HUMAN POSE ESTIMATION

GROUP 1:

Aarathy Sasi
Anju Thampi
Ida Phiyus
Jiya P.S.
Rosemol Augustine
Internal Guide: Dr. Sreeraj M
Assistant Professor, Dept. of MCA

DEPARTMENT OF COMPUTER APPLICATION
FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY(FISAT)

Contents

Abstract

Estimation of human orientation is considered a key configuration to study and understand human behaviours. There have been many works focusing on this. However, most of previous works are sensor-based technologies but visual-based technologies. In this paper, we proposed a method to estimate the orientation of human in a given image patch. We have observed the performance of a feature called Histogram of Oriented Gradients(HOG) on human detection.The results show that the proposed method can effectively estimate the orientation of human pose inside an image with fast and simple additional operations required.

Chapter 1

INTRODUCTION

Analysis of human behaviour from image sequences has attracted much attention for decades because it is very useful and helpful to have a system automatically observing human actions and behaviours via CCTV in many places. For example, detecting theft actions in shop store, alarms for intruders at home, detecting falls in hospital and alerting security guards for harmful actions in train station. Two main keys of this technology is to track configuration of human body along the image sequence and to analyse them through knowledge provided by experts for each applications.While the configurations of interest depends on applications concerned, position and orientation are basic configurations necessary for further analysis.

Although there have been many proposed techniques on human detection and pose estimation to extract location and postures, it is noticed that there have been very few concerning orientation. Estimation of human orientation is considered a key to study and understand human behaviours and there have been many works focusing on doing so. However, most of previous works are sensor-based technologies but visual-based. Based on our knowledges, there are not many visual-based techniques for estimating orientation of human. Here we categorise techniques of orientation estimation in computer vision into two groups,feature-based and silhouette-based. While the feature- based techniques focus on estimating orientation degree from given image features, silhouette-based techniques estimate from foreground pixels obtained from background subtraction process.

Chapter 2

IMPLEMENTATION

We describe the HOG descriptor to tackle the problem of human pose estimation . HOG descriptor is limited to a certain range of geometrical variation not bigger than the bin size. HOG descriptor has been tested by detecting human on dierent orientations of human body to study the human posture .Different images of size greater than 100 is considered as the training data. Few of images are taken from the web using google images and other are real images.Images are classified as positive and negative. In the he captured images we detect the humans.We then apply non-maxima suppression to the bounding boxes using a fairly large overlap threshold to try to maintain overlapping boxes that are still people. Then the information are shown on the number of bounding boxes.Finally the output image.WE consider the Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision and image processing.

The Histogram of Oriented Gradients (HOG) is a feature descriptor . The HOG descriptor is based on image gradients and uses a rectangular environment to collect information for describing the feature.The HOG descriptor is frequently used for detection and thus calculated for all points in an image. The first step of this procedure is, to normalize the image in gamma and color values, to make possible edges detectable in the next steps. Then, the gradient image is computed. This can be done efficiently by convolving the image with the two Sobel-filters [1 0 1] and [1 0 1]T . The results of the two convolutions are converted to one image with gradient magnitudes, and one with gradient orientations. The next step gives the descriptor its abstraction possibilities. The information is accumulated over several pixels, forming a cell. For human detection, the cell size is chosen as 6 6 pixels. For each pixel

within one cell, a vote to an orientation histogram is calculated, equal to its magnitude. For human detection, 9 bins have shown to be the most effective amount of bins. The sign of the orientation is ignored, i.e. the angles are converted to be in [0; 180]. The cells are combined to 3 3 blocks. Several important normalization steps are implemented across blocks and within a cell.

Chapter 3

RESULTS

3.1 Task 1

Our first task was Human Pose Estimation in which detection of humans in images have been long-standing problems in computer vision. Most successful approaches heavily rely on discriminative models to build appearance detectors for body joints and generative models to constrain possible body configurations. We address the problem of estimating 2d human pose from still real world images. The performance of a feature called Histogram of Oriented Gradients (HOG)on human detection was studied and used for this Estimation. We mostly used Modified HOG-based human detector to calculate the score of confidence of the particular images. After detecting the position there were occurrence of a straight line along the body of image.

