Table of Contents

Edge Detection

Edge points are pixels at or around which the image values undergo sharp variations. There are three steps to edge detection. First, it is required that we suppress noise in some way, then detect and enhance edges, and finally localize them accurately.

To perform noise suppression, we may convolve the image with a 2D Gaussian kernel, for instance. Edge enhancement may involve the design of filters that respond to edges. Localization involves deciding which local maxima in the filter's response field are edges. This involves thinning and thresholding on local maxima. A very well known edge detection algorithm is Canny's edge detector.

Edge Descriptors

An edge descriptor would contain attributes such as a) the normal direction to the edge (and conversely, its direction), b) the edge center, and c) the edge strength. The edge center describes the location of the edge, while the edge strength is a measure of local image contrast at the edge location.

Model of the Ideal Step Edge

Mathematically, a step edge (in 1D here) may be described by a discontinuous function:

$$G(x) = \begin{cases} 0 & x<0 \\ A & x\geq 0 \end{cases}$$

We generally assume that the edge enhancement filter is linear, and that the image noise is additive, white-Gaussian. In designing algorithms for noise detection, we need to minimize the probability of false positives. In addition, locating the edge must be bias-free.

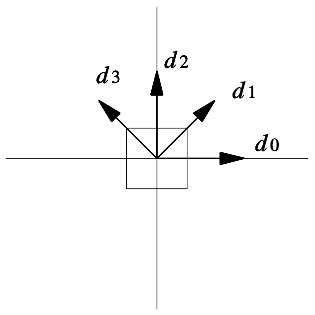
Canny's Edge Detection Algorithm

The algorithm's input is an image I . Let G be a Gaussian with zero mean and standard deviation σ . Then apply a convolution of the original image with the Gaussian. And, for each pixel $J(i,j)$:

- Compute the gradient $\nabla J = (J_x, J_y)^T$
- Estimate the edge strength $e_s(i,j) = \|\nabla J\|_2$
- Estimate edge normal direction $e_o(i,j) = \tan^{-1}\left(\frac{J_y}{J_x}\right)$

Non-Maximum Suppression

Consider the two outputs from Canny's detector, e_s and e_o , and the four basic directions (d_0, d_1, d_2, d_3) from an image location (i,j) .
We need to find d_k that approximates best $e_o(i,j)$, the edge orientation. If $e_s(i,j)$ is smaller than one of its two neighbors along d_k , then $I_n(i,j)=0$, else $I_n(i,j)=e_s(i,j)$ (See figure on the right).



To further enhance edge detection, we can use hysteresis thresholding. The input to this enhancement is I_n , the edge map that has been non-maximum suppressed, $e_o(i,j)$, the edge orientation, and two thresholds, τ_l and τ_h , such that $\tau_l < \tau_h$. Then, for all points in I_n , and scanning it in a fixed order:

- Locate the next unvisited edge point $I_n(i,j)$ such that $I_n(i,j) > \tau_h$
- Starting at that point, follow the chain of connected local maxima in both directions perpendicular to the edge normal, as long as $I_n(i,j) > \tau_l$
- Mark all visited points, and save a list of the locations of all points in the found contour

The output of this method is a set of lists, describing the positions of connected contours.

An Example

Here is an OpenCV example program that uses Canny's edge detection algorithm to extract image edges:

```
#include <stdio.h>
#include <opencv2/opencv.hpp>
```

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv/cvaux.h>


IplImage* doCanny(
              IplImage* in,
              double   lowThresh,
              double   highThresh,
              double   aperture)
{
   if (in->nChannels != 1)
      return(0); // Canny only handles gray scale images
   IplImage* out = cvCreateImage(
                          cvGetSize( in ),
                          in->depth, //IPL_DEPTH_8U,
                          1);
   cvCanny( in, out, lowThresh, highThresh, aperture );
   return( out );
};

int main( int argc, char** argv )
{
   IplImage* img_rgb = cvLoadImage( argv[1] );
   IplImage* img_gry = cvCreateImage( cvSize( img_rgb->width,img_rgb->height ), img_rgb->depth,
1);
   cvCvtColor(img_rgb, img_gry ,CV_BGR2GRAY);
   cvNamedWindow("Example Gray", CV_WINDOW_AUTOSIZE );
   cvNamedWindow("Example Canny", CV_WINDOW_AUTOSIZE );
   cvShowImage("Example Gray", img_gry );
   IplImage* img_cny = doCanny( img_gry, 10, 100, 3 );
   cvSaveImage(argv[2],img_cny) ;
   cvShowImage("Example Canny", img_cny );
   cvWaitKey(0);
   cvReleaseImage( &img_rgb);
   cvReleaseImage( &img_gry);
   cvReleaseImage( &img_cny);
   cvDestroyWindow("Example Gray");
   cvDestroyWindow("Example Canny");
}
```

Here are the input and output images:

Jacksonville
Google