Integrating Bottom-up and Top-Down Information

by

Giuliano Pezzolo Giacaglia

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2013

Author . . . .
Department of Electrical Engineering and Computer Science
January 18, 2013

Certified by . . . .
Patrick Henry Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Integrating Bottom-up and Top-Down Information

by

Giuliano Pezzolo Giacaglia

Submitted to the Department of Electrical Engineering and Computer Science
on January 18, 2013, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

This thesis presents a supervised framework for integrating *Bottom-Up* and *Top-Down* algorithms. This framework was developed with the intuition that for a better model of the human brain it's necessary that the model have tools that mimic the way humans detect and see the world around. The inspiration for the creation of this framework is from the intuition that humans create models of objects around us, and humans use these models to interpret, analyze and discern objects in the world. In this thesis I present 2 implementations of algorithms that were created on top of the presented framework to demonstrate its ingenuity. Both of the algorithms presented in this thesis. The goal of this thesis is to present this novel framework that can integrate different algorithms, and to demonstrate with the presented algorithms that the framework empowers computer vision algorithms and integrates them in a simple way.

Thesis Supervisor: Patrick Henry Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

Acknowledgments

If I were to describe my life in one word that word would be *hustle*. In the urban dictionary hustle is defined as "To do things to get closer to the point you want to get". I've always worked really hard to achieve my aspirations. When I was in Middle School I hustle to get a scholarship for the best high school in Brazil. When I waas in High School I hustle to increase my schorlaship, trying to be amongst the best mathematicians at my age in Brazil. In college in BraziI hustle to survive through the hazing and military training. A semester later at MIT I hustle to pay for food and living. Now I'm hustling to learn and get my Masters.

I was really fortunate to have all these opportunities and I know that there are a lot of people that want to be in the same place but didn't have the same opportunities. Jay-Z in "American Dreamin" explains it well:

"Mama forgive me, should be thinking about Harvard
But that's too far away, niggas are starving
Ain't nothing wrong with my aim, just gotta change the target
I got dreams of bagging snidd-ow, the size of pillows
I see pies every time my eyes clidd-ose
I see rides, sixes, I got to get those
Life's a bitch, I hope to not make her a widow"

MIT was specially hard. It was not only hard because of the endless psets and labs, the midterms and projects, but it was specially hard to do all of this and have to worry about bills. Sometimes there wouldn't be money for lunch.

I've survived. I did it only with the help of my friends and family. I'm extremely thankful to them. I dedicated all my work to them.

To my Mom, Leticia Maria Pezzolo Giacaglia, for all her love. To my dad for all the knowledge and for being an example of honesty and good work ethic. To my brother for always standing on my side.

I want to thank my friends Aldo and Beneah that kept me sane and going. Sometimes we would go partying and "chilling" on the weekend nights so we could forget about the problems of the past week and the problems that we would encounter the next day.

To all my friends in Genesis group, Sam, Hiba, Nikola, Igor, Adam, Matthew, Sila, Mark, Dylan for great discussions that improved this work. But most of all, for all our discussion about other random things that made my experience so great.

To Rafa, Bujao and Pond, for helping me transition to this country.

To Professor Winston, for being an exaamplae of love and care for its students.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The human visual system is a very complex and fascinating system. It is a challenge to create a computer system that achieves the same results as the human visual system achieves. The human visual system has the ability to interpret a lot of data very fast, and because of the way its engineered. For example, most of the information that flows between the eyes and the brain is flowing from the brain to the eyes.

The engineering problem to replicate the visual system is very complex. One important feature of the brain is that it recognizes objects using prior information of the objects.

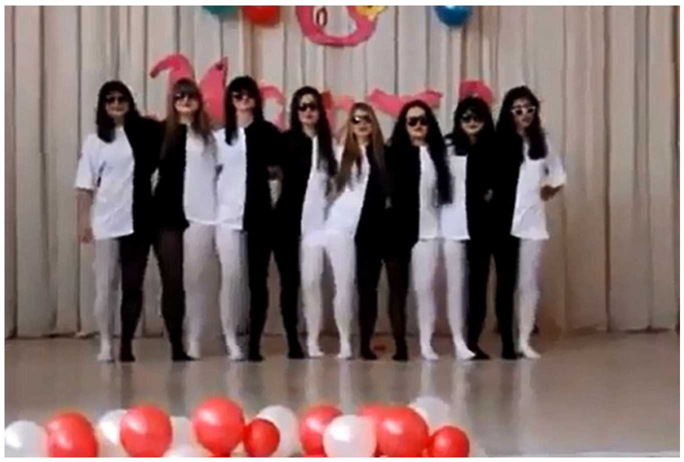


Figure 1-1: This image is a good example of how the human visual system could confuse different objects (people) if it didn't use Top-Down Information.

The figure above is an example of how the brain creates models of the objects in the world around and how integrates that information with the information that the image itself provides.

These two types of inforamtion are called Bottom-Up and Top-Down Information. Top-Down Information is information contained in the model of the objects and high-level structures in the world. Bottom-up information is the information given only by the image.

The problem is how do we integrate these two types of information in a simple model. This model should be simple enough that different types of algorithms created can be coupled easily. This thesis addresses this problem by creating a simple framework to integrate them easily.

