1. **Background Subtraction**
   1. **Definition**

Background subtraction is the way that the system keeps only the hand on images captured from camera, and then the system can detect and keep track the hand through camera.

* 1. **Define Problem**

Images captured from camera that is quite complex in color and these maybe contains the user’s hand or not so we need to remove the colors not belong to the hand for detecting and recognizing.

* 1. **Goal**
* This solution must produce an image containing only the hands within black background and a binary image.
* Example:
  + Image containing the hands
  + Binary Image
  1. **Solution (Phần này hình ảnh ví dụ tui sẽ đưa vào sau)**

To solve this problem, we need to have a background image as a sample, then we will remove the colors belongs to the sample from the following images captured from camera.

We should follow these steps to process background subtraction:

* Specify sample background color:
  + Capture an image as background sample.
  + Convert BGR background sample image to LAB sample image.
  + Get all color pixels from lab sample image.
* Define color range of LAB sample color:
  + With lightness dimension of LAB color space, the range is

[value – 40, value + 20]

* + With A and B for the color-opponent dimensions, the range is

[value – 15, value + 15]

* Create a binary image with black background.
* Check color pixels of the following images is between sample color range:
  + With every single color pixel, if lightness dimension (L) and color-opponent dimensions (A,B) is in color range of background, that means this pixel does not belong to hand.
* Remove the color pixels is between color range:
  + Change all pixels not belong to hand to black color.
* Remove noises and smooth the contour:
  + After the previous step, we have a binary image containing some noises and the contour is not smooth so we need to remove them by morphology transformations.
* Find the hand in binary image:
  + Now we have a binary image containing not only hand but also arm or something else so we need to find out the hand by finding the biggest contour.
  1. **Definitions in image processing**
     1. **BGR Image**
     2. **LAB Image**
     3. **Binary Image**
     4. **Contour**
  2. **Used Image Processing Algorithm**
     1. **Blur Image – Gaussian Blur**
     2. **Convert BGR image to LAB image**
     3. **Morphology Transformations – Opening**
     4. **Find Contour**
  3. **Flow chart**

1. **Features Extraction**
   1. **Definition**

Features extraction is the way to specify 81 features of hand for recognition.

* 1. **Define Problem**

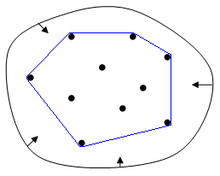
At this time, we just have an image containing only hand and a binary image containing the hand contour and we need to find the best features from these two images.

* 1. **Goal**
* This solution must produce three binary images containing 81 features for hand sign recognition.
* Example
  1. **Solution (Phần này hình ảnh ví dụ tui sẽ đưa vào sau)**

To solve this problem, we need to convert the BGR image containing hand which is produced from the background color subtraction into binary images which can describe most of features.

We should follow the following steps to extract the features of hand:

* Convert the BGR image containing the hand to binary image containing features:
  + To process it we will apply the Adaptive Gaussian Threshold algorithm.
* Produce the binary image containing the hand palm from the binary image containing contour:
  + We will find the point inside the contour and calculate the nearest distance from the point to the contour, and then we find the farthest point from those nearest distance.
  + From the farthest point and the distance, we can draw an circle similar to the hand palm
* Produce the binary image containing the finger lines from the binary image containing contour:
  + First, we find the convex hull of the contour
  + After that, we find the convexity defects of the contour from the convex hull.
  + From these convexity defects we can draw the finger lines
* Calculate the features from these binary images:
  + Calculate the ratio between the height and width of hand
  + Calculate the three ratios between the number of black pixels of three vertical areas below and the total black pixels of binary image.
  + Calculate the three ratios between the number of black pixels of three horizontal areas below and the total black pixels of binary image.
  + Calculate the four ratios between the number of black pixels of four square areas below and the total black pixels of binary image.
  + Calculate the four ratios between the number of black pixels of four triangle corners below and the total black pixels of binary image.
  + Calculate the eight ratios between the number of black pixels of eight vertical areas below and the total black pixels of binary image.
  + Calculate the sixteen ratios between the number of black pixels of sixteen square areas below and the total black pixels of binary image.
  + Calculate the angle features, that means the ratio between the number of angle created by hand fingers and the constant max number of angle (4)
  + Calculate the radius features of the hand palm by find the ratio between the radius and the width or height of hand.
  + Classify the finger lines according to the angle between the line and Ox to calculate features
  + Calculate the nine ratios between the number of black pixels of nine square areas below and the total black pixels of binary image containing hand palm.
  + Calculate the sixteen ratios between the number of black pixels of sixteen square areas below and the total black pixels of binary image containing finger lines.
  1. **Definitions in image processing**
     1. **Convex Hull**

****

**(Một người vẽ lại 1 hình khác hình này lấy của ngta)**

* + 1. **Convexity Defects**
  1. **Used Image Processing Algorithm**
     1. **Find the convex hull (Trên mạng nhìu thuật toán để tìm, lấy đại 1 ví dụ)**
     2. **Find the convexity defects (Cái này cũng vây)**
     3. **Calculate the finger line features**

We specify eight categories to maps these line into. These categories are angle between the line and the Ox axis such as 0 to 45 degree category, 46 to 90 degree, 91 to 135 degree, 136 to 180 degree, -45 to -1 degree, -90 to -46 degree, -135 to -91 degree, and -179 to -136 degree. The angle is calculated by

* 1. **Flow chart**

1. **Support Vector Machine**
   1. **Definition**

Support Vector Machine is a useful technique for data classification that analyzes data and recognizes patterns.

* 1. **Define problem**

Now we have sets of features of the hand signs and we must rely on those sets to recognize them. Therefore, we use library for Support Vector Machine (LIBSVM) to produce a model by basing on the training hand sign feature data and then rely on the model to predict the target values of the other hand sign features.

* 1. **Goal**

This library for Support Vector Machine algorithm builds a model that assigns new examples into specify category and when new hand sign are mapped into that same space and predicted to belong to a category.

* 1. **Support Vector Machine Algorithm**

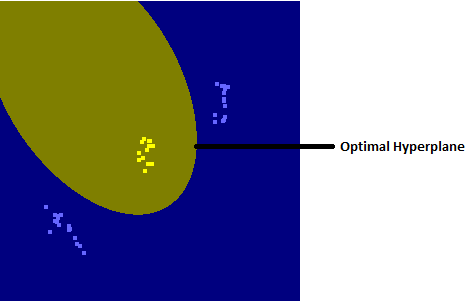
A Support Vector Machine is a discriminative classifier formally defined by constructing set of hyperplanes in a space has high dimension or infinite dimension. Furthermore, the algorithm should output a hyperplane which is optimal to categorize new examples. An optimal hyperplane is a hyperlane has the largest distance to the nearest training data point of classes.

For example:

* In the above picture, we see that there are a lot of lines can resolve the problem but which is better than others?
* The algorithm defines a criterion to estimate the worth of these lines that is the one that represents the largest separation between two set of points, so we choose the optimal separating hyperplane maximizes the distance from it to the nearest data point on each side.

The above example is one technique of Linear SVM. Besides that, there is another technique supported by SVM that is nonlinear classification. This is the way to create nonlinear hyperplanes by applying the kernel trick to margin hyperplanes.

We chose the nonlinear SVM technique because it can map samples into a higher dimensional space and the number of features and sample is quite small. The following image is an example of nonlinear SVM technique:



With multiclass, SVM algorithm aims a common method that is reducing the single multiclass problem into binary classifiers which separate between one class and the remains. That is called one-versus-all. The following example:

