Abstract

Regional heavy rainfall is usually caused by extreme weather conditions. Consequently, heavy rainfall often results in the flooding of rivers and the neighboring low-lying areas, which is responsible for a large number of casualties and considerable property loss. The existing weather forecast systems focus primarily on the analysis and forecast of large-scale areas and do not provide precise instant automatic monitoring and alert feedback for small areas and sections. In our project, we propose a method to detect the flood and automatically monitor the probability of flood level of a specific area based on images using image processing methods to provide instant feedback on flooding events. OpenCV(Open Source Computer Vision) which specializes in real-time computer vision, is used for motion tracking and image processing. It is a library of programming functions mainly aimed at real-time computer-vision, originally developed by Intels research center Nizhny-Navgorod (Russia), later supported by Willow Garage and now maintained by Itseez. The library is cross-platform and free for use under the Open-source BSD license.The system can track objects and analyze if its flooded based on a set of features. The proposed method can better meet the practical needs of disaster analysis and management. The results offer prompt information and reports for appropriate disaster warning and response in disaster affected areas.

Contents

List of Figures

Chapter 1

Introduction

Images are the most common and convenient means of conveying information. A picture is worth a thousand words. Images concisely convey information about positions, sizes and inter-relationships between objects. They portray spatial information that we can recognize as objects. Human beings are good at deriving information from such images, because of our innate visual and mental abilities.

A digital remotely sensed image is typically composed of picture elements (pixels) located at the intersection of each row i and column j in each K bands of imagery. A smaller number indicates low average radiance from the area and the high number is an indicator of high radiant properties of the area. The size of this area effects the reproduction of details within the scene. As pixel size is reduced more scene detail is presented in digital representation. An image is nothing more than a two dimensional signal. It is defined by the mathematical function f(x,y) where x and y are the two co-ordinates horizontally and vertically. The value of f(x,y) at any point is gives the pixel value at that point of an image.

Figure 1.1: Example of an image

The above figure is an example of digital image. But actually, this image is nothing but a two dimensional array of numbers ranging between 0 and 255.

128	30	123
232	123	321
123	77	89
80	255	255

Figure 1.2: Array of numbers between ranging 0 and 255

Each number represents the value of the function f(x,y) at any point. In this case the value 128 , 230 ,123 each represents an individual pixel value. The dimensions of the picture is actually the dimensions of this two dimensional array.

Most of the existing research work is based on finding whether a given video of the disaster affected area is a flood. Our project is related to the images, given an image, the device should be able to detect whether its a flood affected area or not using principles of digital image processing. It estimates the probability of flood from crowd-sourced mobile images of the disaster affected area using fuzzy based semantic scene analysis.

Digital image processing (DIP) deals with manipulation of digital images through a digital computer. It is a sub-field of signals and systems but focuses particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output.

Figure 1.3: Working

In the above figure, an image has been captured by a camera and has been sent to a digital system to remove all the other details, and just focus on the water drop by zooming it in such a way that the quality of the image remains the same.

Most methods for detecting the flood affected area are trained by providing annotated sample images. Here we are giving a set of trained images to the device to detect the flood scene. However, conditions change since training and deployment can be in different locations with widely varying illumination, camera position, apparent object sizes. The generalization ability is compromised to the presence of such changes. Our project is to detect the flood and estimate the extent of flood from a crowd-sourced mobile images of the disaster affected area using fuzzy based semantic scene analysis.

Chapter 2

Problem Definition

Detect flood scenes and estimate the probability of flood from crowd-sourced mobile images from disaster affected areas, using fuzzy based semantic scene analysis

There are many challenges to be faced in detecting the flood scenes and estimating its probability. It is an unconstrained environment where the conditions are not restricted. There are many possible objects in the scene like water, human beings, vehicles, animals, buildings, trees and many more to recognize. The major objects to be detected to confirm if it is flooded or not are the human beings and the flooded water. In our project we have detected those objects which play a major role in the flood scene.

The development of face recognition methods for unconstrained environments is a challenging problem. The aim of this work is to carry out face recognition methods that are suitable to work properly in these environments.

Scene content understanding facilitates a large number of applications, ranging from content-based image retrieval to other multimedia applications. Material detection refers to the problem of identifying key semantic material types (such as sky, grass, foliage, water, snow) in images. In this project, we present an approach to determine scene content, based on a set of individual material detection algorithms, as well as probabilistic spatial context models. A major limitation of individual material detectors is the significant number of misclassifications that occur because of the similarities in color and texture characteristics of various material types.

It is an unconstrained outdoor environment where there are many illuminations conditions. There can be illumination of the water, reflection of

the buildings, trees, vehicles or human beings in water. There can be scenes where water is overflowed into the house because of leakage of the tap. There can be few objects floating in the flooded water which needs to be taken into consideration. There can be objects immersed in water like vehicles. There can be conditions where half of the vehicle is immersed in the water which makes the recognition very challenging.

We need to consider a huge set of training images to get a clear idea of the conditions. There might not be all conditions in a single image, so we need a large training set. But there is a possibility of misclassifications due to this.

A rule set needs to be generated for the fuzzy input. We need to brainstorm the rules in the flood scene. A vehicle or building immersed in water indicates possibility of flood. That can be taken into a rule set. Brownish color water indicates the flood situation. To find out the color of water we need to consider many conditions like illumination correction, gamma correction, equalization and thresholding. A lot of preprocessing is to be done which is a great challenge. So, there are many challenges to be faced to overcome and come out with a proposed mechanism.

Chapter 3

Related Work

A number of approaches have been proposed to detect the human faces. We have detected human beings and the flooded water based on these methods.

"Probabilistic Spatial Context Models for Scene Content Understanding", in this the authors mentioned about the content of the scene. Scene content understanding facilitates a large number of applications, ranging from content based image retrieval to other multimedia applications. Material detection refers to the problem of identifying key semantic material types (such as sky, human beings, buildings, vehicles, water, and trees) in images. This paper states that there are atleast two types of spatial contextual relationships are present in natural images. First, relationships exist between co-occurrence of certain materials in natural images; for example, detection of grass with high probability would imply low snow probability. Second, relationships exist between spatial locations of materials in an image; sky tends to occur above grass, foliage above grass, sky above snow, etc. They have developed a spatial context-aware approach to material classification that uses spatial context constraints to increase the accuracy of the initial classification by constraining the beliefs to conform to the spatial context models. Experimental results show that the spatial context-aware models improve the accuracy of an orientation aware material detection system by 13 percentile.The work concerns setting up context for the detection of one particular object and the control flow is one-way and sequential. So far, they have explored the use of basic spatial context models in an orientation aware materials detection system. Further work is needed in extending these models.

