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

Background subtraction is the way system keeps only the hand on images captured from camera, 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. **Solution**

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 background sample to grayscale.
  + Convert grayscale background sample to lab sample image.
  + Get all color pixels from lab sample image.
* Define color range of sample color:
* Check color pixels is between sample color range:
* Remove the color pixels is between color range:
  1. **Used Image Processing Algorithm**
     1. **Blur Image – Gaussian Blur**

Image filter have important implications in creating effects in the image, some effects using filters such as Blur, Smooth ...

The general rule of the method is a multiplying of image with a matrix filter,

*Idst= M\*Isrc*

Isrc, Idst is the original image and the image after performing with filtering by multiplying filter matrix M. Matrix Msometime also call mask, kernel. With each filter, we have allowed the different matrices M filter, no specific rules for determining M*,* however this matrix has some characteristics as follows:

- The size of the matrix is usually an odd number such as 3x3, 5x5 … Meanwhile, matrix center will be at the intersection of the two diagonals and is imposed on the image that should be multiplying.

- General elements in matrix common with 1. If this general bigger than 1, image through the filter will allow greater brightness than original image. By contrast image obtained is darker than original image.

Example about matrix filter (Matrix filter Sobel by x, y and matrix Gausian Blur)

Specific formula for the following photo filters :

Among them, we calculates multiplying to pixel coordinates *(x,y)* and because we take the center of the filter matrix is root so *u* run from *-n* (left point) and *v* run from *-n* (above side) to n, with *n = (size of mask - 1)/2*. For easier to understand, we consider an example of smoothed using a filter matrix as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 144 | 205 | 203 | 100 |
| 100 | 195 | 200 | 200 | 100 |
| 100 | 200 | 205 | 195 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 200 | 205 | 203 | 100  **=**  **pixel (2,2)** |
| 100 | 195 | 200 | 200 | 100 |
| 100 | 200 | 205 | 195 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9  \* |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 200 | 205 | 203 | 100  **=**  **pixel (2,3)** |
| 100 | 195 | 200 | 200 | 100 |
| 100 | 200 | 205 | 195 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 144 | 167 | 203 | 100 |
| 100 | 195 | 200 | 200 | 100 |
| 100 | 200 | 205 | 195 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9  \* |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

…..

And the final result we have :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 200 | 205 | 203 | 100 |
| 100 | 195 | 200 | 200 | 100  **=** |
| 100 | 200 | 205 | 195 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 100 | 100 | 100 | 100 |
| 100 | 144 | 167 | 145 | 100 |
| 100 | 167 | 200 | 168 | 100 |
| 100 | 144 | 166 | 144 | 100 |
| 100 | 100 | 100 | 100 | 100 |

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9  \* |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

As we see, original image with the image contrast is quite large (the pixel value difference magnitude greater: 100, 200, …), after filtering image contrast or simply faded away (this time the difference between the pixel value decreases : 100, 144, 167 …). Here we will look at some filters in OpenCV.



**Blurs an image using a Gaussian filter.**

void GaussianBlur(InputArray src, OutputArray dst,

Size ksize, double sigmaX, double sigmaY=0,

int borderType = BORDER\_DEFAULT )

**Parameters :**

*src* – input image; the image can have any number of channels, which are processed independently, but the depth should be CV\_8U, CV\_16U, CV\_16S, CV\_32F or CV\_64F.

***d****st* – output image of the same size and type as src.

*ksize* – Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero’s and then they are computed from sigma\* .

*sigmaX*– Gaussian kernel standard deviation in X direction.

*sigmaY*– Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed fromksize.width and ksize.height , respectively (see [getGaussianKernel()](http://docs.opencv.org/modules/imgproc/doc/filtering.html#Mat getGaussianKernel(int ksize, double sigma, int ktype)) for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.

*BorderType* – pixel extrapolation method (see [borderInterpolate()](http://docs.opencv.org/modules/imgproc/doc/filtering.html#int borderInterpolate(int p, int len, int borderType)) for details).

* + 1. **Convert BGR image to GrayScale image**
    2. **Convert GrayScale to LAB image**
  1. **Flow chart**

1. **Detecting Hand**
   1. **Definition**
   2. **Define Problem**
   3. **Solution**
   4. **Used Image Processing Algorithm**
      1. **Morphology Transformations – Opening**
      2. **Find Contour**
      3. **Convex Hull**
      4. **Convexity Defects**
   5. **Flow chart**
2. **Features Extraction**
   1. **Definition**
   2. **Define Problem**
   3. **Solution**
   4. **Used Image Processing Algorithm**
   5. **Flow chart**