3.1.1 Algorithm

step 1: Start

step 2: import the necessary packages

step 3: Initialize the HOG descriptor/person detector

step 4: Given the images that is used for the detection.

step 5: Detect people in the image using hog.detectMultiScale and draw the rectangular boxes.

step 6: Apply non-maxima suppression to the bounding boxes using a fairly large overlap threshold to try to maintain overlapping boxes that are still people.

step 7: Finally draw the bounding boxes for the stored value of array in pick.

step 8: show some information on the number of bounding boxes

step 9:output images

step 10: Stop

3.1.2 Dataset

The dataset used in this project is INRIA person.
1. Contributions

- Images from personal digital image collections taken over a long time period. Usually the original positive images were of very high resolution (approx. 2592x1944 pixels), so we have cropped these images to highlight persons. Many people are bystanders taken from the backgrounds of these input photos, so ideally there is no particular bias in their pose.
- Few of images are taken from the web using google images.

Note

- Only upright persons (with person height greater than 100) are marked in each image

2. Original Images

- Images Folders 'Train' and 'Test' correspond, respectively, to original training and test images. Both folders have two sub folders:
  (a) 'pos' (positive training or test images)
  (b) 'neg' (negative training or test images.)

3. Normalized

- Normalized Images Folders 'train-64x128-H96' and 'test-64x128-H96' correspond to normalized dataset as used in above referenced paper. Both folders have two sub folders:
  (a) 'pos' (normalized positive training or test images centered on the person with their left-right reflections)
  (b) 'neg' (containing original negative training or test images).

Note

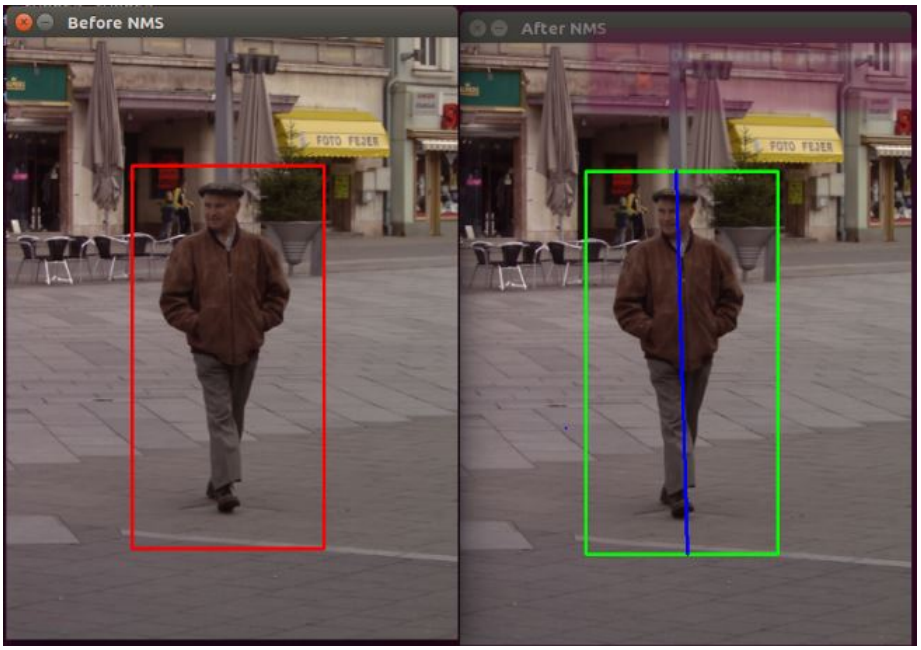
- images in folder 'train/pos' are of 96x160 pixels (a margin of 16 pixels around each side), and images in folder 'test/pos' are of 70x134 pixels (a margin of 3 pixels around each side). This has been done to avoid boundary conditions (thus to avoid any particular bias in the classifier). In both folders, use the centered 64x128 pixels window for original detection task.

3.1.3 Program Code

```
from -future- import print-function
from imutils.object-detection import non-max-suppression
from imutils import paths
import numpy as np
import argparse
import imutils
import cv2
hog = cv2.HOGDescriptor() hog.setSVMDetector(cv2.HOGDescriptor-getDefaultPeopleDetector()
imm = "images.png"
image = cv2.imread(imm)
image = imutils.resize(image, width=min(400, image.shape[1]))
orig = image.copy()
(rects, weights) = hog.detectMultiScale(image, winStride=(4, 4),padding=(8,
8),scale=1.05)
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
pick = non-max-suppression(rects, probs=None, overlapThresh=0.65)
print(rects)
print(pick)
print(len(rects))
```

print(len(pick))
for (xA, yA, xB, yB) in pick:
cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
cv2.line(image, (xA+80,yA), (xB-80,yB), (255,0,0), 2)

3.1.4 Output



3.2 Task 2

In the Second task we assume to estimate the orientation of the person in real time for that we performed camera calibration by which we could obtain the camera parameters like focal length ,distortion co-efficient etc.The image is taken from a fixed distance.Using this distance and the camera parameters the size is computed. The input that we provide is an image and the output we tried to estimate the position of a person by representing with a line.But we estimated the position with line whichever position the line will be formed ,but the risk we faced is that not only the image we capture will be represented with line but also wherever we click the line will be formed so

we planned to detect the only person that we are captured and to represent the line but detection we done but whatever images it all will detected so we tried to estimate the position of the person we detected that also failed line will be formed wherever we click.The image was loaded and displayed. The mouse click coordinates are displayed when double clicked on the image and pressed a.The line will be formed but the final output we are not able to complete what is needed for us.

1. DRAW A LINE BETWEEN TWO MOUSE CLICK COORDINATES AND PRINT ITS LENGTH

The concept of list was used to store the coordinates. The mouse click coordinates were stored into list on clicking them. These two coordinates were given as the endpoints to draw the line. The length of the line is computed using distance formula.Distance formula is as follows: sqrt(pow((jx-ix),2)+pow((jy-iy),2)) where (ix,iy),(jx,jy) are the coordinates of the starting and ending points of the line.

2. CAMERA PARAMETERS MEASURING

For doing this project we need to perform camer calibration which determines the camera parameters.
Two of the camera parameters are:-

- Intrinsic Parameters
  The intrinsic parameters characterize the optical,geometric digital characteristics of the camera such as focal length,principle point,scaling factors,skew factors and lens distortion. These parameters are usually expressed in the form of a matrix, often referred to as the camera matrix.

- Extrinsic parameters
  corresponds to rotational and translational vectors which translates the coordinates of a 3D point to a coordinate.

3. FIND THE ACTUAL SIZE OF THE OBJECT IN THE IMAGE

After learning the camera parameters we had to find the actual size of a selected line. For this we need to know two things,

- The distance between the camera and the image plane(distance from camera to object).
- Focal-length(in mm and pixels per mm) or physical size of the image sensor.

4. USER INTERFACE

The GUI was created using wxPython The user is asked to browse for a jpg file and to enter the distance from camera to object. Corresponding unit of measurement can be selected from radio buttons. Then the user can click on Continue button to get the result or Exit button to exit from the user interface.

