Hyperspectral Image Segmentation by t-Watershed and Hyperspherical Coordinates

Ramón Moreno[1]

Computational Intelligence Group
Universidad del País Vasco, UPV-EHU
http://www.ehu.es/ccwintco

September 11, 2012

[1] ramon.moreno@ehu.es

Introduction

- This work presents a segmentation method based in a tuned version of watershed transform.
- It is based on a chromatic gradient that is made through Hyperspherical Coordinates.
- It avoids the natural oversegmentation induced by the standard watershed
- It overcomes shadows and shines in order to elude false edges.

Introduction

The method is summarized below:

1. Transform the image to **Hyperspherical coordinates**
2. Make the **chromatic gradient**
3. Apply the **t-Watershed** method

Hyperspherical Coordinates

- Let us denote p a hyperspectral pixel color in n dimensional Euclidean space.
- In Cartesian coordinates it is represented by $p = \{v_1, v_2, v_3, ..., v_n\}$ where v_i is the coordinate value of the i-th dimension.
- This pixel can be represented in Hyperspherical coordinates $p = \{l, \phi_1, \phi_2, \phi_3, .., \phi_{n-1}\}$, where l is the vector magnitude that gives the radial distance, and $\{\phi_1, \phi_2, \phi_3, .., \phi_{n-1}\}$ are the angular parameters.

Hyperspherical Coordinates

- This coordinate transformation is performed uniquely by the following expression, for all cases except the ones described below:

$$
\begin{aligned}
l &= \sqrt{v_1^2 + v_2^2 + v_3^2 + \ldots + v_n^2} \\
\phi_1 &= \cot^{-1} \frac{v_1}{\sqrt{v_2^2+v_3^2+\ldots+v_n^2}} \\
\phi_2 &= \cot^{-1} \frac{v_2}{\sqrt{v_3^2+v_4^2+\ldots+v_n^2}} \\
&\vdots \\
\phi_{n-2} &= \cot^{-1} \frac{v_{n-2}}{\sqrt{v_{n-1}^2+v_n^2}} \\
\phi_{n-1} &= 2\cot^{-1} \frac{\sqrt{v_{n-1}^2+v_n^2}-v_{n-1}}{v_n}
\end{aligned}
,
$$

Exceptions: if $v_i \neq 0$ for some i but all of $v_{i+1}, \ldots, v_n$ are zero then $\phi_i = 0$ when $v_i > 0$. When all $v_i, \ldots, v_n$ are zero then ϕ_i is undefined, usually a zero value is assigned.

Hyperspherical Coordinates

- A more compact notation for the hiperspherical coordinates is $p = \{l, \bar{\phi}\}$, where $\bar{\phi}$ is the vector of size $n-1$ containing the angular parameters.
- Given a hyperspectral image $\mathbf{I}(x) = \left\{ (v_1, v_2, v_3, \ldots, v_n)_x ; x \in \mathbb{N}^2 \right\}$, where x refers to the pixel coordinates in the image domain, we denote the corresponding hyperspherical representation as $\mathbf{P}(x) = \left\{ (l, \bar{\phi})_x ; x \in \mathbb{N}^2 \right\}$, from which we use $\bar{\phi}_x$ as the **chromaticity** representation of the pixel's and l_x as its (grayscale) **intensity**.

Hyperspherical Coordinate

- According to the aforegoing coordinate transformation, we can perform the following hyperspectral separation.
- Given a hyperspectral image $\mathbf{I}(x)$ in the traditional Cartesian coordinate representation we can compute the equivalent hyperspherical representation $\mathbf{P}(x) = \left\{(l, \bar{\phi})_x ; x \in \mathbb{N}^2\right\}$.
- Then, we can construct the separate **intensity image** $\mathbf{L}(x) = \left\{(l)_x ; x \in \mathbb{N}^2\right\}$
- And the **chromaticity image** $\mathbf{C}(x) = \left\{(\bar{\phi})_x ; x \in \mathbb{N}^2\right\}$.

Chromatic gradient operator

For two pixels p and q we compute the Manhattan or Taxicab distance on the chromatic representation of the pixels:

$$\angle(p,q) = \sum_{i=1}^{n-1} \left|\bar{\phi}_{p,i} - \bar{\phi}_{q,i}\right| \tag{1}$$

Note that the $\angle(\mathrm{C}_p, \mathrm{C}_q)$ distance is always positive.

