Efficient Stop & Warning Sign and Pedestrian Detection

Castleberry, Cherry, and Firth

December 4, 2012

Overview

- We implemented the following:
  - Stop-sign detector, using the notion of integral images from SURF
  - Warning sign detector, using perspective transformation on a model image, then SURF

Wrappers

- We created a wrapper to the pedestrian detector to gain familiarity with the EmguCV library.
- We also created a wrapper to the built-in EmguCV SURF detector to use as a basis of comparison for our own methods.


STOP
ONE
WAY


DO NOT
ENTER

Stop Sign Detection, Integral Images

- We developed a method for detecting stop signs based upon the use of integral images which we encountered in the SURF algorithm.
- To summarize the method, we use integral images from both the left-hand side (top left) and the right-hand side (bottom right) on only the R channel. Then we consider only the LHS and RHS integral images along the diagonal of the image. We difference these, then fit a Gaussian curve to the resulting vector. We threshold the curve at the points of inflection to generate a bounding box for the sign.


STOP
ONE
WAY

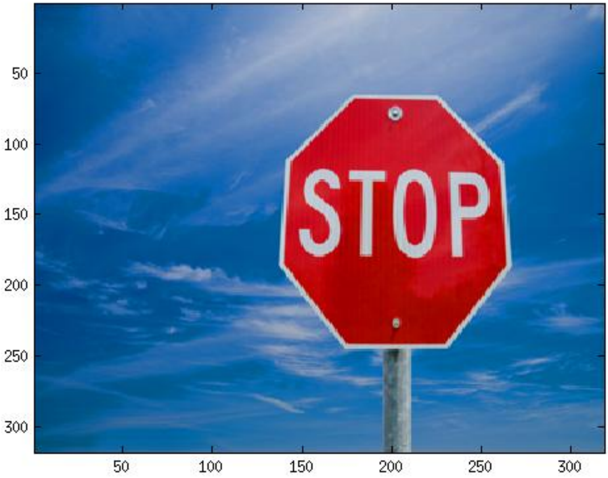
Stop Sign Detection, Integral Images

- To begin our method, we scale the $N \times M$ image to $N \times N$ for $N < M$, $M \times M$ for $M < N$. We require a square matrix to extract a particular vector.
- Recall that the formula for computing the integral image $\mathcal{I}_-$ at a pixel (x, y) with intensity value $I(x, y)$ is:

$$\mathcal{I}_-(x, y) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} I(x, y) \tag{1}$$

- We compute the integral image using the following formula:

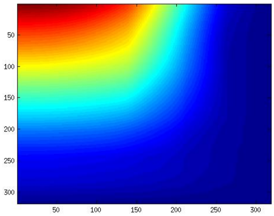
$$\mathcal{I}_{x,y} = \mathcal{I}_{x-1,y} + \mathcal{I}_{x,y-1} - \mathcal{I}_{x-1,y-1} \tag{2}$$

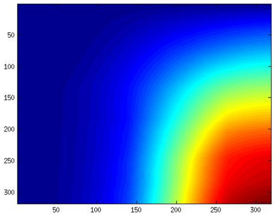

STOP
50
100
150
200
250
300
50
100
150
200
250
300

Stop Sign Detection, Integral Images

- Likewise, we compute an RHS integral image $\mathcal{I}_+$ at a pixel (x, y) with intensity value $I(x, y)$ as:

$$\mathcal{I}_+(x, y) = \sum_{i=N}^{n_x} \sum_{j=N}^{n_y} I(x, y) \tag{3}$$


50
100
150
200
250
300
50
100
150
200
250
300


50
100
150
200
250
300
50
100
150
200
250
300

Stop Sign Detection, Integral Images

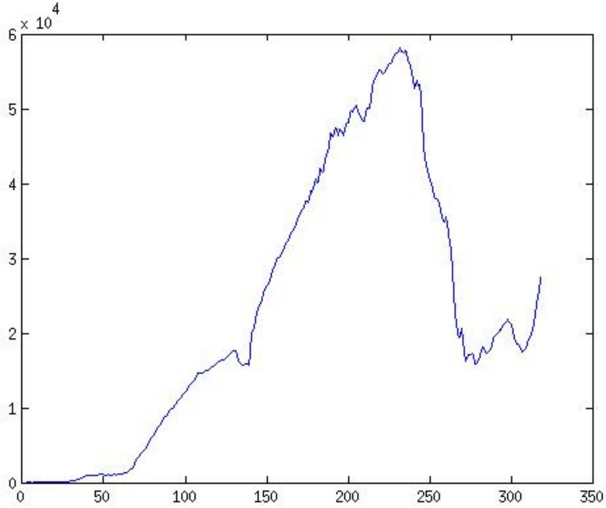
- After obtaining the integral image, we copy its diagonal into a vector u.
- We then apply a finite-difference method to the elements in u and store it in v, as follows:

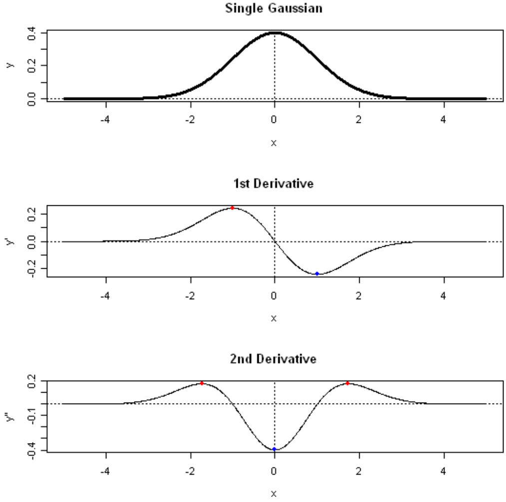
$$v_n = u_n - u_{n-1} \tag{4}$$

The vector v gives the LHS crosshair of the R-channel. For images which have stop signs, v has a Gauss distribution.

Stop Sign Detection, Integral Images

- We apply this finite-difference method for both vectors u_- and u_+ to obtain v_- and v_+.
- Then, we add v_- and v_+ to obtain a vector m.
- Finally, we compute the standard deviation σ of the vector m and its centroid c, then apply a Gaussian fit to the data in m. We call the Gaussian fit G.
- We then apply finite-differencing to the Gaussian fit G to obtain G', then find the indices at $min(G')$ and $max(G')$.
- These indices form the bounding box for the image.


x 10^4
6
5
4
3
2
1
0
0
50
100
150
200
250
300
350

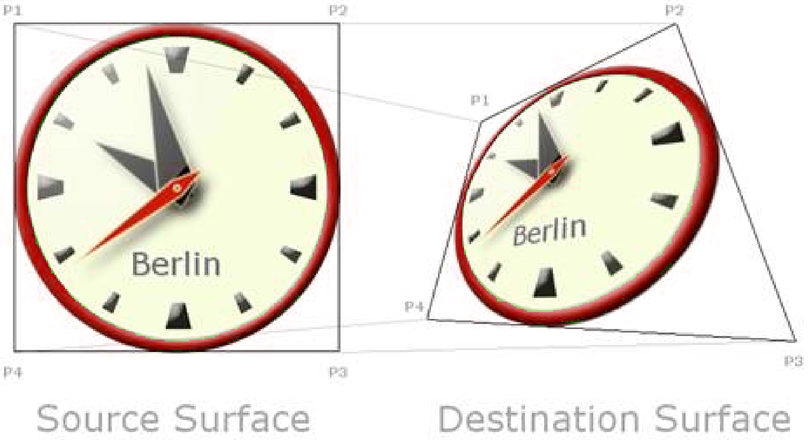

Single Gaussian
y
0.0
0.2
0.4
-4
-2
0
2
4
x
1st Derivative
y'
-0.2
0.0
0.2
-4
-2
0
2
4
x
2nd Derivative
y''
-0.4
-0.1
0.2
-4
-2
0
2
4
x

Warning Signs

- For our warning sign detection, we experimented with SURF in combination with perspective transformations on out-of-plane-rotated signs.
- In particular, we assume that we know the perspective information of an out-of-plane-rotated sign. We apply a perspective transform to a model sign, then apply SURF to the two images, then match.

Warning Signs

- We first hand-annotated N images using a MATLAB code.
- From this, we extracted a set of $4N$ points which give the vertices of the warning sign. We then applied `getPerspectiveTransform()` on the points, which returns a perspective transform matrix `M`.
- We apply this perspective transform matrix to the model with the function `warpPerspective()`.
- We then run SURF on the two images, then compare their feature descriptors to obtain a match within a threshold of t_ϵ.


P1
P2
P4
P3
Berlin
P1
P2
P4
P3
Berlin
Source Surface
Destination Surface

Perspective Transform

- If $a_{x,y,z}$ is the point to be projected, $c_{x,y,z}$ is the camera, $\theta_{x,y,z}$ is the camera orientation and $e_{x,y,z}$ is the position of the viewer relative to the display surface then $b_{x,y}$, the 2D projection of a, is given by:

(5)

- We give an example on the following slide.

Results

- The following table summarizes our results:

A	B	C	D

- As is evident in the above, our accuracy for . . . was an abysmal N%.