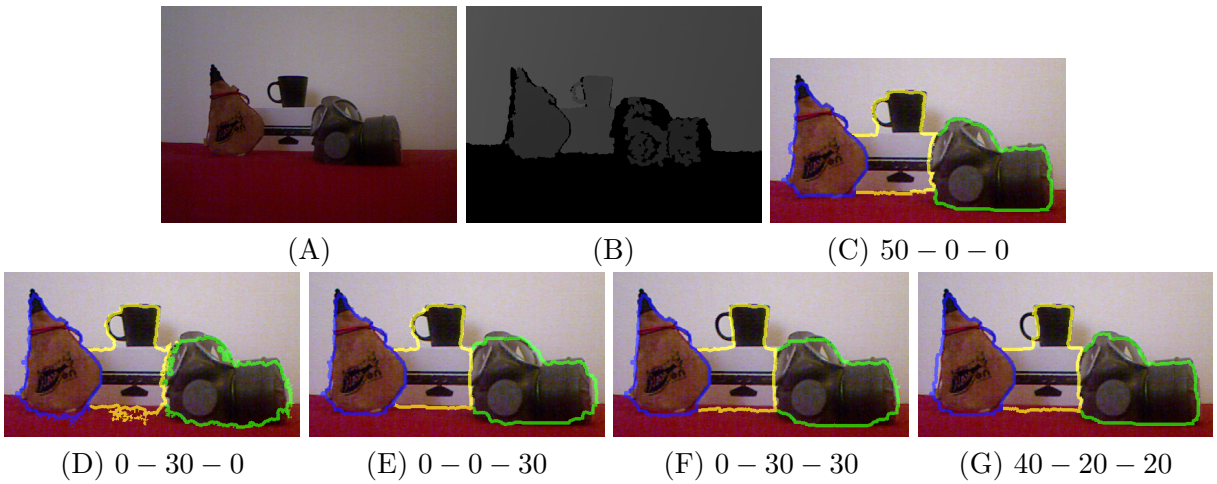


Figure 5.19. ClutteredNS segmentations. **(A)**: Color image. **(B)**: Depth image. **(C)** – **(G)**: Segmentation results for different values of $\lambda^L - \lambda^H - \lambda^C$.

Here, the Luminance cue **(C)** does a very good job, although once again it takes the box and the mug as if they were the same object.

For the depth based cues we see almost a repetition of what happened in Chapter 5.5.6.

Halo **(D)** does a fine segmentation of the wineskin. For the combo box-mug the result is quite similar to the one provided by the Luminance **(C)**, except in the bottom, where we find plenty of noise. For the gas mask, the result lacks accuracy.

Canny **(E)** repeats the same result of Luminance **(C)** with the wineskin, the box and the mug. For the gas mask, the performance is better than Halo **(D)** but not as good as Luminance **(C)**

Using both Halo and Canny **(F)** offers a similar performance as Canny **(E)** standalone for the wineskin, and the only difference with the mug is that less part of the handle is recognized as object. The performance for the gas mask slightly improves, although not reaching the quality of Luminance **(E)**.

The combination of the three cues **(G)**, does not add a substantial change with relation to the segmentation performed by Luminance standalone **(C)**.

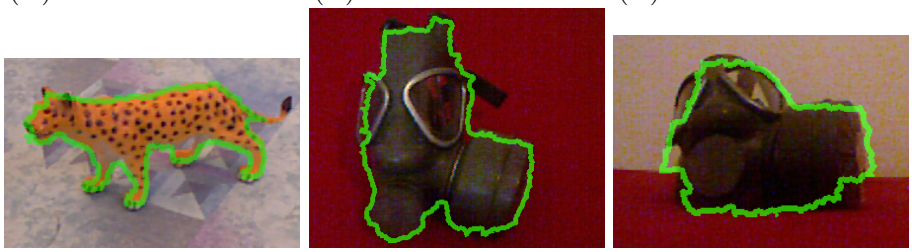
5.6 Gaussian blur

In this section we compare the effect of using a Gaussian blur filter in the Canny and Halo cues with relation to applying them directly, as we saw in Chapter 5.5.

5.6.1 Canny

In almost all the tests performed, applying a Gaussian blur on Canny cue before calculating the energy map showed a worse performance compared with the results obtained by using the cues straight away.