"A Probabilistic Model for Flood Detection in Video Sequences", in this the author proposed a new image event detection method for identifying flood in

videos. Traditional image based flood detection is often used in remote sensing and satellite imaging applications. In contrast, the proposed method is applied for retrieval of flood catastrophes in newscast content, which present great variation in flood and background characteristics, depending on the video instance. Different flood regions in different images share some common features which are reasonably invariant to lightness, camera angle or background scene. These features are texture, relation among color channels and saturation characteristics. The method analyses the frame-to-frame change in these features and the results are combined according to the Bayes classifier to achieve a decision (i.e. flood happens, flood does not happen). In addition, because the flooded region is usually located around the lower and middle parts of an image, a model for the probability of occurrence of flood as a function of the vertical position is proposed, significantly improving the classification performance. Experiments illustrated the applicability of the method has improved the performance in comparison to other techniques. The authors have exploited important visual features of flood not previously discussed in the literature, using a probabilistic model for the position of the flood region in images. We combined color, contrast and entropy, along with their dynamic change from frame to frame. In contrast to other methods which extract complicated features, the features discussed here allow very fast processing, making the system applicable not only for real time flood detection, but also for video retrieval in news contents, which require faster than real-time analysis. The experiments illustrate the applicability of the method, with an average false-negative rate of 0.46 percentile and a false-positive rate of 0.9 percentile.

"Multi-Modal Change Detection, Application to the Detection of Flooded Areas", in this the authors present algorithms, investigating both supervised and unsupervised methods and the use of multi-modal data for flood detection. Interestingly, a simple unsupervised change detection method provided similar accuracy as supervised approaches, and a digital elevation model-based predictive method yielded a comparable projected change detection map without using post-event data. The relatively low accuracy may be due to the presence of pixel level region mixing issue, in which an isolated patch of dry area surrounded by water may be considered as flood by a human operator that defines the extension of the flooded area. Further, the flooded area changed significantly between the two sensor observations(about 4 days apart). This may have created fundamentally different information, increasing the difficulty to dea lwith the temporal shift between the optical and SAR data sets. Failure or success of a change detection algorithm cannot be analyzed only with a confusion matrix, as it is important to understand the

context of the application. For example, missed alarms may be more important than a false alarm in catastrophic scenarios as it is better to check a non-destroyed building than not to visit a destroyed one. These conclusions suggest that research should be directed to investigating new and more powerful input features (as an example, multi-temporal as well as multi-angular information can be exploited) to be fed into the various machine learning schemes, or to a better understanding of the physical behavior of the Earth surface being investigated.

"A Factor Graph Framework for Semantic Video Indexing", in this the author talks about the Video query by semantic keywords as one of the most challenging research issues in video data management. They have presented a framework to obtain models for various objects sites and events in audio and video. They have presented a factor graph and described how it is automatically learnt. This has led to further performance improvement. They have proposed and demonstrated an open ended and flexible architecture for semantic video indexing. In addition to the novel probabilistic framework for semantic indexing, they have also used an objective quantitative evaluation strategy in the form of ROC curves and have demonstrated the superior detection performance of the proposed scheme using these curves. This will lead to an improvement in the baseline performance,as well as system performance. Spatial layout is one aspect that needs to be modeled together with co-occurrence for better modeling of context.

"Image Classification based on Fuzzy Logic", here the author mentions that the major advantage of this theory is that it allows the natural description, in linguistic terms, of problems that should be solved rather than in terms of relationships between precise numerical values. Although the fuzzy logic is relatively young theory, the areas of applications are very wide: process control, management and decision making, operations research, economies and here the most important, pattern recognition and classification. Mathematical function which defines the degree of an element's membership in a fuzzy set is called membership function. The natural description of problems, in linguistic terms, rather than in terms of relationships between precise numerical values is the major advantage of this theory. An idea to solve the problem of image classification in fuzzy logic manner as well as comparison of the results of supervised and fuzzy classification was the main motivation of this work.

Chapter 4

Proposed System

4.1 Overview

Our proposed method is to detect the flood scenes and estimate the probability of flood from crowd-sourced mobile images from disaster affected areas. We use the fuzzy logic based on semantic scene analysis to implement this method.

Firstly, the flood images are fed to the Intelligent Semantic Scene Understanding system. A decision is made by the fuzzy logic system taking the generated rule set into consideration. The probability of flood and the extent of flood is detected.

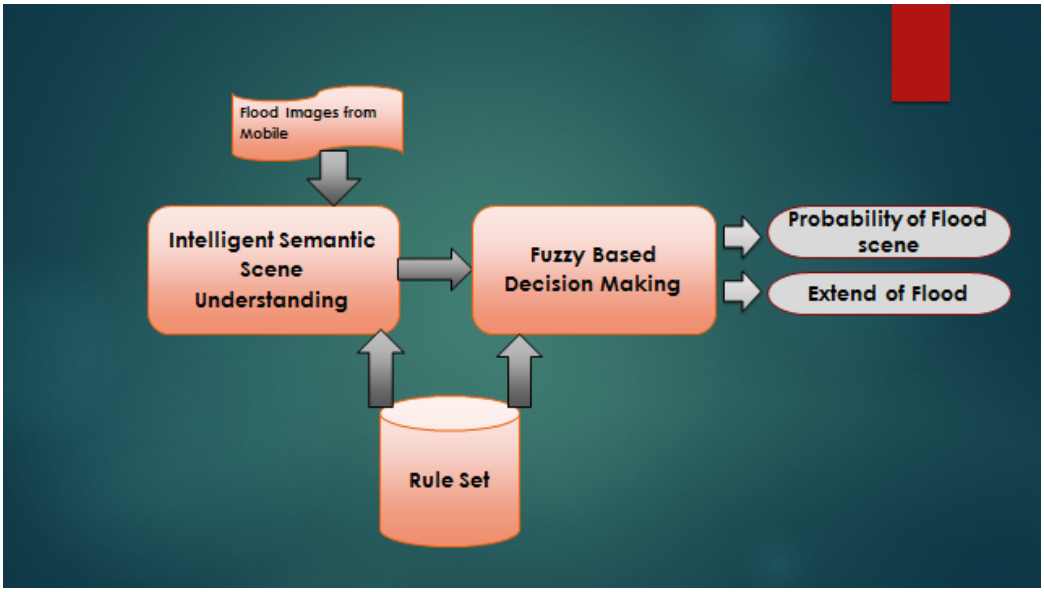


Figure 4.1: Solution Approach

A set of inputs are considered to feed the system. The water is to be segmented out, the humans are to be detected using intelligent semantic scene analysis. The level of water is to be found out and the extent of flood is to be calculated using the fuzzy base decision making.

4.2 Intelligent Semantic Scene Analysis

We have considered a set of images as the input to the system. Few images are shown below in figure 4.2.

Figure 4.2: Input images

The Semantic Scene analysis consists of various steps like preprocessing, segmentation of objects, extraction of features, segmented scene classification, context aware semantic scene analysis and the generation of rule set.

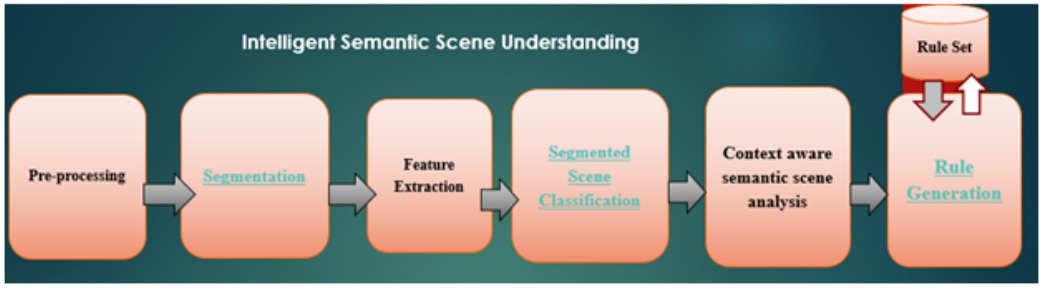


Figure 4.3: Intelligent semantic scene analysis

4.3 Preprocessing

The first step is the preprocessing step wherein illumination of water is corrected using HSV(Hue, Saturation, Value) color space conversion. Performing this removes uneven illumination of the image caused by sensor defaults, non-uniform illumination of the scene or orientation of the objects. The correction is based on background subtraction. This type of correction assumes the scene is composed of a homogeneous background and relatively small objects brighter or darker than the background. Equalization is done for the image. The contrast of the image is adjusted using histogram equalization.