1.1 Vision

The framework presented in this thesis addresses the problem of integrating top-down information and bottom-up information. The framework can be used for detecting, segmenting or finding contours of objects inside a single image. In this thesis we present 2 implementation of the presented framework. An exemplification of the implementation of the presented framework is an algorithm for creating a mask on the pixels that belong to humans. Figure 1 presents an exemplification of the implementation of the framework. Figure 1(a) represents the input of the algorithm and figure 1(b) represents the output of the algorithm. Figure 2 represents another set of input and output for another implementation of the presented framework.

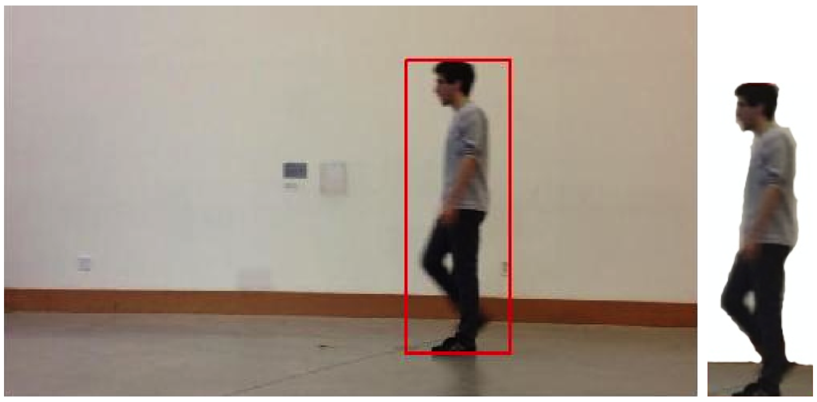


Figure 1-2: Input Image and Output Image for Bottom-Up and Top-Down Segmentation.

1.2 Framework

For simplifying the process of the algorithms, the framework is divided into 3 stages:

1. **Image mask:** The goal of this stage is to filter the input image, therefore allowing the next stage of the algorithm to have a higher accuracy on the creation of the probability map. The image mask is created using Top-Down information, i.e. from a model created of the objects.

2. **Probability map:** The probabily map will be a function from each pixel of the image to a real number from [0,1]. The probability map is created from Bottom-Up information.

3. **Second Image Mask:** This stage uses the model of the object to update the image mask given the probability map.

In any stage of the framework, the model of an object is created from classified training data. An examplification implementation of the framework is presented in Figure 4.

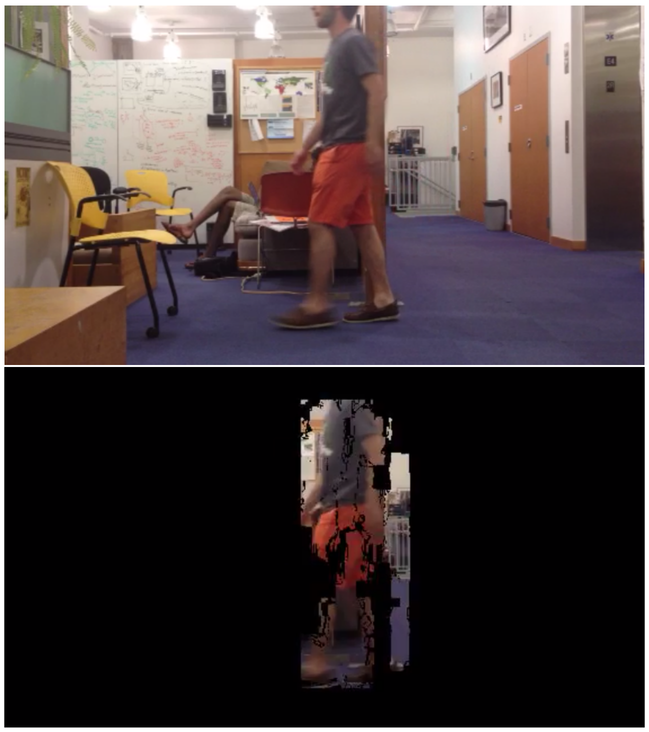


Figure 1-3: These images represent the input and output of a second algorithm using the proposed framework.

1.3 Organization

In **Section 3** the implementation of the creation of HOG filters is explained in detail. In **Section 4** the Top-Down segmentation algorithm, called Ultrametric Contour Map, is explained in detail. **Section 5** presents the implementation of HOG filters for subparts of a specific object. **Section 6** presents the implementation of the Top-Down Segmentation of the image using the model of an object. In **Section 7** the contributions of this thesis are listed.

