Human Detection

By

Jong Park

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

June, 2011

Page

Abstract		4
I.	Introduction	5
II.	Related Works	6
III.	Requirement	9
IV.	Methods Overview	10
V.	Test Plans	13
VI.	Implementation	14
VII.	Test Results	22
VIII.	Conclusion	23
IX.	Bibliography	24

Appendices

A.	Matlab Code	.2	5
----	-------------	----	---

3

Table

1. Table 1 :		The result of the test	18

Figures7

1.	Figure 1 : Example of edge detection	7
2.	Figure 2 : Example for k means clustering	8
3.	Figure 3: K means clustering used on HSV image	8
4.	Figure 4: Procedures	9
5.	Figure 5 : RGB to HSV image	10
6.	Figure 6 : Hue component only image	11
7.	Figure 7 : Sample RGB image for the test	15
8.	Figure 8 : HSV image	16
9.	Figure 9 : Hue only image	17
10.	Figure 10 : Possible candidate of human selected	17
11.	. Figure 11 : Histogram of the vertical projection of hue component	
	image	18
12	. Figure 12 : Histogram of the horizontal projection of hue component	

image	18
13. Figure 13 : Sample image for the 2 nd test	19
14. Figure 14 : The image after selection procedure	20
15. Figure 15 : Saturation image	20
16. Figure 16 : Image with low saturation	21

Abstract

This report presents a method and test results for human detection in mountain area especially from long range. In many cases of human detection procedures involve shape recognition procedures using template images or edge detection. However, there is a limitation of using such methods in mountain area due to the fact other objects like trees and grass may cover most of body parts. My method involves HSV transformations and clustering based on local maxima which is faster and accurate for my goal

I. Introduction

My project was started based on the automated helicopter project for rescue mission to the place that cannot be easily reached. The subject of human detection is not something new. There are several different ways such as motions sensor, heat sensor, moving object sensor. Although, there isn't yet perfect method to find human in images because of fact about almost infinite cases of different shape, color and orientation and also possible mismatch with other objects, its use can be significant. For example, searching missing person in mountain area can be very helpful since several mini automated helicopters can be used to quickly respond to find person in urgent situations or can be used to acquire data for number of hikers for possible prevention of accidents in mountain area. For that reason and the interest of computer vision, I intend to find algorithm that suits several requirements which is discussed in page 8.

II. Related Works

There are several methods for human detection in image processing. One is edge detection



Figure 17 : Example of edge detection

This method would only work for simple image such as above image. The edge detection method would find many edges in mountain areas.

Another method would be clustering method such as k means. This method calculates the distance from clusters to each point and keeps tracking those points to group together to specified number of clusters. So if RGB image is inserted as a input, the algorithm will sort each points to specified number of clusters based RGB value of each point. The sample image is at the next page.

As you can see, K means clustering works pretty well if target object has clearly different RGB value. Unfortunately, when sample image was tested with this method, the result was unconvincing.



Figure 18 : Example for k means clustering

Figure 19: k means clustering used on HSV image

Right image is HSV format image. Red spots indicate the person. Based on the image, the person looks apparent, but after same function that used for figure 2 was used, the result provide the image that is not much useful. There are other methods such as a human face recognition technique using Eigen vectors. However, mostly it is tested in same background setting. Also, Histogram of oriented gradient descriptors is a popular choice for object detection. What it does it that it computes the distribution of intensity gradients or edge directions then separates human from the background. However, the images for my program consist person whose body parts can be covered by trees or grasses and many edges will be detect in mountain areas from rocks and trees which could make this method inaccurate.

III. Requirement

- The program must use relatively low resolution image
 - > Many cameras don't have high resolution features
 - High resolution image takes lots of memory
- The program must be fast enough to be used in flying objects
 - The program is intended to be used for the camera inside of flying object such as helicopter

IV. Methods Overview



2. Filter and Selection: as you can see figure 4 images, there must be a filtering

procedure in order to reduce noises then it will select high hue components out from the image with set or calculated threshold.