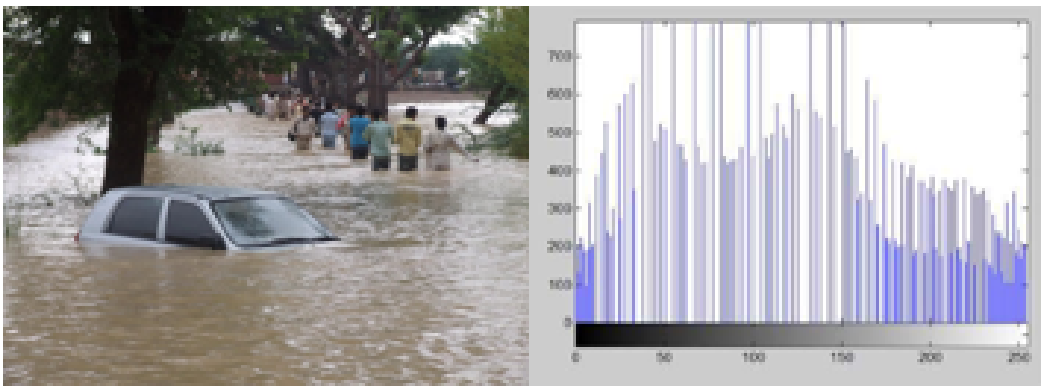


Figure 4.4: Histogram graph

Figure 4.5: Histogram Equalization

Gamma correction is applied to the images. This helps us in encoding or decoding the luminance in the image. A gamma value g¡1 is called encoding gamma where is compresses the power of the image, conversely gamma value g¿1 is called decoding gamma which is the expansive power of the image. Here we use the expansive gamma(g¿1) to enhance the power of the image.

Figure 4.6: Gamma correction with g=5

Figure 4.7: Gamma correction with g=3

4.4 Segmentation

We segmented out the objects in the scene using color based segmentation. All the objects for the flooded water is segmented out by masking the objects. Masking is nothing but filtering which convolves the mask with the image. We move the mask from point to point on the image to segment out the objects.

Figure 4.8: Masked image

4.5 Feature extraction and segmented scene classification

The extraction of features in the scene are based on color, boundary and texture. The human faces are detected using the Viola-Jones algorithm which works on Haar-classifiers. We have tried this on a set of input images which is shown below.

Figure 4.9: Face detection

Now the coordinates of the face are extended to find the human body. Few images are shown below.

Figure 4.10: Human body detection

Now, the water is segmented considering the color-segmentation technique. By thresholding the histogram equalized images, the flooded water is segmented out.

Figure 4.11: Flooded water

4.6 Scene analysis

The images in which the human-beings (face and body) are detected, are cropped out and the contour is found. Then flood-fill function is applied to that particular image so that the background is black and the human is white in color. Now the Zhang-Suen thinning algorithm is applied to skeletonize the flood-filled image assuming we can find out the length of the human immersed in flooded water. But due to many objects in the scene, the skeletonizing did not give an appreciable output.

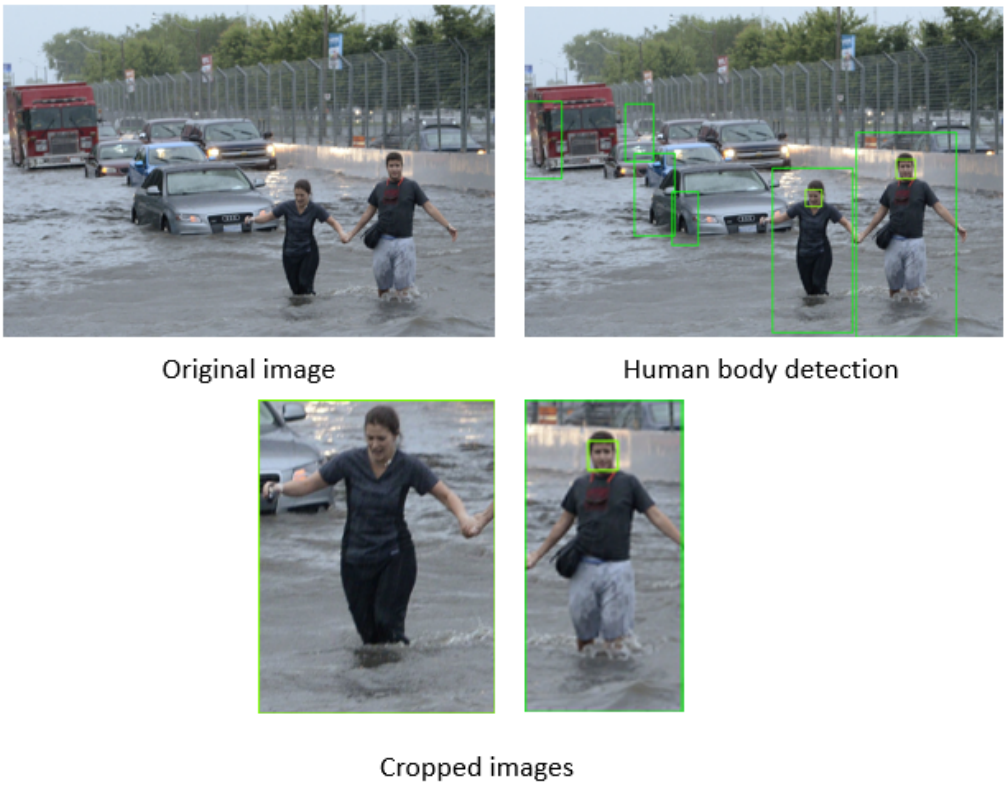


Figure 4.12: Cropping

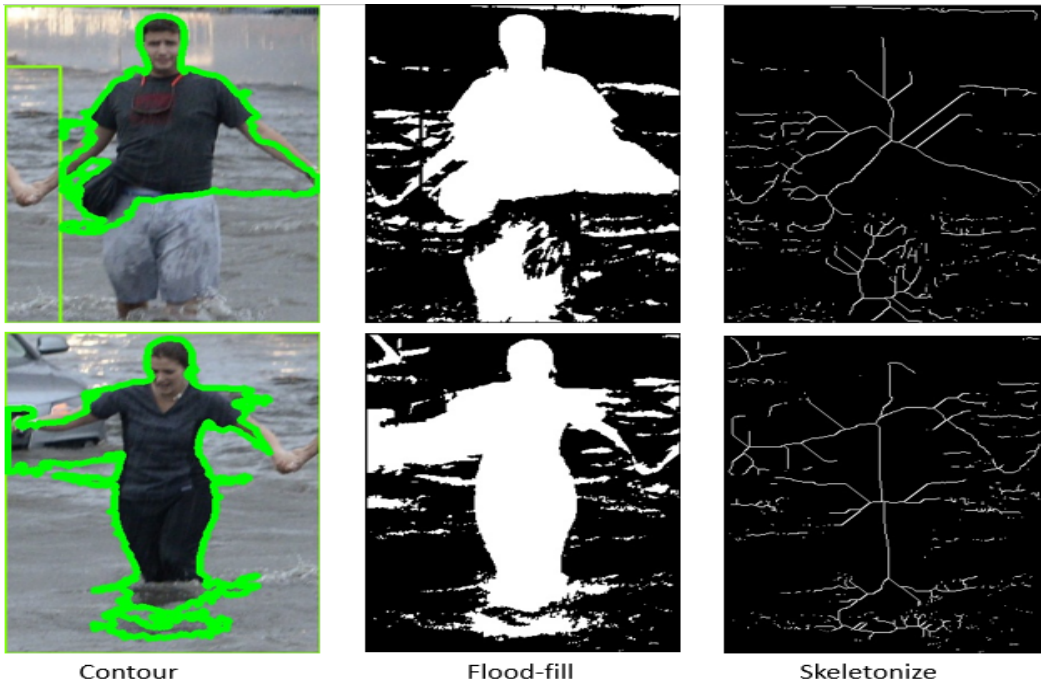


Figure 4.13: Contour, flood-fill and skeletonize

Since, skeletonizing the image did not fetch us an appropriate output, we found out the depth of the human immersed in flooded water. To do the same, the faces are detected and the bounding box for face and body is found out. The human body is segmented into eight boxes assuming the sizes of each box as the first top-most (face) box. It is assumed that the first box represent face, the succeeding three boxes represent the torso part, the next two boxes refer till knee and the final two boxes refer till the feet. The mean of color for each box is found out respectively and the difference of the same between each consequent boxes id found out. The highest mean color difference value is taken into consideration. This represents that there is a drastic color difference between that respective box and the preceding box, which means the flooded water is till the end of the preceding box. Finally, the depth of the human immersed in flooded water is found out.

Figure 4.14: Detection of Human Depth

Flooded water is segmented by finding out the contour of the color-segmented image.

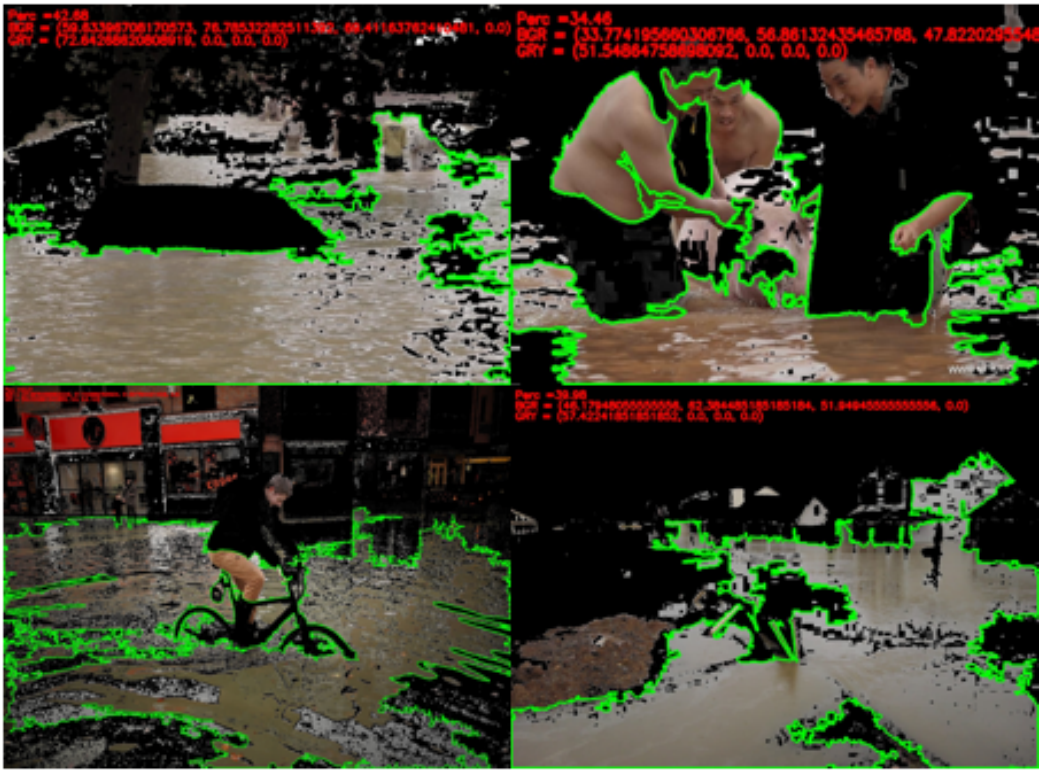


Figure 4.15: Largest contour of flooded water

Thus, the flooded water is segmented and the human are detected including the depth till which the human is immersed in flooded water.

4.7 Knowledge base

A set of rules are generated to feed them as input to fuzzy based decision making logic. These rules are based on the flood scenes. For example, consider the image shown below.

Figure 4.16: Rule set generation

Based on the inferences from the above image, the following rules can be generated :

- The vehicle is half immersed in water
- The water is brown in color
- Human beings are drowned in water till knee depth
- Half of the tree trunk is immersed in flooded water
- There is no rain

Likewise, rules are generated and are given as an input to the fuzzy logic. The output from semantic scene analysis is shown below.

Figure 4.17: Output from semantic scene analysis

Chapter 5

Testing and result analysis

The analysis is done by the fuzzy logic. In our system, the fuzzy logic gives the probability of the flood. Dealing with simple black and white answers is no longer satisfactory enough. It solves problems using the degree of membership.

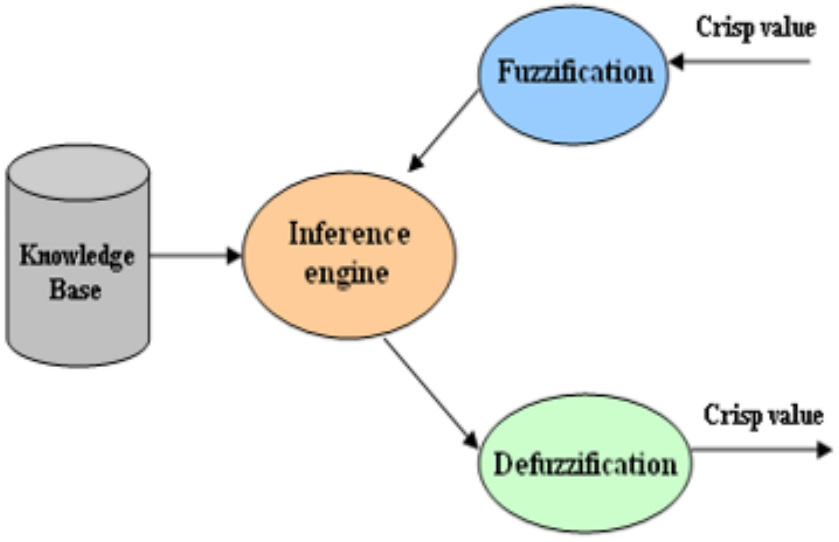
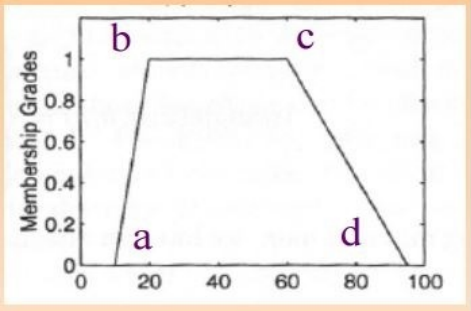