3.2.1 Algorithm

Step1: start

Step2: capture the image using web camera

Step 3: Compute Pixels per millimeter

Step 4: Find the object size in pixels

Step 5: Convert px/mm in the lower resolution Size of object in the image sensor

Step 6: mark the end points

Step 7: draw the line

Step 8: stop

3.2.2 Program Code

```
import wx
import numpy as np
import cv2
import glob
import math
import os
import wx.lib.
from imutils.object-detection import non-max-suppression
from imutils import paths
import argparse
import imutils
ix,iy,jx,jy=-1,-1,-1,-1
list=[]
def mouseclick1(event,x,y,flags,param):
global ix,iy,jx,jy
global list
if event == cv2.EVENT-LBUTTONDBLCLK:
list.append([x,y])
class Mywin(wx.Frame):
dist=0
def OnRadiogroup(self, e):
global dist
dist=float(self.t2.GetValue())
rb = e.GetEventObject()
unit = rb.GetLabel()
if unit=='mm':
dist=dist
if unit=='cm':
dist=dist*10
if unit=='m':
dist=dist*1000
def OnTimeToClose(self, evt):
"""Event handler for the exit button click."""
exit()
def OnGoButton(self, evt):
"""Event handler for the continue button click."""
```

criteria = (cv2.TERM-CRITERIA-EPS + cv2.TERM-CRITERIA-MAX-ITER,
30, 0.001)
pattern-size = (7,5)
objp = np.zeros((np.prod(pattern-size), 3), np.float32)
objp[:,:2] = np.indices(pattern-size).T.reshape(-1, 2)
objpoints = []
imgpoints = []
images = glob.glob('sample*.jpg')
for fname in images:
img = cv2.imread(fname)
gray = cv2.cvtColor(img,cv2.COLOR-BGR2GRAY)
ret, corners = cv2.findChessboardCorners(gray, (7,5),None)
h, w = img.shape[:2]
if ret == True:
objpoints.append(objp)
corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
imgpoints.append(corners2)
rms, camera-matrix, dist-coefs, rvecs, tvecs = cv2.calibrateCamera(objpoints,
imgpoints, (w, h), None, None)
fx,fy=camera-matrix[0][0],camera-matrix[1][1]
fxy=(fx+fy)/2
f=3.6
m=fxy/f
cv2.imshow("Before NMS", orig)
cv2.imshow("After NMS", img)
print img
path=self.fbb.GetValue()
img = cv2.imread(path)
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor-getDefaultPeopleDetector())
(rects, weights) = hog.detectMultiScale(img, winStride=(8, 8),padding=(16,
16), scale=1.05)
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
pick = non-max-suppression(rects, probs=None, overlapThresh=0.65)
for (xA, yA, xB, yB) in rects:
cv2.rectangle(img, (xA, yA), (xB, yB), (0, 255, 0), 2)
cv2.imshow('image',img)
cv2.setMouseCallback('image',mouseclick1)

```
while(1):
k = cv2.waitKey(20) 0xFF
if k==27:
break
if k == ord('a'):
ix,iy=list[0][0],list[0][1]
jx,jy=list[1][0],list[1][1]
d=math.sqrt(math.pow((jx-ix),2)+math.pow((jy-iy),2))
mid=(((ix+jx)/2),((iy+jy)/2)-10)
h1, w1 = img.shape[:2]
x=(w1*m)/w
sizeimgsen=d/x
realsize=(dist*sizeimgsen)/(f)
strval=str(realsize)
img1 = cv2.line(img,(ix,iy),(jx,jy),(0,0,255),3)
font = cv2.FONT-HERSHEY-SIMPLEX
cv2.putText(img1,strval,mid, font, .5, (0,255,0), 2, cv2.LINE-AA)
cv2.imshow('image',img1)
del list[:]
cv2.waitKey(0)
cv2.destroyAllWindows()
def -init-(self, parent, title):
wx.Frame.-init-(self, parent, -1, title,pos=(150, 150), size=(1000, 500))
panel = wx.Panel(self)
vbox = wx.BoxSizer(wx.VERTICAL)
hbox1 = wx.BoxSizer(wx.HORIZONTAL)
self.fbb = wx.lib.filebrowsebutton.FileBrowseButton(panel,labelText="Select
a jpg file:")
hbox1.Add(self.fbb, 1, wx.ALIGN-LEFT)
vbox.Add(hbox1)
hbox2 = wx.BoxSizer(wx.HORIZONTAL)
l2 = wx.StaticText(panel, -1, "Enter the distance from object to camera:")
hbox2.Add(l2, 1, wx.EXPAND—wx.ALIGN-LEFT—wx.ALL,5)
self.t2 = wx.TextCtrl(panel)
hbox2.Add(self.t2,1,wx.EXPAND—wx.ALIGN-LEFT—wx.ALL,5)
self.t2.Bind(wx.EVT-TEXT,None)
vbox.Add(hbox2)
hbox5 = wx.BoxSizer(wx.HORIZONTAL)
```

```
self.rb1 = wx.RadioButton(panel,11, label = 'mm',style = wx.RB-GROUP)
self.rb2 = wx.RadioButton(panel,22, label = 'cm')
self.rb3 = wx.RadioButton(panel,33, label = 'm')
hbox5.Add(self.rb1,1)
hbox5.Add(self.rb2,1)
hbox5.Add(self.rb3,1)
self.Bind(wx.EVT-RADIOBUTTON,self.OnRadiogroup)
vbox.Add(hbox5)
hbox4 = wx.BoxSizer(wx.HORIZONTAL)
l3 = wx.StaticText(panel, -1, "INSTRUCTION: double click the Endpoints
and press 'a'")
SetForegroundColour((255,0,0))
hbox4.Add(l3, 1, wx.EXPAND—wx.ALIGN-LEFT—wx.ALL,5)
vbox.Add(hbox4)
hbox3 = wx.BoxSizer(wx.HORIZONTAL)
exitbtn = wx.Button(panel, -1, "Quit",pos=(20,120))
hbox3.Add(exitbtn,1)
gobtn = wx.Button(panel, -1, "Start",pos=(60,120))
hbox3.Add(gobtn,1)
vbox.Add(hbox3)
self.Bind(wx.EVT-BUTTON, self.OnTimeToClose, exitbtn)
self.Bind(wx.EVT-BUTTON, self.OnGoButton, gobtn)
panel.SetSizer(vbox)
self.Centre()
self.Show()
self.Fit()
app = wx.App()
Mywin(None, 'Estimation of Human Pose')
app.MainLoop()
```