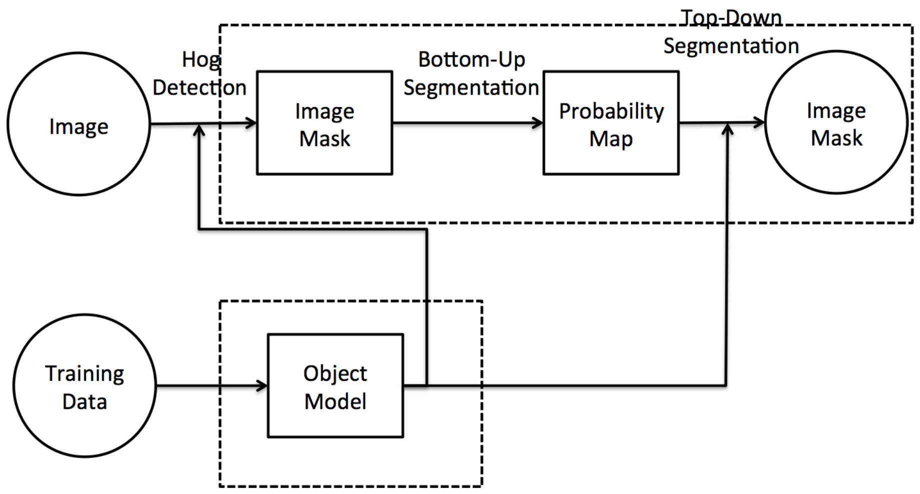


Figure 1-4: This shows an exemplification of the framework presented in this thesis. The first stage is implemented using a HOG Filter of an object and applying the filter to the original image. The second stage implements a Bottom-Up segmentation of the image. The third stage is a Top-Down Segmentation of the image filtering the sunparts of an object with HOG filters for the respective subparts.

Chapter 2

Usage Examples

Image Segmentation: We present a system that learns the model of a object and uses it as a basis for segmenting an image.

Object detection: The framework presented in this thesis can be used for the refination of models for object detection.

3D Reconstruction: With this framework it's possible to create 3D models of objects that are refined to each object class.

The next 2 subsections give a brief overview of the 2 algorithms that were implemented using the described framework. The core differences of the 2 algorithms is that the first algorithm implements a form of n-cuts for bottom-up segmentation of the image, called Ultrametric Contour Map, and the second algorithm does not implement bottom-up segmentation, and for top-down segmentation it implements Ullman's algorithm. The first algorithm is called Ultrametric Contour Map and the second algorithm is called Ullman's algorithm.

2.1 Ultrametic Contour Map

The following list enumerates how this algorithm is implemented based on the framework presented in this thesis.

1. **Image Mask using HOG filters:** For creating the first image mask, the algorithm implements a HOG filter for a determined object.

2. **Bottom-up Segmentation:** For determining the probability map, this algorithm implements Ultrametic Contour Map presented by Arbaleaz et Al. in ARBALAEZ. Figure 5 represents the input and the output of this algorithm.

3. **Image mask with Top-Down Segmentation:** For the last stage of this algorithm, the image will be masked using different HOG filters. Each HOG filter will represent a subpart of the image.

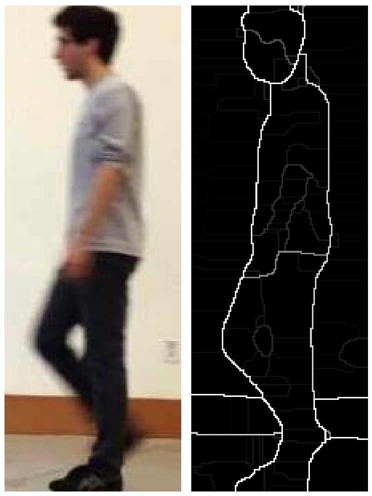


Figure 2-1: The input and the output of running the Ultrametric Contour Map algorithm. The output is a probability map. Figure 5(b) represents the probability map. The more white the pixel, the closer to 1 is the value of the pixel, anagolously the more black the pixel is, the closer the value at that pixel is to 0.

2.2 Ulman's Algorithm

The stages of the 2nd algorithm are the following:

1. **Image Mask using HOG filters:** This stage is the same as the one presented in the first algorithm.

2. **Probability Map:** This algorithm will skip this step and it will the constant unitary function fo every pixel.

3. **Image mask with Top-Down Segmentation:** This algorithm filters the image by creating a set of fragments that represent the determined object. The fragments will be laid on top of the image by finding the correspondent region with the highest correlation to the finding the fragments that have highest correlation with the image.

Chapter 3

Previous Work

Object Detection and Image Segmentation are important areas of Computer Vision. Most of the papers dealing with the problem solve each one separetely. Object detection can help segmention an image and vice-versa. In this chapter I describe some of the most influential and the state-of-the-art work on both areas.

3.1 HOG Detection

Work done by Dalal and Triggs [?] and work done by Felzenszwalb [?] identify objects based on histograms of oriented gradients.

[?] improves the algorithm by creating a root filter and part filters for subparts of the object. Deformations mixture models help to create a multi-view representation of a determined object.

3.2 Image Segmentation

Work done by Arbelaez et Al segments natural images with a seed point [?] and without one [?] inside the object of interest.

[?] defines a metric between regions inside the image. This metric creates a hierarchical image segmentation.

3.3 Top-Down Segmentation

[?] segmented the images using training data to represent the edges and the sorroundings of the edges of object classes. Each set of edge and the respective boundary is called a fragment. For each object class [?] extracts the fragments that represent the best the class object.

Given a new image [?] compares the fragments that it has with the image and try to superimpose the fragments, like a jigglesaw puzzle. With the fragments in place, [?] can infer where the edges should be, and therefore it segments the image.

