Vision Recognition

April 13, 2011

The quadrotor we are preparing for the challenge is required to operate in a GPS denied area. Without GPS, only a few tools are available to the quadrotor to calculate its location. The quadrotor will really principally on a laser scanner which will simultaneously scan its surroundings and build a map of its surroundings. This is done using a Simultaneous Localization and Mapping algorithm(SLAM). A key step of this algorithm is known as loop closure. As the quadrotor gets further away from its initial location, the uncertainty of its current position increases. This uncertainty can be greatly minimized if the quadrotor recognizes a location that it has been before. This allows it to update the entire map so that it becomes an accurate representation of the real thing.

Loop Closure

We achieve loop closure by regularly taking a sample picture from the video feed on the quadrotor and comparing it to a database. Instead of containing images, the database contains thousands of features that were found in sample images. The amount of computing power required to build the database is large, which makes it impractical to create the database from images that were captured on site, instead sample pictures are taken beforehand of places with similar surroundings.

SURF

The loop closure program works by first extracting features from an image. There are several different types of features that can be extracted, but the method that we used is called SURF, which extracts features that are scale and rotation invariant. These features are often abstract or very small so don't expect to understand what features the computer is seeing. For information on how SURF works we recommend you go to http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/. It explains the SIFT algorithm, which is the algorithm which SURF is based on. Our code is written in OpenCV, which already has a SURF implementation.

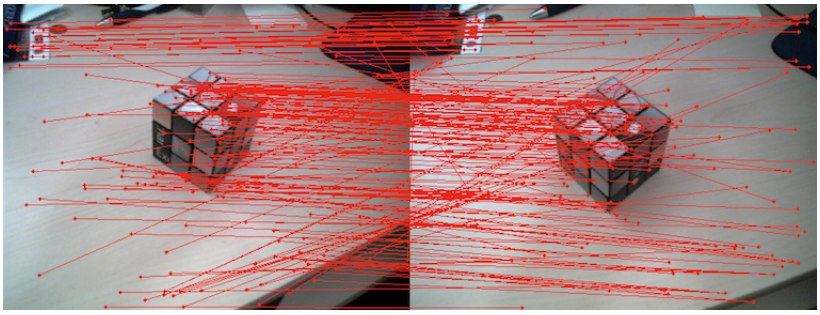

Figure 1: Surf Features on a Rubiks cube

The SURF function outputs two vectors. The first vector is called 'imagedescriptors'. It contains a 64 bit description of each feature found in the image(making it 64 by N). The other output is a vector called imagekeypoints. This vector stores the location of each feature found on the image. To make it so the computer does not have to process every single feature (There will be potentially hundreds of thousands) The features in the images are grouped into clusters using k-means.

K-means

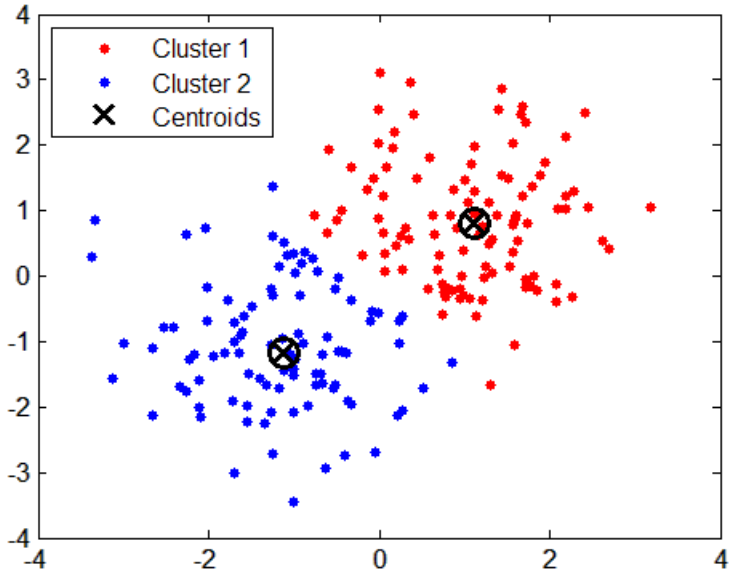

Figure 2: K-means Centers

The features found in an image will not be evenly distributed. Instead they will be clustered around objects found in the image as shown in Figure 2. The K-means algorithm is a method of finding the location of the center of each cluster on the image. Each feature is assigned to a k-means center based on its proximity which is calculated using the Mahalanobis distance (eq. 1). We create a vector that contains the number of

the center that each feature is closest to(we'll call it label). For example, say we have an image that outputs a imagedescriptors vector that is 140 features long. Each feature in this vector will be labeled in the second vector according to the center it is nearest to.

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} C_2 \\ C_3 \\ C_1 \\ \vdots \end{bmatrix}$$

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)} \tag{1}$$

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_i)^T(x_i - \mu_i) \tag{2}$$

D_M = Mahalanobis distance
S = covariance matrix
x = imagekeypoints
μ = k-means centers
n = number of features