Figure 5.1: Structure of fuzzy logic

A fuzzy set is a set whose elements have degrees of membership. An element of a fuzzy set can be full member (100 percentile membership) or a partial member (between 0 percentile and 100 percentile membership). That is, the membership value assigned to an element is no longer restricted to just

two values, but can be 0, 1 or any value in-between. Mathematical function which defines the degree of an element's membership in a fuzzy set is called membership function. It is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. Here, we use the trapezoidal member function. The trapezoidal member function is specified by four parameters {a,b,c,d} as follows:



$$trapezoidal(x; a, b, c, d) = \begin{cases} 0, & x < a \\ (x - a)/(b - a), & a \le x < b \\ 1, & b \le x < c \\ (d - x)/(d - c), & c \le x \le d \\ 0, & x \ge d \end{cases}$$

Figure 5.2: Trapezoidal member function and equation

From Feature Extraction, the contour with largest area is calculated and the average color of the same contour is retrieved. Scene Analysis gives the depth till which the human is immersed in flood water, in percentage. These are the input to fuzzy system, which will be converted into fuzzy universe variables namely, color, area and depth respectively and the output being the degree of flood.
The universe variables are split into three parts representing the corresponding low, medium and high states of the respective universe:
color : light, medium, dark
area : small, medium, large
depth : till_knee, till_waist, till_neck
flood : low, medium, high
The antecedent describes to what degree the rule applies, while the conclusion assigns a fuzzy function to each of one or more output variables.The set of rules in a fuzzy expert system is known as knowledge base. We have generated a knowledge base with 10 rules. They are :

- If the depth is tillneck AND the color is medium AND the area is small, then the flood will be medium.
- If the depth is tillneck AND the color is light, then the flood will be medium.
- If the depth is tillneck, then the flood will be high.

- If the depth is tillwaist AND the color is dark AND the area is large, then the flood will be high.
- If the depth is tillwaist AND the color is light AND the area is small, then the flood will be low.
- If the depth is tillneck, then the flood will be medium.
- If the depth is tillknee AND the color is dark AND the area is small, then the flood will be low.
- If the depth is tillknee AND the color is medium AND the area is large, then the flood will be medium.
- If the depth is tillknee AND color is dark, then the flood will be high.
- If the depth is tillknee, then the flood will be low.

In the process of fuzzification, membership functions defined on input variables are applied to their actual values so that the degree of truth for each rule premise can be determined. Here, the inputs to the fuzzy based decision making are color, area and depth.

