Image Segmentation using Mean Shift

Yang Ziyi

November 24, 2017

1 Implementation Details

1.1 Features processing

Our algorithm makes use of three types of features: spatial information, LAB color information and texture information.

Given an RGB color image, we first transform it into LAB color format, and use these 3 channels as features.

The texture features are obtained by first transforming the image into a grayscale one, then filtering the grayscale image with 8 oriented even- and odd-symmetric Gaussian derivative filters and a DoG (difference of Gaussians) filter. [1] It is optional to include them as features.

Each feature channel is then normalized to assume values in $[0, 1]$, with one exception: for spatial features, the shorter side is normalized to $[0, 1]$, while the longer side is scaled accordingly so that pixels remain squares in the feature space.

1.2 Mean shift

In theory, at each mean shift step, every point that defines the underlying probability density function needs to be taken into account. However, since the Gaussian distribution decays exponentially, points that are far away have little impact on the mean shift step.

Our algorithm first calculates a distance $\boldsymbol{d}$ according to σ (standard deviation of the Gaussian distribution), and then at each mean shift step only takes into account points that are within this distance (in the feature space).

More concretely, at each mean shift step, the current spatial position is rounded to a grid point $\boldsymbol{p}$, and points that are spatially within cretain **range** from $\boldsymbol{p}$ are examined: if in the feature space its distance from current state is smaller than $\boldsymbol{d}$, it will be taken into account.

In principle the **range** should be set larger than $\boldsymbol{d}$. But in practice, we find that fixing the **range** to a smaller value (in order to reduce computational cost) does no harm. By doing this, we are sort of using a mixture of truncated Gaussian distributions as the underlying probability density function.

1.3 Acceleration

We also implement an accelerated version of mean shift [2], which is based on the observation that trajectories followed by different points towards the same mode collect close together. In reverse, trajectories that go close together tend to converge to the same mode.

The algorithm goes as follow. The image plane is discretized into subpixel cells. As one point iterates, we record its spatial trajectory in the cells. And if its trajectory meets an previous trajectroy, we halt the iteration of this point, and assign the mode that previous trajectory converges to to the current trajectory.

2 Documentation

2.1 Major functions

compute_features.m turns an RGB image into LAB color features and texture features (optional). This function depends on:

- **texture_features.m**, whick takes in a grayscale image and output a multi-channel image storing the texture features.

rescale_augment.m normalizes the feature channels to $[0, 1]$, and add spatial features as 2 new channels. The output will be used in:

- **meanshift.m**
- **meanshift_fast.m**

meanshift.m takes in a multi-channel image (representing a collection of pivot points in the feature space, which defines the underlying probability density function), and for each point, computes its convergence point in the feature space following the mean shift algorithm. Thus the output is still a collection of points in the feature space.

coloring.m takes in the output of **meanshift.m**, and perform a simple clustering of the points according to one criterion: two points belong to the same cluster if and only if they are within certain distance from each other. Its output is a coloring scheme, each pixel assume an integer value indicating which cluster it belongs to. This function depends on:

- **wander_explore_multi_channel.m** starting from a point, wander around to visit all points that belong to the same cluster, so that we can color them.

meanshift_fast.m takes in the same input as **meanshift.m**, and performs the accelerated version of mean shift as described in section 1.3. Its output is a coloring scheme, each pixel assume an integer value indicating which cluster it belongs to.

2.2 Helper functions

apply_filter.m A wrapper of conv2() that applies to multi-channel images.

gaussian_filter.m generates Gaussian filters

SearchPoints.m generates a list of relative grid positions that need to be examined at each mean shift step.

color_def.m defines Kelly's 22 colors of maximum contrast.

hex2rgb.m turns hex color format into RGB color, this function is written by Chad Greene [3].

myimshow_cluster.m visualizes a coloring (clustering) scheme.

inspection.m A 3D visualization of the LAB color feature space, with each point colored by a coloring (clustering) scheme.

merge_patches.m Given a coloring (clustering) scheme, this function allows the user to merge some of the clusters by simply clicking at them in the image and press the Enter key.

recoloring.m After we merge some patches in an image, the coloring scheme may has some color that correspond to no pixel. This function will recolor the image to remedy such a situation.

color_pick.m Given a coloring (clustering) scheme, this function allows the user to know the color of a pixel by clicking at it in the image.

demo.m demo2.m demo3.m Some demonstrations.

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[2] Miguel A Carreira-Perpinán. A review of mean-shift algorithms for clustering. *arXiv preprint arXiv:1503.00687*, 2015.

[3] Chad Greene. MathWorks File Exchange rgb2hex and hex2rgb. `https://www.mathworks.com/matlabcentral/fileexchange/46289-rgb2hex-and-hex2rgb`. Accessed: 2017-11-23.