HOG Descriptor in MATLAB

09 May 2013

To help in my understanding of the HOG descriptor, as well as to allow me to easily test out modifications to the descriptor, I wrote functions in Octave / Matlab for computing the HOG descriptor for a detection window.

You can find the source code at the project page on GitHub. Or you may simply download a zip of the project directly here.

HOG Tutorial

For a tutorial on the HOG descriptor, check out my HOG tutorial post.

Source files

`getHOGDescriptor.m` - Computes the HOG descriptor for a 66x130 pixel image / detection window. The detection window is actually 64x128 pixels, but an extra pixel is required on all sides for computing the gradients.

`getHistogram.m` - Computes a single 9-bin histogram for a cell. Used by 'getHOGDescriptor'.

Octave code is compatible with MATLAB, so you should also be able to run these functions in MATLAB.

Differences with OpenCV implementation

- OpenCV uses L2 hysteresis for the block normalization.
- OpenCV weights each pixel in a block with a gaussian distribution before normalizing the block.
- The sequence of values produced by OpenCV does not match the order of the values produced by this code.

Order of values

You may not need to understand the order of bytes in the final vector in order to work with it, but if you're curious, here's a description.

The values in the final vector are grouped according to their block. A block consists of 36 values: 1 block * 4 cells / block * 1 histogram / cell * 9 values / histogram = 36 values / block.

The first 36 values in the vector come from the block in the top left corner of the detection window, and the last 36 values in the vector come from the block in the bottom right.

Before unwinding the values to a vector, each block is represented as a 3D dimensional matrix, 2x2x9, corresponding to the four cells in a block with their histogram values in the third dimension. To unwind this matrix into a vector, I use the colon operator ':', e.g., A(:). You can reshape the values into a 3D matrix using the 'reshape' command. For example:

```
% Get the top left block from the descriptor.
block1 = H(1:36);

% Reshape the values into a 2x2x9 matrix B1.
B1 = reshape(block1, 2, 2, 9);
```

Send your feedback

Please let me know if you find any bugs, opportunities for optimization, or any other discrepancies from the original descriptor.

Related posts

Concept Search on Wikipedia 22 Feb 2017

Getting Started with mlpack 01 Feb 2017

Word2Vec Tutorial Part 2 - Negative Sampling 11 Jan 2017