3.2.3 Output

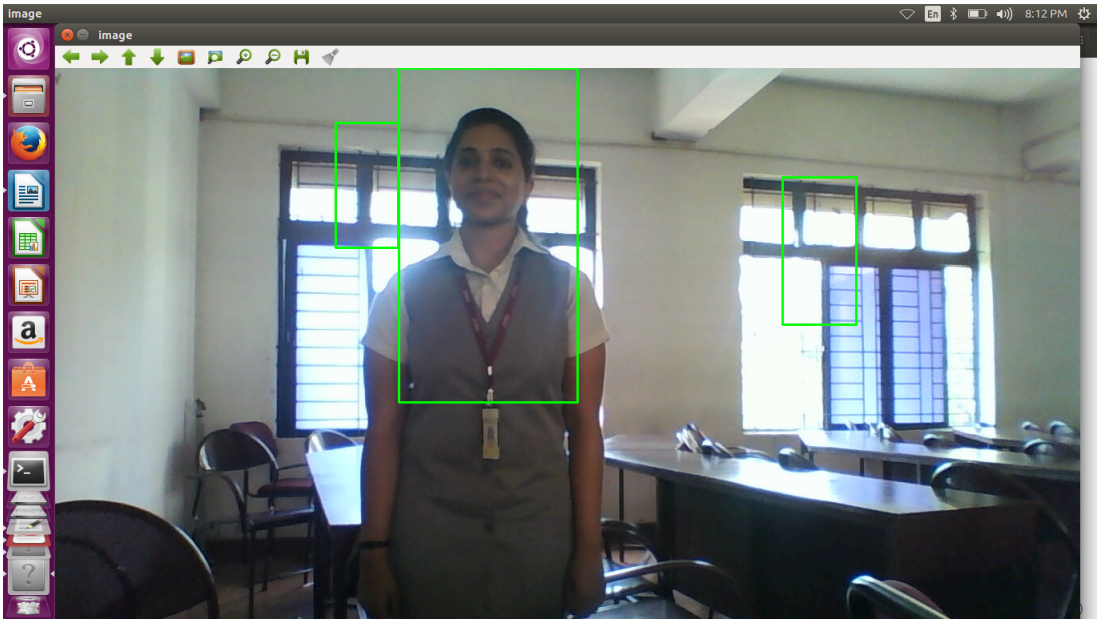


image
8:12 PM
image

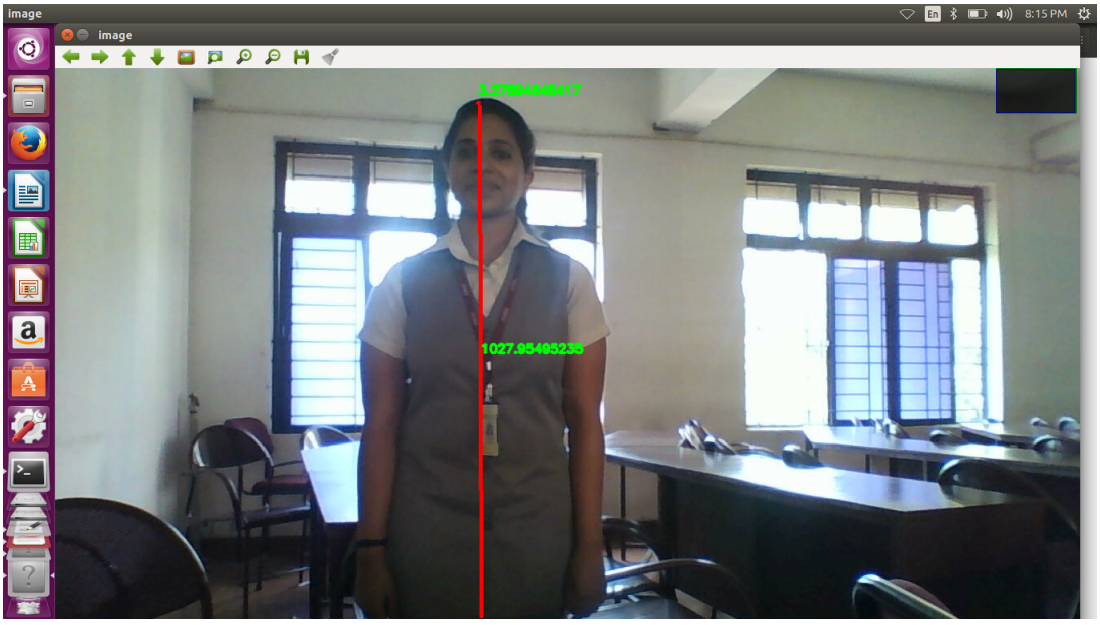


image
8:15 PM
image
3.27804848417
1027.95495235

Chapter 4

ROLE OF MEMBERS

1. Anupa Philip : Scrum Master
   - Organises Daily scrum meetings
   - Worked behind the program code.
2. Aarathy Sasi: Team Leader
   - Assigned different tasks to the team members after dividing it into sub tasks,
   - Worked behind the program code to detect the humans.
3. Anju Thampi
   - Worked behind the code to drawn a line in human sternum.
4. Rosemole Augustine
   - Worked behind the latex report.
   - Worked behind the program code to detect the humans.
5. Jiya P.S
   - Worked behind the latex and beamer reports.
   - Worked behind the code to drawn a line in human sternum.
6. Ida Phiyus
   - Worked behind the program code and searching more about different algorithms .