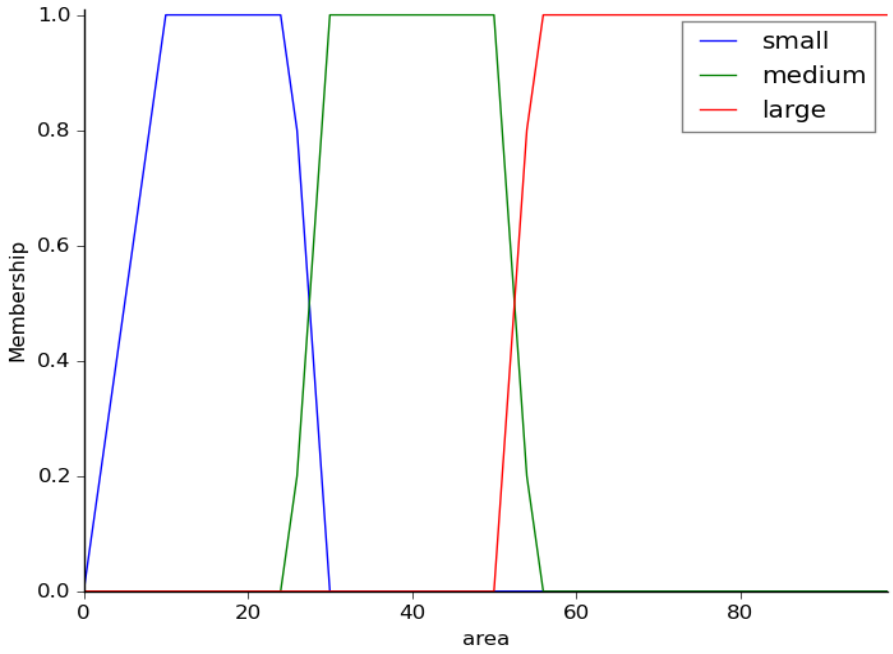


Figure 5.3: Member function with area as input

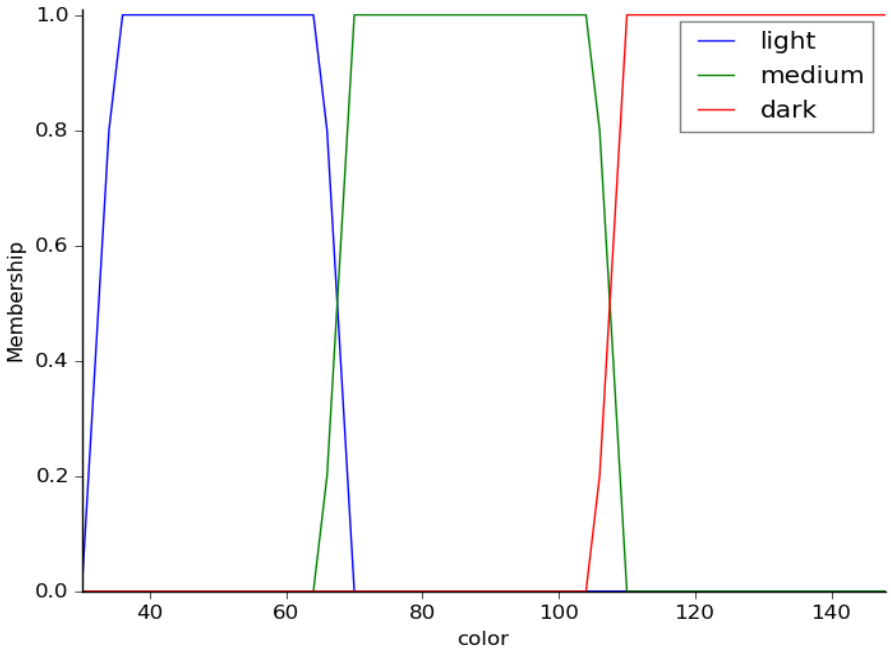


Figure 5.4: Member function with color as input

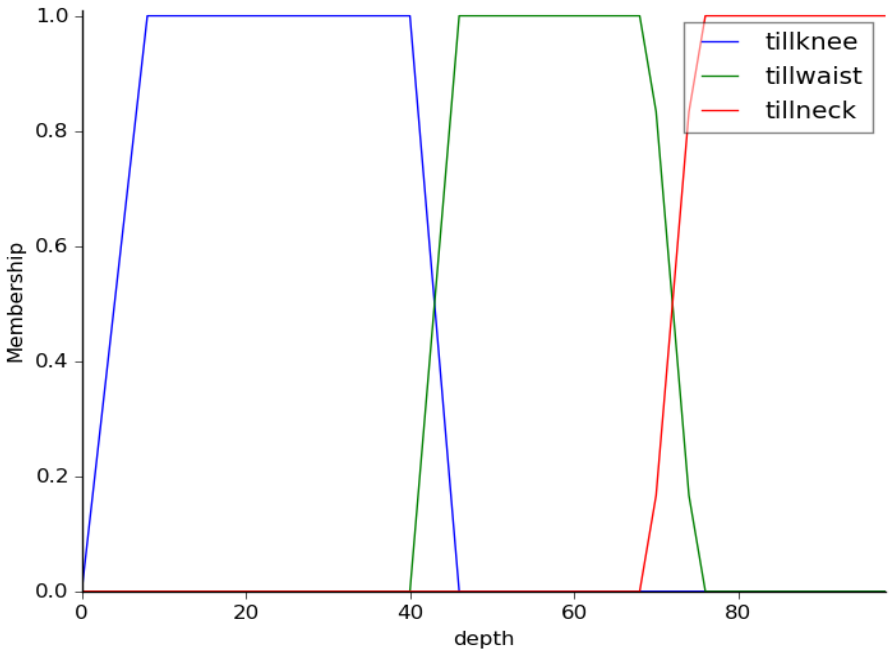


Figure 5.5: Member function with depth as input

In the process of inference, truth value for the premise of each rule is

computed and applied to the conclusion part of each rule. This results in one fuzzy set to be assigned to each output variable for each rule. The use of degree of support for the entire rule is to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. The consequent specifies a fuzzy set to be assigned to the output.

In Defuzzification, the fuzzy output set is converted to a crisp number, given fuzzy sets and corresponding membership degrees. These will have a number of rules that transform a number of variables into a fuzzy result, that is, the result here is the probability of flood with a trapezoidal function as shown in figure below.

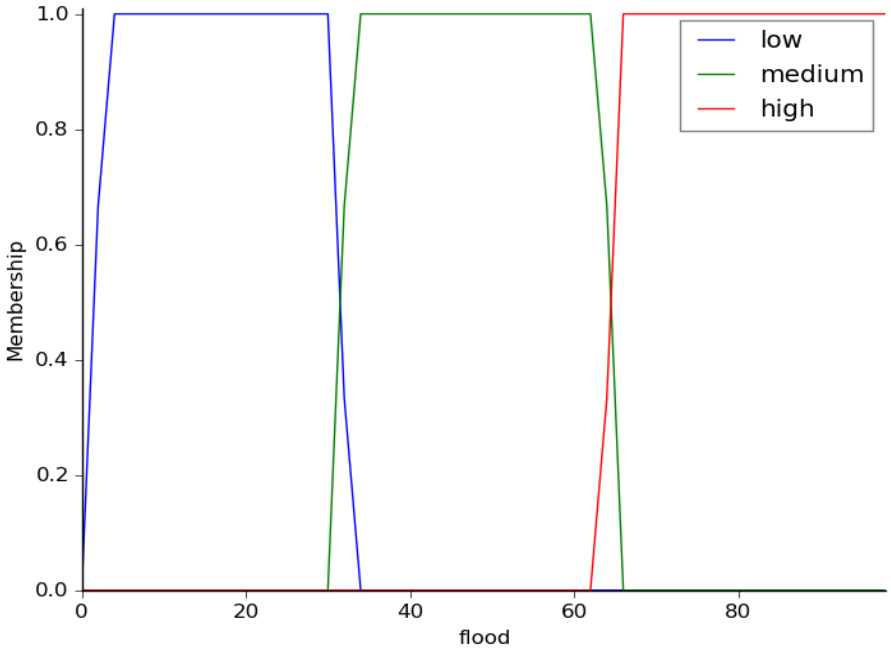


Figure 5.6: Probability of flood

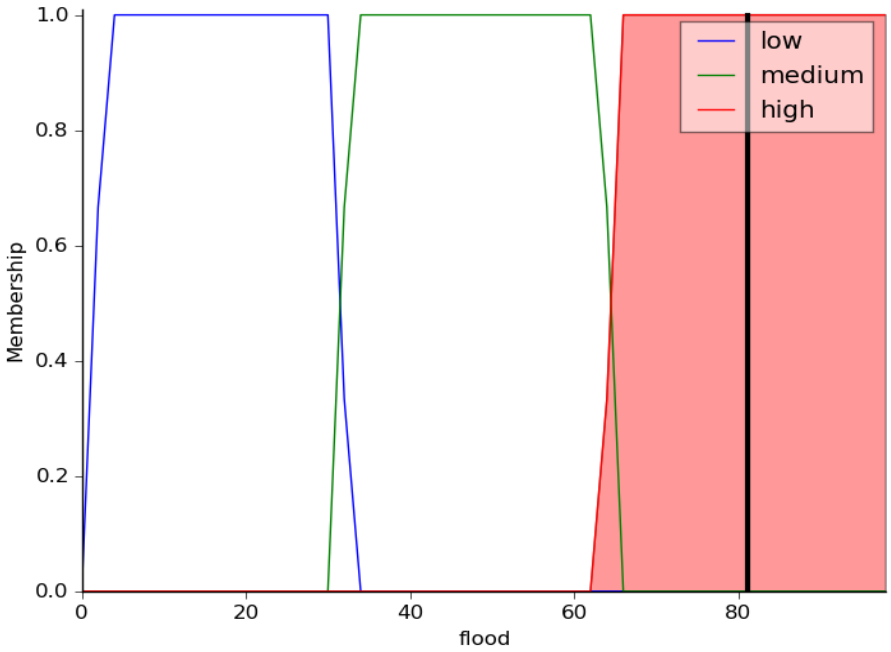


Figure 5.7: Output of flood with crisp value

Chapter 6

Conclusion

The Gamma correction is carried out for the original input image and the flooded water is segmented. The largest contour for the flooded water is found out. The human-beings are detected and boxes are drawn in particular proportions as mentioned. The average color for each box is determined and the difference is calculated. The box which has the highest mean color difference value is considered which gives the depth of the human body immersed in water. Thus, the probability of flood is detected using fuzzy inference system which takes color, area and depth as input and the trapezoidal function as membership function considering the depth of human beings immersed as the rule set or knowledge base. Thus, the probability of flood is detected using fuzzy based semantic scene analysis.
The current algorithm has been tested on a sample set of 200 images with 25 images that does not contain flood. Out of the 175 flood images the system gave positive result for 130 images and 7 out of 25 non-flooded images were also detected as flood.
False positive : 28%
False negative : 25%
Sensitivity : 74.28%

Image No.	Contour Area (%)	Average Color (30 - 120)	Depth (%)	Flood (%)
1	42.19	44.52	0.00	16.90
6	15.20	35.00	87.50	81.14
7	42.00	60.00	50.00	48.00
10	28.70	85.70	0.00	16.90
18	10.00	34.00	37.50	16.66
14	25.00	51.00	0.00	14.50
20	8.00	31.00	80.00	81.14
26	8.00	35.00	62.50	80.40
28	2.00	15.00	75.00	81.08
33	16.00	56.00	37.50	16.66

Figure 6.1: Output of flood with crisp value

References

[1] Amit Singhal, Jiebo Luo, Weiyu Zhu, "Probabilistic Spatial Context Models for Scene Content Understanding".