3.4 Object Detection from Contour

[?] implemented object detection using the contours of the images. First, they detect the edges of a bunch of training images and create a model of the contour of objects by defining the center of the object and creating fragments of the contours of the objects. After creating a model of the possible fragments that represent the contour of the object. The edges of an image are extracted using the Canny Edge Detector.

Given a new image, they try to find what is the point that represents the center of the object. The center of the object will be such that the distance between the distance between the contours of the representation of the object and the edges of the image being compared is minimal. Then, the distance between the two images is the distance when the center is at the defined location.

3.5 Chamfer Distance

The chamfer distance is calculated as the sum of the distances between each pixel and the set of edges. The distance between each pixel and the set of edgels is equal to the distance of the pixel and the closest pixel if the images were superimposed.

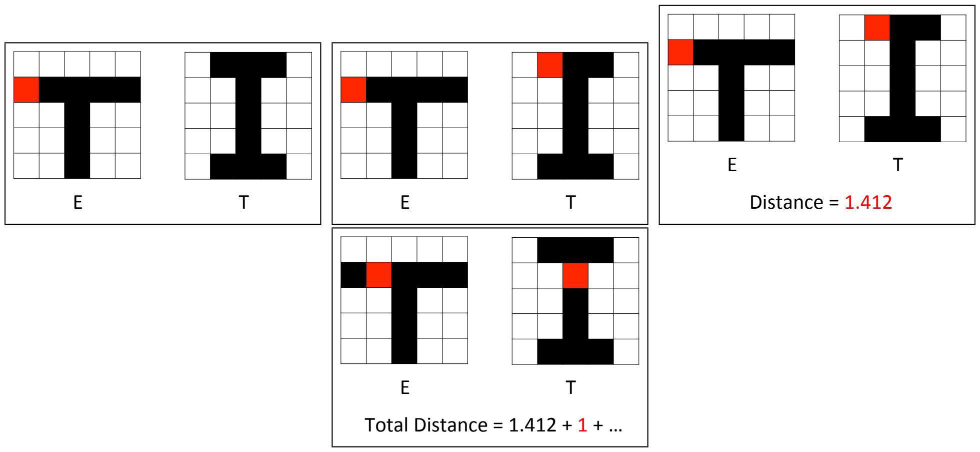


Figure 3-1: This is an example of how to calculate the chamfer distance between 2 sets of edges.

Chapter 4

HOG filters

The creation of a HOG filter for a determined object can be divided into 5 parts:

1. Compute gradients;
2. Orientation Binning.
3. Descriptor Blocks
4. Contrast normalize over overlapping spatial blocks;
5. Collect HOG's over detection window;
6. Linear SVM.

Ater the creation of the SVM, a new data point is classified as an object by running the 4 initial parts of the algorithm and classifying the new data point using the created SVM.

4.1 Compute gradients

For computing gradients, the algorithm runs two 2D masks [-1,0,1] for the x-axis and y-axis for each color channel. There is no need to do Gaussian smoothing in the image. Gaussian smoothing does not improve performance of HOG filters as shown by [**?**].

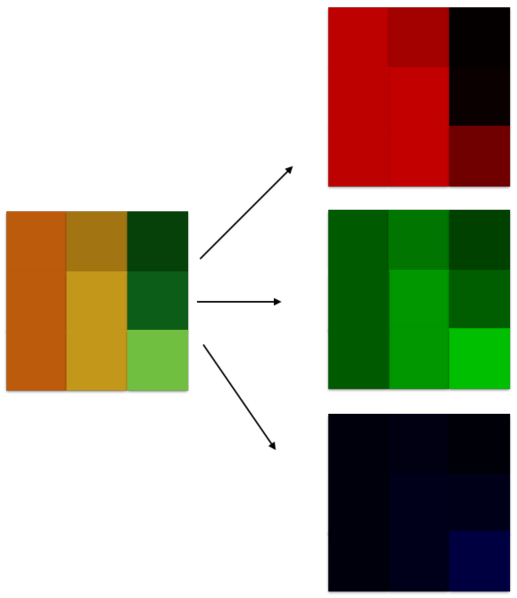
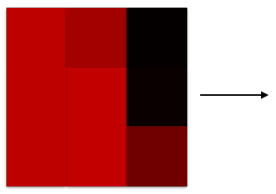


Figure 4-1: Representation of a 9x9 to the RGB colors. Each 9x9 square on the right of the image represent the magnitude of the color on the respective color space.



189	163	5
189	195	11
189	195	112

Figure 4-2: The red color channel to the correspondent value block.

4.2 Orientation Binning

The algorithm discretizes the space of angles into 9 bins between the angles 0^o and 180^o degrees (”unsigned” gradient). Figure 8 represents the vote of the set of pixels. The weight of the vote of each pixel is equal to the magnitude of the gradient of each pixel. Each pixel will be added to a different bin by its proximity to it.

4.3 Descriptor blocks

Because there is a great variation of the strenght of the illumination and because of the big difference of constrast between the foreground image and the background

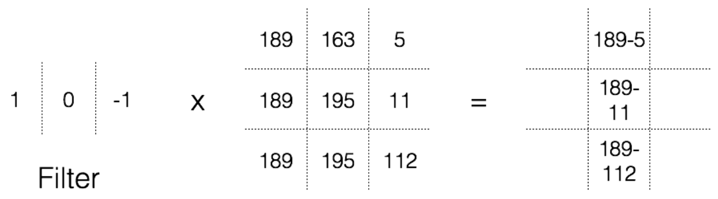


Figure 4-3: Multiplying the first matrix with the horizontal 2D mask [1,0,-1].