(A) $0-0-30$ Normal (B) $0-0-30$ Normal (C) $0-0-50$ Normal



(D) $0-0-30$ Gauss (E) $0-0-30$ Gauss (F) $0-0-50$ Gauss

Figure 5.20. Canny Gauss. **(A)** – **(C)**: Segmentation results for different images using normal cues. **(D)** – **(F)**: Segmentation results for different images using Gaussian blur.

In Figure 5.20, we see that while in the Leopard scene **(A)** and **(D)** the segmentation actually profits from the transformation, for the other two scenes the performance is worse that using the cue straight away.

5.6.2 Halo

When using the Gaussian blur filter with the Halo cue, the results obtained were also worse than when applying the cue directly. In Figure 5.21, we see that the segmentations using Gauss **(D)** – **(F)** segmented similar areas, albeit with noisier boundaries, than the normal ones **(A)** – **(C)**.

(A) $0-30-0$ Normal (B) $0-30-0$ Normal (C) $0-50-0$ Normal

(D) $0-30-0$ Gauss (E) $0-30-0$ Gauss (F) $0-50-0$ Gauss

Figure 5.21. Halo Gauss. **(A)** – **(C)**: Segmentation results for different images using normal cues. **(D)** – **(F)**: Segmentation results for different images using Gaussian blur.

5.7 Distance transform

Similar to what de did in Chapter 5.6, in this section we will compare the effect of using the distance transform of the Canny and Halo cues with relation to applying them directly, as we saw in Chapter 5.5.

5.7.1 Canny

As it is seen in Figure 5.22, applying the distance transform to Canny does not improve the results of the segmentations, and actually in some cases it just makes them worse.

(A) $0-0-30$ Normal (B) $0-0-30$ Normal (C) $0-0-50$ Normal

(D) $0-0-30$ D.T. (E) $0-0-30$ D.T. (F) $0-0-50$ D.T:

Figure 5.22. Canny distance transform. **(A)** – **(C)**: Segmentation results for different images using normal cues. **(D)** – **(F)**: Segmentation results for different images using distance transform.

5.7.2 Halo

The effect of the distance transform on the Halo cue produces segmentations with steadier cues than the original ones. However, the performance in individual cases is mixed. The relevant examples are shown in Figure 5.23.

In the Leopard scene, the general shape of the object is well segmented with both methods. However, the cue using the distance transform **(D)** fails to identify the space between the legs of the toy while the normal cue **(A)** does a perfect job.

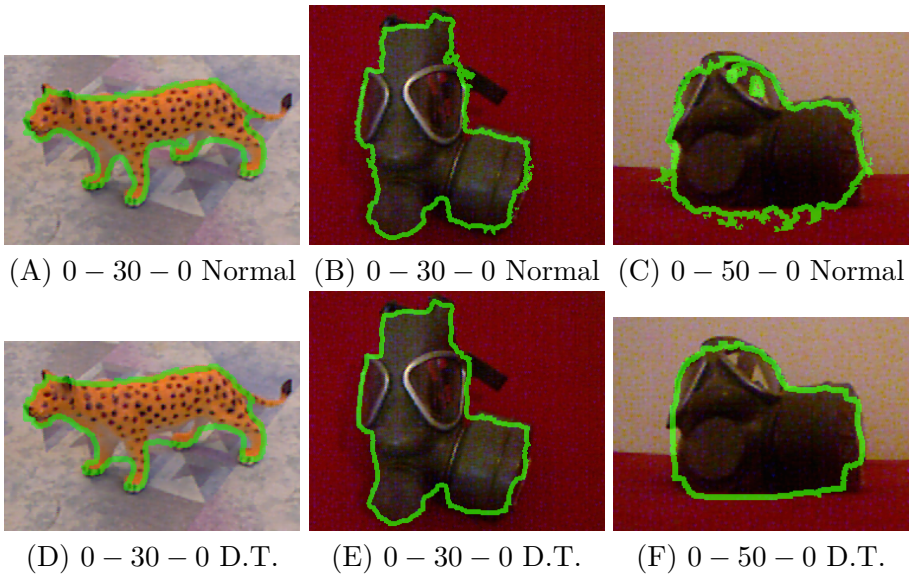
In the GasmaskNB scene, both areas are the same, though, as observed in the general trend, the boundaries shown by the cue using the distance transform **(E)** are less noisy than the ones using the normal cue **(B)**.