Figure 22 : Hue component only image

The above image show just hue component image. Brighter spots are correspondent to spots with higher hue values. Appropriate threshold must be set to select those spots that contain human but no other objects. And it must be the case that covers the most of picture taken at different setting with different people

3. Handling Noise/Error: the phase above may not remove all the possible spots that. So, during this phase, the program will determine if selected image is human or not. In most cases, the error comes from the part of the rocks that is shadowed by its shape.

4. Segmentation: once the part of the image is selected as a candidate for human, it must segment it in order to locate location in the images and number of people that are in the image.

Further details and codes will be shown in IV. Implementation section

V. Test Plans

Images sets for tests were not found online. So test will be conducted using pictures taken by myself at the bishop mountain in San Luis Obispo. And the code was developed as I keep testing with example images. The possible restriction was set in order to develop algorithm.

VI. Implementation



Figure 23 : Sample RGB image for the test

The above image will be run by my code. This image is taken at the bishop peak mountain. The resolution is 600 x 800 pixels. People are marked by the box for the convenience to locate them (above picture is not the result of my program.) It would help if the information about the distance from target objects from where the picture is taken and the possible pixel sizes of human is known. So, my program is designed with those variables that are controlled by the user. To begin, the first step is to transform this RGB image to HSV image. Since, I am using Matlab for writing the program, the built-in function output= rgb2hsv(input); is used.



Figure 24 : HSV image

The above picture is the result of rgb2hsv transform. Although, humans seem distinguishable in the eyes, running k means clustering would not work (page 6). And as you can see, there are tiny red spots spread throughout in the image. Those will be cleared by the median filter. The median filter takes the part of the matrix and changes the values of all elements to median values. And I chose the size of matrix to be [10 5] since people in the images are more likely to be vertically longer.



Figure 25 : Hue only image



Figure 26 : Possible candidate of human selected

Figure 9 shows filtered HSV image with only hue component. The brighter spot indicates a pixel with higher hue value. Figure 10 is created by a threshold value so those three bright spots are selected. In some cases, a threshold value will not separate human objects from other objects right away like the example above. And extra work has to be done which will be explained later. The next step is to separate each of pixels to form groups so number of people can be counted. And it will be done by histograms. First, figure 10 will be projected to vertical axis and horizontal axis



Figure 11 shows the histogram for a projection to vertical axis. X-axis represents row 1 to 600. I created the function find local maxima which isn't necessarily a peak in this. Because we don't want to find the peak since what we are interested in is a center location where pixels are closely grouped. The program will go from left to right searching for those local maxima. Figure 12 is same as above. But it is histogram of a projection to the horizontal line.



Projection of hue component image

So, with a list of data that are correspondent to white pixel in figure 10, the program will go through the loop, comparing each pixel location to the location of local maxima grouping together to three locations in the example. The result is in the table below.

Person #	Location(row)	Location(col)
1	309	418
2	309	535
3	34	662

Table 1: The result of the test

Each location in the table 1 corresponds to that of person's location in the original image. This example worked out great; however some examples are trickier. I mentioned that shadow in the rocks creates the pixel with high hue value which sometimes does not go away by median filtering and selection by threshold. And in order to remove a pixel, I will use saturation image. Figure 13 is an example image for this.



Figure 29 : Sample image for the 2nd test



Figure 30 : The image after section procedure

This is the result of phase 3. And there's a spot at the right upper corner which I do not want. And I am going to remove it by using the fact that the shadow has low saturation values.



Figure 31 : Saturation image

The image in figure 15 is an image with saturation component only. As you can see, upper right corner spot has low saturation values. And in order to distinguish low saturation component, I will use histogram equalization to enhance the image. After histogram equalization and separating the low saturation sections from the image, the result looks like figure 16.