[2] M.Naphade and T.S.Huan, "A factor graph framework for semantic indexing and retrieval in video".

[3] E.Saber, A.M.Tekalp, R.Eschbach, and K. Kno, "Automatic image annotation using adaptive color classification".

[4] Paulo Vinicius, Koerich Borges, Joceli Mayer, Ebroul Izquierdo, "A probabilistic model for flood detection in video sequences".

[5] Y. Wang, Z. Liu, and J. C. Huang, Multimedia content analysis using both audio and visual clues, IEEE Signal Processing Magazine.

[6] C. L. Lai, J. C. Yang, and Y. H. Chen, A real time video processing based surveillance system for ood detection".

[7] S.G.Narasimhanand S.K.Nayar, Interactive segmentation of materials in an image using physical models".

[8] Nathan Longbotham, Fabio Pacici, Taylor Glenn, "Multi-Modal Change Detection, Application to the Detection of Flooded Areas".

[9] Ramesh Bharath, Lim Zhi Jian Nicholas and Xiang Cheng, "Scalable Scene Understanding Using Saliency-Guided Object Localization".

[10] P. Schyns and A. Oliva, From blobs to boundary edges: Evidence for time-and spatial-scale-dependent scene recognition.

[11] A. Oliva and A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope".

[12] Milind Ramesh Naphade, Igor V. Kozintsev, and Thomas S. Huang, "A Factor Graph Framework for Semantic Video Indexing".

[13] D. Zhong and S. F. Chang, Spatio-temporal video search using the object-based video representation.

[14] Y. Deng and B. S. Manjunath, Content based search of video using color, texture and motion".

[15] I. Nedeljkovic, "Image classification based on fuzzy logic".

[16] Terano T., Asai K., Sugeno M., "Fuzzy systems theory and its applications".