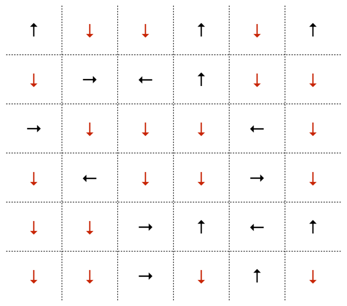


Figure 4-4: Orientation binning is done in 6x6 pixel cells. Each pixel will contribute to one direction of the gradient. The direction of the pixel at the center of the cell is determined by voting of the pixels on that cell. In this example, the orientation of the bin will be down.

image, it's important to have blocks that integrate information of a set of neighbor pixels. The set of neighbor sets are called blocks. The blocs will be normalized to reduce the impact of these differences when identifying objects.

In this paper I will use blocks of size 16x16 pixels. Each block will therefore contain four 8x8 pixel cells.

4.4 Contrast normalize over overlapping spatial blocks

Each vector of pixels will be normalized to reduce the impact of brightness of certain areas when calculating the gradients. Each vector v of pixels will be normalized by:

$$\hat{v} = \frac{v}{||v||_2}.$$

4.5 Collect HOG's over detection window

The HOG's are collected over a 64x128 window.

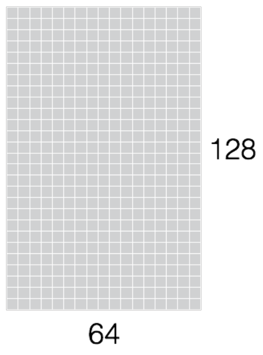


Figure 4-5: The window size of each block.

4.6 SVM

After collecting the data. The data is compared to other data points using a linear SVM.

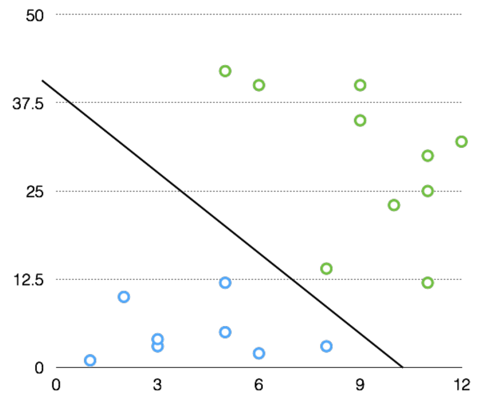


Figure 4-6: The training data.

Chapter 5

Image Segmentation

Image semgmentation is one of the essential parts of the algorithm. Image segmentation is the process of partitioning an image into multiple regions, known as super-pixels.

For Image Segmentation we will use the same technique presented by Arbalaez et Al. in [**?**]. The technique is a variant of Normalized Cuts.

Normalized Cuts is an algorithm that associates a distance metric between every pixel and its neighbors. The algorithm finds cuts that maximize the distance between two regions. The distance between regions is defined as the sum of all pixels that were separated by the cut.

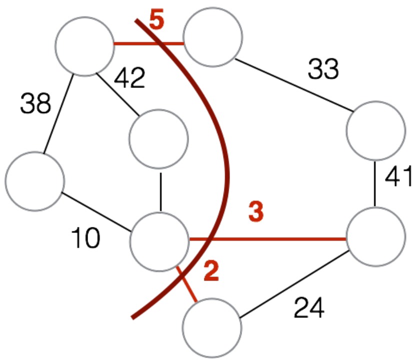


Figure 5-1: Minimum Cut with sum equal to 10.

Definition: Given an affinity matrix W whose entries encode the similarity between pixels, one defines diagonal matrix $D_{ii} = \sum_j Wij$. Solving for the generalized eigenvectors of the following linear system:

$$(D - W)v = \lambda Dv$$

After this step, each pixel is represented on the basis of the eigenvectors and then K-Means clustering is applied to cluster the regions.

The above solution usually breaks regions that have smooth gradients. Even though it breaks smoothness, the above solution provides a very good clue where the regions should be. The solution given by Arbalaez et Al. is to use the same idea but to separate regions by boundaries that are not smooth.

Ultrametric Contour Map

Formally, a *Hierarchical Segmentation Operator* (HSO) is a function from (P_0, λ) to the partition P_λ with the following properties:

$$P_\lambda = P_0, \forall \lambda \leq 0$$

$$\exists \lambda_1 \in R : P_\lambda = \Omega, \forall \lambda \geq \lambda_1$$

$$\lambda \leq \lambda' \Rightarrow P_\lambda \sqsubseteq P_{\lambda'}.$$

The initial graph is equal to $G = (\mathcal{P}_0, \mathcal{K}_0, W(\mathcal{K}_0))$. The nodes are the initial regions, i.e. each pixel is a node, the edges are the contours $\mathcal{K}_0$ separating adjacent regions , and the weights of the edges $W(\mathcal{K}_0)$ are a measure of dissimilarity between regions. The algorithm merges regions that are the most similar, iteratively.