But the most surprising example can be found in the GasmaskNS scene. In it we see how the segmented scene by the normal cue **(C)** has plenty of noise, while the distance transform **(F)** is able to eliminate most of that noise and perform a generally more accurate segmentation of the object.

5.7.3 Performance of the transformations

Comparing all the results, the only combination that tends to improve the segmentation process is applying the distance transform to the Halo cue.

Applying the Gauss blur to Canny and Halo decreases the performance of the segmentation. This happens due to the characteristics of the cue after applying



(A) $0-30-0$ Normal (B) $0-30-0$ Normal (C) $0-50-0$ Normal

(D) $0-30-0$ D.T. (E) $0-30-0$ D.T. (F) $0-50-0$ D.T.

Figure 5.23. Halo distance transform. **(A)** – **(C)**: Segmentation results for different images using normal cues. **(D)** – **(F)**: Segmentation results for different images using distance transform.

the filter. As the areas in which the energy function promotes to produce the segmentation get weaker and wider, we loose accuracy in the performance.

For the distance transform, we observe a different performance change from applying it to Canny and to Halo. This is due to the different characteristics between he cues. As in Canny the relevant area is thin, the addition of the gradient 'tail' has a stronger impact than in Halo, were the relevant information usually is wide.

Chapter 6

Discussion

The work presented in this thesis has accomplished the provision of an alternative attention mechanism to select fixations points for the segmentation, as well as two sets of cues based exclusively on the depth data retrieved from the Kinect. The first set of cues is based on the detection of edges in said depth data, while the second set relies on the lack of depth information in specific areas.

6.1 Kinect

With the substitution of the 2 sets of stereo cameras from the original method with the Kinect, the capability of focusing the segmentation effort just on the relevant parts of the scene, a feature that had its advantages, is lost. However, in exchange, a better depth map is obtained.

Everything being taken into account, this change demonstrates to be a successful one, as the depth-based novelties introduced by this thesis prove to be robust and useful, as we will discuss in the rest of this section.

6.2 Symmetry saliency

The symmetry saliency method, as presented in this thesis, looks for symmetrical features in the depth map. It proves to provide idoneous results when the surface and background of the scenes are smooth and well defined. Also, it overcomes problems that might be caused by similarities between the colors of the object and the surface and / or background.

However, as the amount of objects in the scene increases, or in situations in where the surface and / or background have edgy and irregular surfaces, the reliability of the method decreases, proving that it may be valid for certain scenes, but not in the general.

In Chapter 7 we present some ideas on how this could be improved.

6.3 Segmentation cues

In this section we discuss the effects of both sets of cues independently, and also the results achieved by combining them with the original Luminance cue from the framework.

It is worth noting that the results provided by the Luminance cue out of the box were quite satisfactory, although it had problems differentiating objects close to each other.

6.3.1 Canny

Applying Canny straight away, the performance supplied is quite satisfactory, most of the times it is on par with the results offered by the Luminance cue, and on a few cases it is able to outperform it. Its performance is diminished when the depth map lacks much relevant information, but the result is not greatly affected.

While using the Canny cue after applying a Gauss blur filter, the accuracy of the segmentation tends to decrease noticeably. In very particular cases the segmentation is improved, but those are the fewer.

Finally, while using the distance transform, we obtain slightly worse results than when using the cue straight away. Also, the lines in the segmentations prove to be noisier and less steady.

After all the experiments performed, the recommended use of Canny standalone cue is to apply it directly, without any kind of transformation.

Comparing the results of the Canny edge detection with the results provided by the framework, we see that the energy function chosen for the direct application adequately employs the available information to perform the segmentation.

That is not the case for the results provided by the cue after applying either the Gauss or the distance transformations.

6.3.2 Halo

When the depth map provides good information, the performance of the direct application of Halo is on par with the Luminance cue. When the map is lacking relevant information, the results, although covering areas similar to the objects to segment, are extremely noisy.

One strong point of this cue is that it distinguishes between objects close to each other with a better reliability than both Luminance and Canny cues.

The results provided by the Halo cue after smoothing it with a Gauss filter do not offer any improvements to the original cue. The areas segmented are almost the same, but noisier after the smoothing.