Figure 7 : Image with low saturation

Then, the program will compare those spots in figure 14 to figure 16. The algorithm works like this; if a spot has a similar size for both high hue and low saturation values or has high values for both hue and saturation, they will be considered as a person because the image of the human can have both low and high saturation component. However, if a spot has high hue values but surrounding area has low saturation values. Then, that spot will be disregarded from the detection. My program asks users to put 4 inputs. One is input image which is RGB color form. One is threshold value that separates candidates for

possible humans and non-humans. In Most cases, number between $0.5 \sim .6$ detects most of humans except those who have low hue components than environment. Next inputs are min and max. Min is the smallest pixel size for human. It might best to set 0 if you are unsure. Max is the biggest pixel size for human. This information depends on how far the camera is. Knowing this information will increase the possibility of correct detection.

VII. Test Results

Total images	# of people	People found	percentage
35	47	32	68%

Table 2 : Test result

This program heavily depends on the input of users. In most cases, people will be found if they have high hue component which most people do. However, search can be limited if there's shadow throughout the mountain or too dark that colors changes too much. Most daytime would work fine. So far, this program would not be practical to be used yet. Also, better sample data would have made the code more accurate.

VIII. Conclusion

The test result is not much satisfying. Using HSV transformation to detect was easier approach. However, the variation of background setting made it difficult to isolate only humans. There are possible solutions such as implementing object recognition based on the sample image or template to detect objects or isolate objects. However, the speed of system could decrease due to complex procedures. Also, adding features in order to cover few problems that are only noticeable in few images can be wasting and also decrease the speed. In order to improve quality of the program, more and the better samples would be required. Since it is targeted to be used for pictures taken from aerial area, using samples that are collected in the mountain might not give accurate result.

IX. Bibliography

"Part-Based Shape Similarity Project." *Shape Similarity Project*. Department of Computer and Information Sciences. Web. 12 June 2011. http://knight.cis.temple.edu/~shape/partshape/overview/1.php>.

Turic, Hrvoje, Vladan Papic, and Hrvoje Dujmic. *A Procedure for Detection of Humans from Long Distance Images*. Tech. 50th International Symposium ELMAR-2008, 2008. Print.

Appendices

A. Matlab Code

```
function [ out ] = test( imset, threshold , min , max )
%this function find the person in rgb image
%rgb to hsv transformation
%imset : original input image
%imset.hsv : HSV image
imset.hsv = rgb2hsv(imset.orig);
%hue only image
%imset.hsv(:,:,1) : hue only image
%[10,5] : dimension of matrix for median filter
%imset.hue : filtered hue image
imset.hue = medfilt2(imset.hsv(:,:,1),[10,5]);
%saturation only image
%imset.hsv(:,:,2): saturation only image
%[10,10] : dimension of matrix for median filter
%imset.sat = filtered saturation image
imset.sat = medfilt2(imset.hsv(:,:,2),[10,10]);
%select only low saturation components
%imset.sat: saturation only image
%imset.sat2 : low saturation image
imset.sat2 = filtlsat(imset.sat);
%select candidate for possible humans
%imset.hue : hue only image
%threshold : value to differenciate possible human and
others
%imset.hue2 : images with possible candidates
imset.hue2 = thresh(imset.hue, threshold);
%segmentation
%imset.hue2
%imset.localmax : location for local maxima of possible
candidate
%imset.groups : all pixels location that belong to the
each local maxim
%found : 1 if local maxim found
[imset.localmax, imset.groups, found] =
detect(imset.hue2);
```

```
if(found)
   %remove posssible shadows
   %imset.localmax : location for local maxima of
possible candidate
   %imset.groups : all pixels location that belong to
the each local maxim
   %imset.final = final out
   imset.final =
hscompare(imset.sat2,imset.localmax,imset.groups,20);
else
   %nothing found
   imset.final = zeros(size(imset.hue));
end
%count the number of people found
[row,~] = find(imset.final>min&&imset.final<max);</pre>
npeople = length(row);
imset.num = npeople;
%send out to the oustide
out = imset;
end
```

```
function [ sat2 ] = filtlsat( sat )
%this function filter low saturation component using
histogram
%equalization
%sat : saturation only image
%sat2 : low saturation only image(1 for low saturation)
sat = histeq(sat);
sat2 = (sat<0.07);</pre>
```

```
end
```

```
function [ hue2 ] = thresh( hue, threshold )
%select possible human based on threshold
  [m,n] = size(hue);
  hue2 = double((hue>threshold));
  %if there are too many values with high, ignore
image
  if(nnz(hue2)>((m*n)/3))
     hue2(:,:) = 0;
  end
```