To calculate the Hierarchical Segmentatio Operator we first calculate the distance between every pixel at differente orientations:

Given the *Hierarchical Segmentation Operator* (HSO) in terms of the contour, demonstrated in Figure **??**, the *Ultrametric Contour Map* (UCM) can be defined the real-valued image obtained by weighting each boundary by its scaled of disappearance.

The *Ultrametric Contour Map* is used to define the distances between regions.

Figure **??** presents an example of a UCM map constructed from the Hierarchical Segmentation Operator.

5.1 Distance

5.1.1 Each angle

To calculate the distance $G(x, y, \theta)$ between 2 pixels we get a circle of radius σ and calculate:

$$G(x, y, \theta) == \frac{1}{2} \sum_i \frac{(g(i) - h(i))^2}{g(i) - h(i)}.$$

For each direction θ we have a function of the distance between neiborhood pixels. To smooth out the distance, I apply a second-order Savitzky-Golay filter on the direction of the angle.

5.1.2 Summing up

$$mPb(x, y) = max_\theta(\sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)(x,y,\theta)})$$

5.1.3 Eigenvectors

Within a fixed radius r we calculate:

$$W_{i,j} = e^{-max_{p \in \bar{xy}} mPb(p)}$$

r is 5 in this case and is 0.1.

This step is done so that neigborhood pixles that edges that are distant apart, have a magnitude that is small. The function e^{-x} is represented in the figure below:

The eigenvectors are found through the following formula:

$$(D - W)v = \lambda v$$

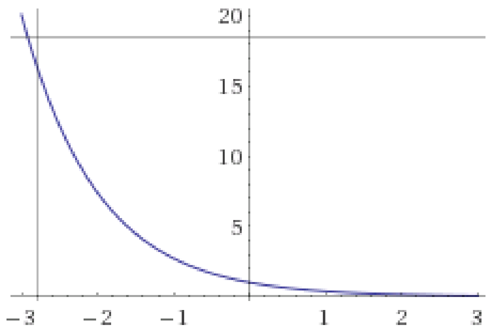


Figure 5-2: e^{-x} from -3 to 3.

5.1.4 New Distance

The n eigenvectors with the biggest eigenvalues have the correspondent eigenvectors that represent the highest variability axis of the points. I pick the highest $n = 16$ eigenvectors, as [?] did. The eigenvectors are combined, and we get the following:

$$sPb(x, y, \theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} v_k(x, y)$$

$$gPb(x, y, \theta) = \left(\sum_s \sum_i \beta_{i,s} G_{i,\sigma(i,s)(x,y,\theta)} \right) + \gamma sPb(x, y, \theta)$$

5.2 Oriented Watershed Transform

I assume that the previous algorithm outputs a function $E(x, y, \theta)$, for every position (x, y) and angle θ. The following part of the algorithm does not depend on how the previous algorithm but only on what it outputs.

Given $E(x, y, \theta)$ I calculate the maximum over all the angles:

$$E(x, y) = max_\theta E(x, y, \theta)$$

I estimate an orientation at each pixel on an arc from the local geometry of the arc itself. The orientations are obtained by approximating the watershed arcs with line segments.

We recursively subdivide any arc which is not well fit by the line segment connecting its endpoints. By expressing the approximation criterion in terms of the maximum distance of a point on the arc from the line segment as a fraction of the line segment length, we obtain a scale-invariant subdivision. We assign each pixel x; y on a subdivided arc the orientation; of the corresponding line segment.

Next, we use the oriented contour detector output to assign each arc pixel a boundary strength of We quantize in the same manner aso this operation is a simple lookup. Finally, each original arc in K0 is assigned weight equal to average boundary strength of the pixels it contains. Comparing the middle left and far right panels in Fig. 12 shows that this reweighting scheme removes artifacts.

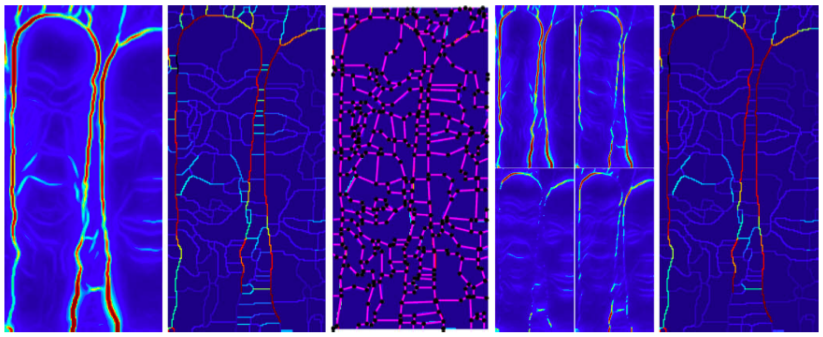


Figure 5-3: Image extracted from [?] exemplifying the process of Oriented Watershed Transform.

Chapter 6

HOG filters for the subparts

The goal of this stage is to annotate the parts of the image that belong to the object and the parts of the image that do not belong to the object. For processing the parts of the image separately, the image will be separated into parts by taking the Ultrametric Contour Map, and given a threshold λ, it will output 0 or 1 for each pixel.