The performance from the cue while using the distance transform is surprising, as it is not only able to make steadier and less noisy segmentations, but is also able to improve quite a lot the results for scenes in which the depth map lacks plenty of relevant information.

Given the results shown in the experiments, the recommended use of Halo standalone cue is to use it either directly or after applying the corrected distance transform.

The energy functions seem to be well chosen both for the direct application of the cue and for after using the distance transform, as in both cases the data from the Halo map is correctly translated to its application in the framework.

6.3.3 Combined cues

The combination of different cues between them proves quite satisfactory. Usually, when the two depth-based cues are combined, they provide better results that when they are used independently, with an accuracy on par with Luminance standalone.

Scenes with lack of relevant depth information prove to be harder to segment, but even on those cases, the performance is not far from the results obtained while using just Luminance.

When the three cues, Luminance, Halo and Canny, are used together, the best segmentation results are reached. Even when the depth cues standalone perform worse than the Luminance cue, this one is able to benefit from the new additions to the framework.

Chapter 7

Future Work

Because of the bleeding edge nature of the system this work is based on, both due to the up-to-the-minute status of the framework presented by Björkman and Kragic and its innovative approach, as well as the recent apparition of such hardware devices like PrimeSense Kinect, there is a whole world of possibilities open in front of us.

The software approach is very flexible, and although in the beginning is hard to grasp the specifics behind every block of code due to its high complexity, once it is understood it becomes a very powerful tool to be expanded.

Hardware such as Kinect has appeared very recently, and because of that, its full capabilities are still far from being fully taken advantage of. In the next few years we expect thousands of lines of research, such as the one presented in this report, to be opened based on this kind of devices, exploiting them in every thinkable way.

Also, just as the come into sight of a device that provides a high quality depth map has inspired this new set of cues to be added to the framework, the materialization of new kinds of hardware for this software to rely on, will bring new possibilities to add to the segmentation method.

But based just on the current hardware widely available, and the ending point of this thesis, this are some suitable lines of research to follow:

- In the symmetry saliency mechanism used, the system can be made more robust by finding a suitable way to combine the information obtained from the symmetry from both the depth and the color information. Both ways have their advantages and disadvantages, and they can complement each other.
- While this segmentation method performs well while segmenting objects, it still does not provide us with information about the objects. If we focus into robotics, a good idea would be to integrate it into an object class identification framework. This way we would not be able only to know where an object is, but also what it is.

Bibliography

[1] V. Álvarez Barrientos. *Brain Image Processing for Revealing Early Alzheimer's Disease Signs.* Kungliga Tekniska Hogskölan, 2011.

[2] L. Wang, T. Yang, Y. Tian *Crop Disease Leaf Image Segmentation Method Based on Color Features.* IFIP International Federation for Information Processing, vol. 258; Computer and Computing Technologies in Agriculture, vol. 1, pp. 713-717, 2008.

[3] A. Martínez Usó, F. Pla, P. García Sevilla *Multispectral Image Segmentation for Fruit Quality Estimation.* Proceeding of the 2005 Conference on Artificial Intelligence Research and Development, 2005.

[4] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, H. Chen. *The recognition and tracking of traffic lights based on color segmentation and CAMSHIFT for intelligent vehicles.* IEEE Intelligent Vehicles Symposium (IV), pp. 431-435, 2010.

[5] J.F. Khan, S.M.A. Bhuiyan, R.R. Adhami. *Image Segmentation and Shape Analysis for Road-Sign Detection.* IEEE Transactions on Intelligent Transportation Systems, vol. 12, Issue 1, pp. 83-96, 2011.

[6] R. Cucchiara, M. Piccardi, P. Mello. *Image Analysis and Rule-Based Reasoning for a Traffic Monitoring System.* IEEE Transactions on Intelligent Transportation Systems, vol. 1, Issue 2, pp. 119-130, 2000.

[7] A. Guarnieri, A. Vettore. *Automated Techniques for Satellite Image Segmentation.* Symposium on Geospatial Theory, Processing and Applications, 2002.

[8] B.C. Jiang, C.-C. Wang, D.-M. Tsai, C.-J. Lu. *LCD Surface Defect Inspection Using Machine Vision.* Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference, 2004.