Chromatic gradient operator

The **row** pseudo-convolution operator is defined as

$$CG_R\left(\mathrm{C}\left(i,j\right)\right) =$$

$$\sum_{r=-1}^{1} \angle\left(\mathrm{C}\left(i-r,j+1\right),\mathrm{C}\left(i-r,j-1\right)\right),$$

and the **column** pseudo-convolution is defined as

$$CG_C\left(\mathrm{C}\left(i,j\right)\right) =$$

$$\sum_{c=-1}^{1} \angle\left(\mathrm{C}\left(i+1,j-c\right),\mathrm{C}\left(i-1,j-c\right)\right),$$

so that the color distance between pixels substitutes the intensity subtraction of the Prewitt linear operator.

The hyperspectral **chromatic gradient magnitude** image is computed as:

$$CG(x) = CG_R\left(x\right) + CG_C\left(x\right) \tag{2}$$

t-Watershed Algorithm

The algorithm inputs:

1. A *gradient* image (IG)
2. A initial *threshold* (thr)
3. The *amount of iterations* (steps).

The outputs are :

1. The watershed image (WS)
2. An image with the labeled regions (IL).

t-Watershed Algorithm

- First, it initializes the output images, and defines the intensity jump for thresholding into the image gradient.
- By using a threshold, it finds the regions with minimal gradient, and helped by the primitive 'bwlabel' initializes the image of labels.
- Then the algorithm begins with the flooding process who is going to finish after 'steps' iterations
  - It calculates the new threshold (thi), and using it on the image gradient, finds the new pixels to label
  - For each pixel who is not labeled already, it finds into the respective neighborhood if some of them has a label
  - Depending of the labels found into the neighborhood, algorithm does different things:
    - If there is not labels, it creates a new label and assigns it to the current pixel
    - If there is only a label, it assigns it to the current pixel
    - If there are several labels
    - – If the gradient intensity is lower than the parameter 'thr', merges all regions in a label and assigns it to the current pixel
    - – In other case, it marks it as watershed pixel

Experiment 1

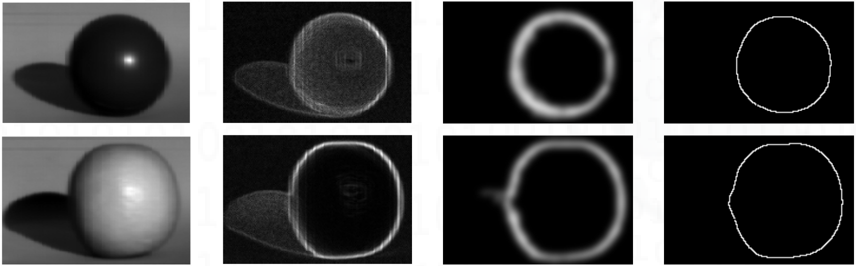

Figure: t-Watershed segmentation of images taken with SOC 710 camera

Experiment 2

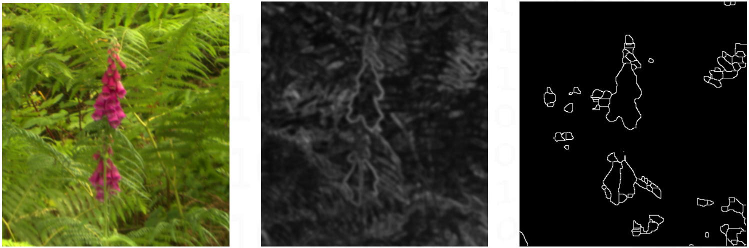

Figure: t-Watershed segmentation of some images from the Foster's database.

Experiment 3

Figure: t-Watershed segmentation of some images from the Foster's database.

Experiment 4

Figure: t-Watershed segmentation of some images from the Foster's database.

Experiment 5

Figure: t-Watershed segmentation of some images from the Foster's database.

Conclusions

- It uses an adjustment of the watershed transform (t-Watershed).
- It avoids oversegmentation of regions with poor or high gradient, in the first case by using a threshold and in the second case by using Gaussian blurring.
- It shows a good behavior avoiding shadows and shines, it is leaded by DRM looking for true surface: the diffuse component.
- It shows a good behavior too in natural scenes.

Thanks for your attention!