For each pixel p, the function will return $f(p) = 1$ if the ultrametric contour map returns a value for that pixel that is greater or equal to λ. And $f(p) = 0$ if the ultrametric contour map returns a value for that pixel that is less than λ.

Each subpart of the object will be detected using a HOG filter for each of the subparts. There is a probability assigned to each subpart being part of the object. This will be the prior of the each subpart of the object.

The subparts that have an associated HOG Filter are:

1. Face
2. UpperBody
3. Lowerbody

The reason for the choice of these 3 subparts is that each subpart is usually together when the image is segmented.

Figure 6-1: Set of image faces used for training the HOG filter model for the human face.

Chapter 7

Ullman's algorithm

7.1 Training Data

The training data consist of a set of pairs of images. The pair of images contain an image of the object class and an image of its contour.

7.1.1 Contour

The contours will be of the size of the image. The contour image will contain zeros and ones. The ones represent the contour of the image.

7.2 Finding the Contour

Given the input of the image we want to find if each pixel is either in the contour or not, i.e. for every pixel p if $I(p) = 1$ or $I(p) = 0$.

To be able to do that, we get each fragment from the Training data and correlate to parts of the image to detect if that part of the image is part of the image and if its a contour or not.

7.2.1 Correlation

The distance between two regions of the same size is done by first changing the representation of the image from the RGB color channel to the LAB color channel. Then, the distance of two regions R_1 and R_2 is calculated as:

$$\sum_{ij \in R_1} (R_1(i,j) - R_2(i,j))^2$$

The second way of calculating the distance between two regions is to cluster different colors and then calculate the distance between the histograms of the regions:

$$\frac{1}{2}\sum_{k=1}^{K} \frac{(h_1(k) - h_2(k))^2)}{h_1(k) + h_2(k)}.$$

7.3 Textons

For improving the algorithm I added textons as information for comparing different fragments. Textons aid when representing the same kind of material, therefore the distance between textons will represent well the distance between different object materials.

To calculate textons we follow the following formula, for each pixel we calculate the following vector:

$$v_i = p * F_i,$$

where the filter i is represented by i.

I calculate the response for all the training data and then cluster all the vectors for all pixels into 32 clusters using K-means.

Each pixel will (then) have an associated texton.

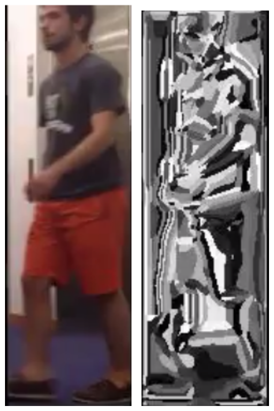


Figure 7-1: Input Image and Textons extracted from the image.

7.3.1 Distance

Let R_1 and R_2 be two regions of the same size, then their distance can be calcualted as follows:

$$D(R_1, R_2) = \frac{1}{2} \sum_{k=1}^{K} \frac{(h_1(k) - h_2(k))^2)}{h_1(k) + h_2(k)},$$

The formula above calculates the distance between the distribution of the histogram distribution for the two regions.

7.3.2 Example

For example, the figure below shows a histogram of two different regions.

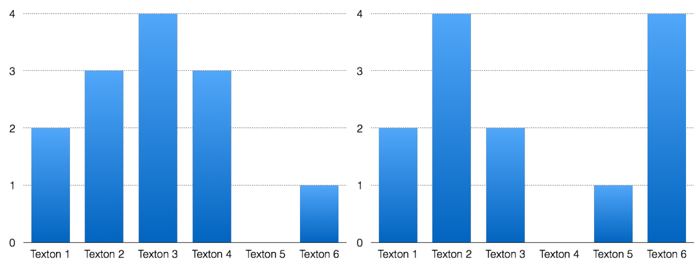


Figure 7-2: Two different histograms representing how many textons of each type are in each region.

The difference between the regions is:

$$\frac{1}{2} \times \frac{(2-2)^2 + (4-3)^2 + (4-2)^2 + (3-0)^2 + (0-1)^2 + (1-4)^2}{(2+2) + (4+3) + (4+2) + (3+0) + (0+1) + (1+4)}$$

$$= \frac{1}{2} \times \frac{0+1+4+9+1+9}{4+7+6+3+1+5}$$

$$= \frac{1}{2} \times \frac{24}{26} = \frac{12}{26} \approx 0.46$$

7.3.3 New Distance

The new distance will be the weighted sum of the distance between the LAB color space and the distance between the textons of the pixels.

7.4 Best matching

Given many fragments and a image we want to find the best matching fragment on top of that image, like a jiggpuzzle. We calculate the distance between the fragment and all the fragments on that image. We find the minimum of those distances.

The chosen fragment will be the one with the minimum between the minimuns. We add the k fragments with minimum distance to the image fragments. The output will be the covered part of the image by the k fragments.