end

```
function [ localmax, groups , found] =
detect( img, local)
%segment pixels to group
%img : input img(hue2)
%local : choose to use peak or not
Slocalmax: array that has a value of 1 for local
maxima(cencentrated point)
%groups: cell that has array of pixels location for
each locam maxima
%found : if nothing found, pass 0
   %create array and cell to store location
information
   [m,n] = size(imq);
   localmax = zeros(m,n);
   groups = cell(m,n);
   [row, col] = find(img);
   %generate histograms of image projected to x-axis,
y-axis
   vp = sum(img, 2);
   hp = sum(img, 1);
   %in a case you would want to use peak value to find
local max
   try
       if(local == 'peak')
          %find locations of local maximum peaks
          [~,vlocs] = findpeaks(vp);
          [~,hlocs] = findpeaks(hp);
       end
   catch ME;
          %find locations of local max(concentrated
point)
          vlocs = locmax(vp);
          hlocs = locmax(hp);
   end
   %compare each point and put it to appropriate local
max
   for i=1:length(row)
          vdiff = abs(vlocs - row(i));
          hdiff = abs(hlocs - col(i));
          [~,vmap] = min(vdiff);
          [~, hmap] = min(hdiff);
          x = vlocs(vmap);
          y = hlocs(hmap);
          if(x \sim = 0 \& \& y \sim = 0)
              [1, ~] = size(groups{x, y});
              localmax(x,y) = localmax(x,y) +1;
              groups{x, y}(1+1, 1) = row(i);
```

```
groups{x,y}(l+1,2) = col(i);
found = 1;
else
found = 0;
end
end
```

end

```
function [ locs ] = locmax( input )
%Find the local max(highest points) by searching for
start point of
% positiple slope and the last point for the negative
slope
%input : a row represents the location and values
represent number of
%data in the location(projection)
%locs : locatinos of loca max
prev = input(1);
count = 1;
locs = zeros(1);
first=0;
for i=2:length(input)
   current = input(i);
   if(prev==0)
       if(current>0)
        first = i;
      end
   else
       if(current==0)
         last = i;
         locs(count) = round((first+last)/2);
         count = count+1;
      end
   end
   prev = current;
end
end
```

```
function [ final ] = hscompare( sat2, localmax, groups,
size )
%this functions compares hue and saturation value and
%produce the result based on the relationship
%if hue is high and saturation is high, the possibility
of
%human goes up. if hue is high and saturation is low,
check for surrounding
%then it makes the decision
%sat2 : filtere saturation image
% localmax : array for local max
% groups : cell that contains list of pixels in local
maxima
% size : size to compare hue and saturation
[m,n] = size(localmax);
[row, col] = find(localmax);
final = zeros(m,n);
%loop through each local max(group of pixels)
for i=1:length(row)
   %get a matrix size of hue component to check with
saturation component
   vc = row(i);
   hc = col(i);
   vmin = min(groups\{vc, hc\}(:, 1));
   vmax = max(groups\{vc, hc\}(:, 1));
   hmin = min(groups\{vc, hc\}(:, 2));
   hmax = max(groups\{vc, hc\}(:, 2));
   vdiff = vmax - vmin;
   hdiff = hmax - hmin;
   %if saturation component is high, decide it to be
human
   check = sat2(vmin:vmax,hmin:hmax);
   if(length(find(check)) < length(groups{vc,hc}));</pre>
   else
       try %if candidate is at the edge, ignore them
          check = sat2(vmin-size:vmax+size,hmin-
size:hmax+size);
          length(find(check));
      catch ME
          localmax(row(i), col(i)) = 0;
       end
      %check the surromding area for possible low sat
shadows
       if(length(find(check))>(numel(check)/4));
          localmax(row(i), col(i)) = 0;
      end
   end
```

```
final = localmax;
end
```