[9] G.S. Gupta, T.A. Win, C. Messon, S. Demidenko, S. Mukhopadhyay. *Defect analysis of grit-blasted or spray-painted surface using Vision Sensing Techniques.* Image and Vision Computing New Zealand, pp.18-23, 2003.

[10] P. Viola, M.J. Jones. *Robust Real-Time Face Detection.* International Journal of Computer Vision 57, pp. 135-154, 2004.

[11] J. Bruce, T. Balch, M. Veloso. *Fast and inexpensive color image segmentation for interactive robots.* Proceedings of International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2061-2066, 2000.

[12] J. Shotton, J. Winn, C. Rother, A. Criminisi. *TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation* 9th European Conference on Computer Vision, 2006.

[13] D.M. Greig, B.T. Porteous, A.H. Seheult. *Exact maximum a posteriori estimation for binary images* Journal of the Royal Statistical Society Series B, vol. 51, No. 2, pp. 271-279, 1989.

[14] D.R. Martin, C.C. Fowlkes, J. Malik. *Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, No. 1, 2004.

[15] M. Harville, G. Gordon, J. Woodfill. *Foreground segmentation using adaptive mixture models in color and depth* Proceedings IEEE Workshop on Detection and Recognition of Events in Video, 2001.

[16] M. Björkman, D. Kragic. *Active 3D scene segmentation and detection of unknown objects.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[17] B. Rasolzadeh, M. Björkman, K. Huebner, D. Kragic. *An Active Vision System for Detecting, Fixating and Manipulating Objects in Real World.* International Journal of Robotics Research (IJRR), 2010.

[18] PrimeSense. *The PrimeSensor^{TM} Reference Design 1.08.* 2010.

[19] D. Kragic. *Image Segmentation and Hough Transform.* DD2423 Image Processing and Computer Vision. Kungliga Tekniska Hogskölan, 2010.

[20] J. Canny. *A computational approach to edge detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 8, 1986, pp. 679-714.

[21] R. Ohlander, K. Price, D. Raj Reddy. *Picture segmentation using a recursive region splitting method.* Computer Graphics and Image Processing, vol. 8, Issue 3, pp. 313-333, 1978.

[22] J. MacQueen. *Some methods for classification and analysis of multivariate observations.* Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297, 1967.

[23] C.R. Brice, C.L. Fennema. *Scene analysis using regions.* Artificial Intelligence, vol. 1, Issues 3-4, pp, 205-226, 1970.

[24] S.L. Horowitz, T. Pavlidis. *Picture Segmentation by a Directed Split and Merge Procedure.* Proceedings International Conference on Pattern Recognition (ICPR), pp. 424-433, 1974.

[25] Y. Boykov, M.-P. Jolly. *Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images.* Proceedings of International Conference on Computer Vision (ICCV), vol. I, pp. 105-112, 2001.

[26] Y. Weiss. *Correctness of local probability propagation in graphical models with loops.* Neural Computation, vol. 12, pp. 1-41, 2000.

[27] R. Potts. *Some generalized order-disorder transformation.* Proceedings of the Cambridge Philosophical Society, vol. 48, pp. 106-109, 1952.

[28] D.Geman, S.Geman, C.Graffigne, P.Dong. *Boundarydetection by constrained optimization.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 7, pp. 609-628, 1990.

[29] C. Rother, V. Kolmogorov, A. Blake *Grabcut – interactive foreground extraction using iterated graph cuts.* ACM Transactions on Graphics (SIGGRAPH), August 2004.

[30] B. Bayer. *U.S. Patent No. 3,971,065.* 1976.

[31] R. Sara. *Finding the largest unambiguous component of stereo matching.* Proceedings 7th European Conference of Computer Vision (ECCV), vol. 2, pp 900-914, 2002.

[32] J. R. Pomerantz. *Colour as a gestalt: Pop out with basic features and with conjunctions.* Visual Cognition, 14(4- 8):619-628, 2006.

[33] G. Kootstra, N. Bergström, D. Kragic. *Using Symmetry to Select Fixation Points for Segmentation.* 20th International Conference on Pattern Recognition (ICPR), pp.3894-3897, 2010.

[34] G. Borgefors. *Distance Transformations in Digital Images.* Computer Vision, Graphics and Image Processing 34, pp. 344-371, 1986.

TRITA-CSC-E 2011:103
ISRN-KTH/CSC/E--11/103-SE
ISSN-1653-5715

www.kth.se