7.5 Fragments

With the contour and the image in hands, we get fragments of the foreground. If the image is inside of the contour, it means that it is a fragment of the image. The fragments will be represented as the fragments themselves. The pixels that belong to the object class are white. The pixels that are not part of the object are black. The fragment and the cover is represented at

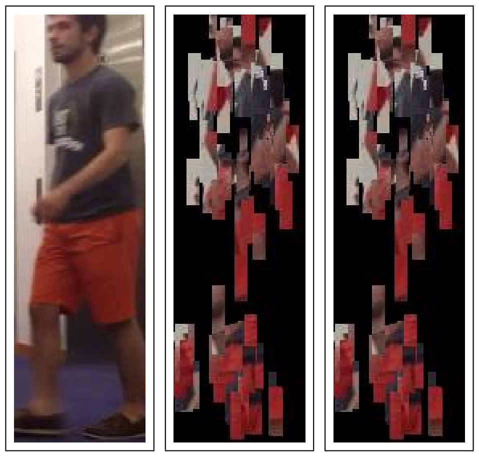


Figure 7-3: The first image is the block that represents the image of a person. The second image shows what happens when the fragments are superimposed without using the information of the textons. The third image shows the superimposition using textons.

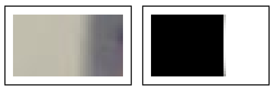


Figure 7-4: Fragment represented and the fragment cover at the top of the image.

7.6 Algorithm

For each image the following is done:

```
$\max_{Fragments} = -\infty$ ;
for $F \in Fragments$ do
    $max_F = -\infty$ ;
    $p_f = (0,0)$ ;
    while not end of the figure do
        $v = I_{[(x,x+\delta x),(y,y+\delta y)]} * F$;
        if $max_F < v$ then
            $max_F = v$ ;
            $p_f = (x,y)$ ;
        end
        Move $(x,y)$ ;
    end
    if $max_{Fragments} < max_F$ then
        $max_{Fragments} = max_F$ ;
    end
end
```

Algorithm 1: *Finding the relation between a fragmnet and an image.*

We pick the positions that maximize the value max_F.

Chapter 8

Training Data

For training the algorithm I use the PASCAL dataset and the training data was divided into 3 different parts:

1. Annotations
2. Image Sets
3. JPEG Images

Each one of the different parts will be described below:

8.1 Annotations

The annotations will have the information of the parts that we are creating the HOG filter of. We will create HOG filters for:

1. Face
2. LowerBody
3. UpperBody

An example of such a file is
$< annotation >$

```
        < folder > GIU2013 < /folder >
        < filename > face1 < /filename >
        < source >
        < size >
                < width > 375 < /width >
                < height > 500 < /height >
                < depth > 3 < /depth >
        < /size >
        < segmented > 0 < /segmented >
        < object >
                < name > Face < /name >
                < pose > Unspecified < /pose >
                < truncated > 0 < /truncated >
                < difficult > 0 < /difficult >
                < bndbox >
                        < xmin > 0 < /xmin >
                        < ymin > 0 < /ymin >
                        < xmax > 375 < /xmax >
                        < ymax > 500 < /ymax >
                < /bndbox >
        < /object >
< /annotation >
```

8.2 Image Sets

Image Sets contain files that will represent the training data. File will be a list of files, with the respective annotation if it belongs to the object class being classified. For example, the face object class has a file name face.xml and with the following contents:

face1.xml 1 face2.xml 1

...

lowerbody123.xml -1

lowerbody124.xml -1

8.3 JPEG Images

8.4 Results

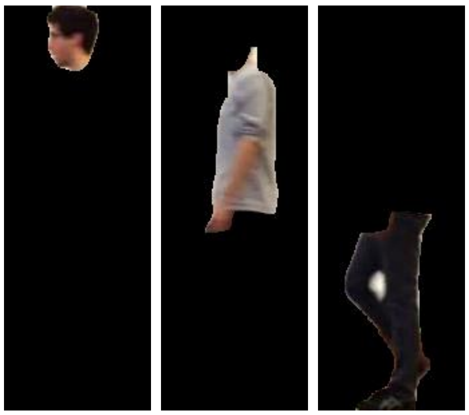


Figure 8-1: The positive images.

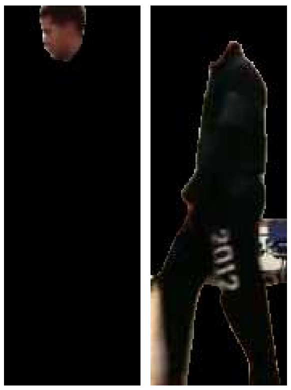


Figure 8-2: The positive images.

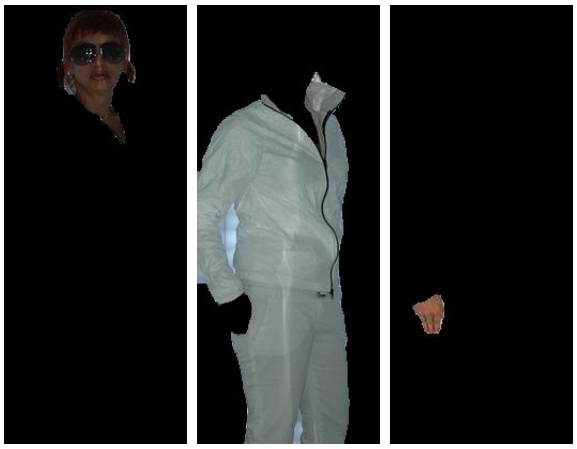


Figure 8-3: The positive images.

Chapter 9

Contributions

- Created algorithm that implements a combination of *Bottom-Up* and *Top-Down Segmentation.*