Chapter 5

PROOF OF CONCEPT

14-11 MVA2011 IAPR Conference on Machine Vision Applications, June 13-15, 2011, Nara, JAPAN

Estimation of Human Body Orientation using Histogram of Oriented Gradients

Kittipanya-ngam Panachit
Institute for Infocomm Research
1 Fusionpolis Way
Singapore, 138632
pngam@i2r.a-star.edu.sg

Ong Soh Guat
Institute for Infocomm Research
1 Fusionpolis Way
Singapore, 138632
sgong@i2r.a-star.edu.sg

Eng How-Lung
Institute for Infocomm Research
1 Fusionpolis Way
Singapore, 138632
hleng@i2r.a-star.edu.sg

Abstract

Estimation of human orientation is considered a key configuration to study and understand human behaviours. There have been many works focusing on this. However, most of previous works are sensor-based technologies but visual-based technologies. In this paper, we proposed a method to estimate the orientation of human in a given image patch. We have observed the performance of a feature called Histogram of Oriented Gradients (HOG) on human detection and found the sensitivity of this feature in in-plane rotation. It is so sensitive that it could be used in classifying images of rotated and non-rotated human body. Therefore we have drawn an idea to exploit a modified HOG descriptor with 'bin shifting' technique to estimate the orientation of human. The results show that the proposed method can effectively estimate the orientation of human pose inside an image with fast and simple additional operations required.

1 Introduction

Analysis of human behaviour from image sequences has attracted much attention for decades because it is very useful and helpful to have a system automatically observing human actions and behaviours via CCTV in

Estimation of human orientation is considered a key to study and understand human behaviours and there have been many works focusing on doing so. However, most of previous works are sensor-based technologies[1] but visual-based. Based on our knowledges, there are not many visual-based techniques for estimating orientation of human. Here we categorise techniques of orientation estimation in computer vision into two groups, feature-based and silhouette-based. While the feature-based techniques focus on estimating orientation degree from given image features, silhouette-based techniques estimate from foreground pixels obtained from background subtraction process[2, 3]. Iwasawa *et al.*, [2] track the change of the principal axis of the foreground area obtained from background subtraction in Thermal images. Lee *et al.*, [3] estimate ellipse parameters covering foreground image which contains orientation information. Bay *et al.*,[4] proposed to estimate the main orientation of the target from wavelet features. Chen *et al.*, [5] to estimate the orientation from Weber Local Descriptor.

Here we observed the performance of a feature called Histogram of Oriented Gradients (HOG) [6] on human detection and found the sensitivity of this feature in in-plane rotation. It is so sensitive that it could classify rotated and non-rotated human body. Therefore this paper suggests an idea to exploit a modified HOG

Chapter 6

FUTURE ENHANCEMENT

As a future work we can include inexpensive programmable depth sensors, wearable devices, and smart phones in our project. The system can be designed in a way to track the activities of consented workers in a workplace using the depth sensors, alert them discreetly on detection of non-compliant activities, and produce cumulative reports on their performance.

Chapter 7

CONCLUSION

Human pose Estimation from images is a software developed to measure the orientation of a person .This software we some how completed keeping in mind that it meets all the basic requirements but the real time estimation of human orientation we faced lot of risk .The software takes an image as the input.The calibration process is performed by the software itself which gives us the values of various camera parameters.These parameters along with the distance of the object to the camera gives the actual value of the object in the image. The GUI for the software was developed using wxPython.The file path is browsed by the user and the distance is entered to continue the processing.

Chapter 8

REFERENCES

[1]Kittipanya-ngam Panachit,Ong Soh Guat,Eng How-Lung:Estimation of Human Body Orientation using Histogram of Oriented Gradients,IEEE Trans

[2]https://realpython.com/blog/python/face-detection-in-python-using-a-webcam/

[3] https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/

[4]//opencvexamples.blogspot.com/2013/09/opencv-example-to-load-and-display-image.html