Appendix A

Appendix

The code is attached here :

```
# import the necessary packages
#from __future__ import print_function
import numpy as np
import argparse
import cv2
import time

# load the original image
original = cv2.imread('7.jpg') #args["image"])
cv2.imshow("Original image",original)


########### Gamma Correction ####################


def adjust_gamma(image, gamma=1.0):
        # build a lookup table mapping the pixel values [0, 255]
        #  to their adjusted gamma values
        invGamma = 1.0 / gamma
        table = np.array([((i / 255.0) ** invGamma) * 255
                for i in np.arange(0, 256)]).astype("uint8")

        # apply gamma correction using the lookup table
        return cv2.LUT(image, table)


gamma = 0.5
```

```
gamma_result = adjust_gamma(original,gamma)
cv2.imshow("Gamma Correction",gamma_result)

########### END OF GAMMA CORRECTION ##############

########### HISTOGRAM EQUALISATION ###############
'''
hist_equalized = cv2.cvtColor(gamma_result,cv2.COLOR_BGR2YCrCb)
y,cr,cb = cv2.split(hist_equalized)
equ = cv2.equalizeHist(y)
img = cv2.merge((equ,cr,cb))
hist_equalized = cv2.cvtColor(img,cv2.COLOR_YCrCb2BGR)
cv2.imshow('equalized',hist_equalized)
cv2.waitKey(0)
'''
########## END of Histogram Equalisation #########


########## Color Segmentation ####################
hsv = cv2.cvtColor(gamma_result, cv2.COLOR_BGR2HSV)
lower_brown = np.array([0,0,0])
upper_brown = np.array([30,255,255])
mask_segmentation = cv2.inRange(hsv, lower_brown, upper_brown)
segmented_image = cv2.bitwise_and(gamma_result,gamma_result,
mask= mask_segmentation)
cv2.imshow('Image Segmentation',segmented_image)
#cv2.waitKey(0)
########## END of Color Segmentation #############

########## Contour Area and Color ################

im2 = segmented_image #Copy
abc = np.zeros(segmented_image.shape, np.uint8)
height, width, channels = segmented_image.shape
size = height * width
mask_intensity = np.zeros(segmented_image.shape, np.uint8)

imgray = cv2.cvtColor(segmented_image,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray, 127,255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)
im2,contours, hierarchy = cv2.findContours(thresh,
```

```
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

num = 0
largest_area = 0
count = 0

for cnt in contours:
        cv2.drawContours(im2,[cnt],-1,(0,255,0), 3)
        #cv2.imshow('trial',abc)
        area = cv2.contourArea(cnt)

        if area > largest_area:
                largest_area = area
                count  = num
        num = num + 1

amount =  (largest_area / size) * 100
round_per = float("{0:.2f}".format(amount))

print "Largest area   = " + str(largest_area)
print "Contour Number = " + str(count)
print "Size = " + str(size)
print "Percentage of flood water = " + str(amount)
cnt = contours[count]
cv2.drawContours(segmented_image, [cnt], 0, (0,255,0), 3)

############## Code for Intensity ##############

result = cv2.bitwise_or(segmented_image,
mask_intensity, mask_intensity)
# cv2.imshow('mask', mask)
mean_val1 = cv2.mean(mask_intensity, mask = None)
print "Average  BGR = " + str(mean_val1)
gray_seg_image = cv2.cvtColor(mask_intensity, cv2.COLOR_BGR2GRAY)
mean_val2 = cv2.mean(gray_seg_image, mask = None)
p,q,r,s = mean_val2
# 4 values unpacked by the Color Space
# p=graysacale intensity
print "Average Gray Intensity = " + str(p)
cv2.imshow("Largest Contour",result)
#cv2.waitKey(0)
```

```
############ End of Intensity ################

############ Code for Depth ##################

face_cascade1 = cv2.CascadeClassifier('/home/wanderer/OpenCV/
opencv-3.1.0/data/haarcascades/haarcascade_frontalface_alt.xml')

face_img = original

f=0
avg = []
faces = face_cascade1.detectMultiScale(face_img,scaleFactor=1.025,
minNeighbors=3,minSize=(30, 30),flags = cv2.CASCADE_SCALE_IMAGE)
for (x,y,w,h) in faces:
        cv2.putText(face_img, "{}".format(f), (x, y),
        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 3)
        cv2.rectangle(face_img,(x,y),(x+w,y+h),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+h),(x+w,y+(2*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(2*h)),(x+w,y+(3*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(3*h)),(x+w,y+(4*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(4*h)),(x+w,y+(5*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(5*h)),(x+w,y+(6*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(6*h)),(x+w,y+(7*h)),(255,0,0),2)
        cv2.rectangle(face_img,(x,y+(7*h)),(x+w,y+(8*h)),(255,0,0),2)
        rect1 = face_img[y:y+h, x:x+w]
        rect2 = face_img[y+h:y+(2*h), x:x+w]
        rect3 = face_img[y+(2*h):y++(3*h), x:x+w]
        rect4 = face_img[y+(3*h):y++(4*h), x:x+w]
        rect5 = face_img[y+(4*h):y++(5*h), x:x+w]
        rect6 = face_img[y+(5*h):y++(6*h), x:x+w]
        rect7 = face_img[y+(6*h):y++(7*h), x:x+w]
        rect8 = face_img[y+(7*h):y++(8*h), x:x+w]
        #print "Mean = " + str(rect1.mean())
        rect = []
        rect.append(rect1.mean())
        rect.append(rect2.mean())
        rect.append(rect3.mean())
        rect.append(rect4.mean())
        rect.append(rect5.mean())
        rect.append(rect6.mean())
```

```
        rect.append(rect7.mean())
        rect.append(rect8.mean())
        diff = []
        for i in range(0,7):
                diff.append(rect[i+1]-rect[i])
        max_val = max(diff)
        diff_ind = diff.index(max_val)

        print "max val = " + str(max_val)
        print "diff ind = " + str(diff_ind)
        avg.append(diff_ind)
        print " "
        f = f+1
        '''
        0 = head
        1 = neck
        4 = waist
        6 = knee
        '''
if len(avg) == 0:
        print "No face Detected"
else:
        avg_depth = sum(avg)/len(avg)
        print "Average Depth = " + str(avg_depth)
        avg_depth = float(avg_depth + 1)
        temp = float(avg_depth/8)
        print temp
        avg_dep = temp * 100
        perc_subm = 100 - ( avg_dep )
        print "Percentage submerged = " + str(perc_subm)

cv2.imshow('Human_Depth',face_img)

return_list = []
return_list.append(amount)
return_list.append(p)
return_list.append(avg_depth)

def dip_out():
        return return_list
```

```
prompt = cv2.waitKey(0)
if prompt == ord('q') | prompt == ord('Q'): # Press Q to exit
        cv2.destroyAllWindows()

"""
================================================
Fuzzy Control Systems: Flood Probability
================================================
* Antecednets (Inputs)
   - 'depth'
        * Fuzzy set : tillknee, tillwaist, tillneck
   - 'color'
        * Fuzzy set : light, medium, dark
   - 'area'
        * Fuzzy set : small, medium, large
* Consequents (Outputs)
   - 'flood'
        * Fuzzy set : low, medium, high
"""
import os
import numpy as np
import skfuzzy as fuzz

# New Antecedent/Consequent objects hold universe variables and
# membership functions
color = fuzz.Antecedent(np.arange(30, 150, 2), 'color')
area = fuzz.Antecedent(np.arange(0, 100, 2), 'area')
depth = fuzz.Antecedent(np.arange(0, 100, 2), 'depth')
flood = fuzz.Consequent(np.arange(0, 100, 2), 'flood')

# Creating membership fuctions for each antecedent and consequent
color['light'] = fuzz.trapmf(color.universe, [30, 35, 65,70])
color['medium'] = fuzz.trapmf(color.universe, [65, 70, 105,110])
color['dark'] = fuzz.trapmf(color.universe, [105, 110, 150,150])

area['small'] = fuzz.trapmf(area.universe, [0, 10, 25,30])
area['medium'] = fuzz.trapmf(area.universe, [25, 30, 50,55])
area['large'] = fuzz.trapmf(area.universe, [50, 55, 100,100])

depth['tillknee'] = fuzz.trapmf(depth.universe, [0, 8, 40,46])
depth['tillwaist'] = fuzz.trapmf(depth.universe, [40, 46, 69,75])
```

```
depth['tillneck'] = fuzz.trapmf(depth.universe, [69, 75, 100,100])

flood['low'] = fuzz.trapmf(flood.universe, [0, 3, 30,33])
flood['medium'] = fuzz.trapmf(flood.universe, [30, 33, 63,66])
flood['high'] = fuzz.trapmf(flood.universe, [63, 66, 100,100])


# Graphs plotted for color, area, depth and flood.
color.view()
area.view()
depth.view()
flood.view()

"""
Fuzzy rules
-----------

Now, to make these triangles useful, we define the
*fuzzy relationship* between input and output variables.
For the purposes of our example, consider three simple rules:

1. If the depth is tillneck AND the color is medium AND
        the area is small, then the flood will be medium.
2. If the depth is tillneck AND the color is light, then
        the flood will be medium.
3. If the depth is tillneck, then the flood will be high.

4. If the depth is tillwaist AND the color is dark AND
        the area is large, then the flood will be high.
5. If the depth is tillwaist AND the color is light AND
        the area is small, then the flood will be low.
6. If the depth is tillneck, then the flood will be medium.

7. If the depth is tillknee AND the color is dark AND
        the area is small, then the flood will be low.
8. If the depth is tillknee AND the color is medium AND
        the area is large, then the flood will be medium.
9. If the depth is tillknee AND color is dark, then
        the flood will be high.
10. If the depth is tillknee, then the flood will be low.
"""
```

```
rule1 = fuzz.Rule(depth['tillneck'] & color['medium'] & area['small'],
        flood['medium'])
rule2 = fuzz.Rule(depth['tillneck'] & color['light'] ,flood['medium'])
rule3 = fuzz.Rule(depth['tillneck'] ,flood['high'])

rule4 = fuzz.Rule(depth['tillwaist'] & color['dark'] & area['large'],
        flood['high'])
rule5 = fuzz.Rule(depth['tillwaist'] & color['light'] & area['small'],
        flood['low'])
rule6 = fuzz.Rule(depth['tillwaist'] ,flood['medium'])

rule7 = fuzz.Rule(depth['tillknee'] & color['dark'] & area['small'],
        flood['low'])
rule8 = fuzz.Rule(depth['tillknee'] & color['medium'] & area['large'],
        flood['medium'])
rule9 = fuzz.Rule(depth['tillknee'] & color['dark'] ,flood['medium'])
rule10 = fuzz.Rule(depth['tillknee'] ,flood['low'])


"""
Control System Creation and Simulation
---------------------------------------

Now that we have our rules defined, we can simply create
a control system via:
"""
flood_ctrl = fuzz.ControlSystem([rule1, rule2, rule3, rule4, rule5,
        rule6, rule7, rule8, rule9, rule10])

"""
In order to simulate this control system, we will create a
``ControlSystemSimulation``.
"""
flooding = fuzz.ControlSystemSimulation(flood_ctrl)

"""
We can now simulate our control system by simply specifying the
inputs and calling the ``compute`` method.
"""
# Pass inputs to the ControlSystem using Antecedent labels with
# Pythonic API
```

```
# Note: if you like passing many inputs all at once,
# use .inputs(dict_of_data)
# depth - 0-100%
# color - 30-150 range
# area - 0-100%

flooding.input['depth'] = 80
flooding.input['color'] = 120
flooding.input['area']  = 70

# Crunch the numbers
flooding.compute()

"""
Once computed, we can view the result as well as visualize it.
"""
print "Probaility = "
print flooding.output['flood']
flood.view(sim=flooding)
try:
    input("Press enter to continue")
except SyntaxError:
    pass
```