Virtual Mirror – A Hassle Free Approach To the Use of Trial Room

CONTENTS

INTRODUCTION

Shopping for clothes online has obvious drawbacks. The selling items are inaccessible to the customers, so it is impossible for the customers to physically try on the clothes on themselves until the items are delivered. Customers always make decisions according to the pictures of models online. However, items look good on models does not imply they look good on the buyers as well. Therefore, the inaccessibility of purchasing items at selection time potentially increases the rate of item returns as a result of poor fit.

To conclude what has been observed online shopping has unavoidable limitations which give rise to customers' unsatisfied shopping experiences as well as the retailers' losses of potential sells. In order to deal with the "lose-lose" situation, people are keen on looking for strategies and techniques. Then, an idea called "Virtual Dressing Room" is put forward. Customers would be able to perceive the visual image of how they look like by trying on clothes virtually so that they can easily filter out some selections without taking turns to use the fitting rooms. Compare with the Physical dressing room, "Virtual Dressing Room" takes each user time. Thus, it increases the shopping efficiency for all customers, hence, enhances the shopping experience.

This project report describes an interactive visual computer system called "Virtual Dressing Room" which implemented the concept of "Virtual Try-On" with the help of a script written with Python programming language and OpenCV library. A virtual dressing room (also often referred to as a virtual fitting room and the virtual changing room although they do, on examination, perform different functions) is the online equivalent of an in-store changing room. It enables shoppers to try on clothes to check one or more of size, fit or style, but virtually rather than physically. The following sections give further details about this project in terms of its objectives, related works, system design, development tools, implementation iterations, evaluations, and feedback.

Virtual Dressing Room is an interactive system that mimics the real fitting experiences. Displaying the user's whole body image is a practical concern. Unlike a usual user interface, the screen requires a large portion of center space to display the user's mirror image. Then, the remaining part of the screen should be utilized effectively in order to display other information, such as instructions and functional menus.

The main objective of this project is to build a virtual system where e-commerce users or customers could try out clothing items before they make their purchases without trying on the clothes for real. In this way, the user can try out clothes quickly and easily which can improve their decision process and saves their meaningful time as well.

LITERATURE SURVEY

Upon doing extensive research on the development of Virtual Dressing Room or other such projects done so far, we came across a relatively fewer number of results as this topic has been a research topic for a decade now. However, we did come across a few amazing projects done similar but with the use of advanced hardware and complicated programming which made such a surreal experience come alive on a better scale. All of the projects we came across had made use of the Microsoft Kinect, which is basically a depth sensor camera used for the accurate 3D tracking and positioning. In the lack of such sophisticated hardware, we had to make use of the smartphone cameras to generate the virtual experience which leads us to what we have developed in this project. What we came to know that smartphone cameras are not a reliable source for this kind of stuff as the object tracking was not reliable from this camera. The design process of identifying subsystems and establishing the framework for a subsystem control and communication is called architectural design. It is a creative process where one tries to establish a system organization that will satisfy the functional and non-functional requirements of the system. It represents the structure of data and program components. It represents a set of abstractions that enable software engineers to describe architecture in predictable ways. The product of the architectural design process is an architectural design document which includes graphical representations of the system along with associated descriptive text. The architectural design process is concerned with the establishment of a structural framework. It defines the major components of a system and communication between those components. Figure gives an architectural overview of the project. The "Virtual Mirror" implements a real-time virtual trial room system using the camera. The system first captures the image Then data generated is the skeletal data of the user which is then sent to Unity. This skeletal data is then used in Unity to generate the clothes and impose it on the user. A live streaming video of the user with the clothes imposed is then displayed as the output on the screen.

2.1 Existing System

Compared to shoes, masks, glasses, and watches, virtual try-on 3D clothing still remains a challenge. The reason is that clothes are deformed when taking the shape of a person's body. Thus, for proper AR experience, a deep learning model should identify not only basic key points on the human body's joints but also the body shape in 3D.

Looking at one of the most recent deep learning models DensePose aimed to map pixels of an RGB image of a person to the 3D surface of the human body, we can find out that it's still not quite suitable for augmented reality. The DensePose's inference speed is not appropriate for real-time apps, and body mesh detections have insufficient accuracy for the fitting of 3D clothing items. In order to improve results, it's required to collect more data which is a time and resource-consuming task.

alternative is to use 2D clothing items and 2D people's silhouettes. That's what

Zeekit company does, giving the users a possibility to apply a number of clothing types (dresses, pants, shirts, etc.) to their photo.

2.2 2D Clothes Transferring

Strictly speaking, the method of 2D clothes images transferring cannot be considered as Augmented Reality, since the "Reality" aspect implies the real-time operation, however, it still can provide an unusual and immersive user experience. The behind technologies comprise Generative Adversarial Networks, Human Pose Estimation, and Human Parsing models. The 2D clothes transferring algorithm may look as follows:

1. Identification of areas in the image corresponding to the individual body parts
2. Detection of the position for identified body parts
3. Producing of a warped image of a transferred clothing
4. Application of a warped image to the image of a person with the minimum produced artifacts

2.3 Smart Mirror in Market of Fashion

There are several fashion brands that have already started to use smart mirrors to improve their customer experience. These brands include:

- Topshop
- Ralph & Lauren
- H&M
- Zara
- Burberry

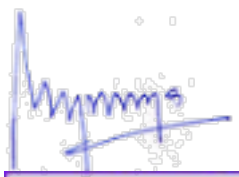
The advent of the latest technologies such as AR, VR, AI, and more have helped businesses to improve their overall customer experience and drive sales. AR-powered smart mirrors that are rapidly becoming a part of physical shopping stores have transformed the way customers used to shop. It helps in delivering personalized shopping experience, acts as a shopping assistant and offers tips and suggestions, provides the ease of trying out clothes, and increases sales, and renders many other benefits.

SCOPE OF PROJECT

Scope of Project

The main scope of the project Virtual Mirror – A Hassle Free Approach to the Use of Trial Room is to provide a virtual trail room to its users. Here an idea called the "Virtual Try On" using The Virtual Mirror is put forward. Customers would be able to perceive the visual image of how they look by trying on clothes virtually, so that they can easily filter out some selections without taking turns to use the fitting rooms and the embedded feature of e-commerce can be utilized to make purchases more convenient. Comparing with "Physical Try On", "Virtual Try On via a Virtual Mirror" takes much less time. Thus, it increases the shopping efficiency for all customers and enhances the shopping experience.

By providing your customers top-class services such as trying out clothes using a smart mirror, you can improve customer engagement to a great extent. It would also help you in achieving the desired level of customer satisfaction. Trying various clothes using a virtual mirror not only saves their time but also makes them try different clothing without the need of using a trial room. A smart mirror is also capable of displaying a 360 view of your virtual model while you try new clothes.



METHODOLOGY

HTML: The **HyperText Markup Language**, or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

CSS: **Cascading Style Sheets** (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Python: **Python** is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms,including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

DETAILS OF WORKING PROCESSES

Software Requirements:

- Front End: HTML, CSS
- BackEnd: Python
- Operating System: Windows any version
- IDE – Microsoft Visual Studio

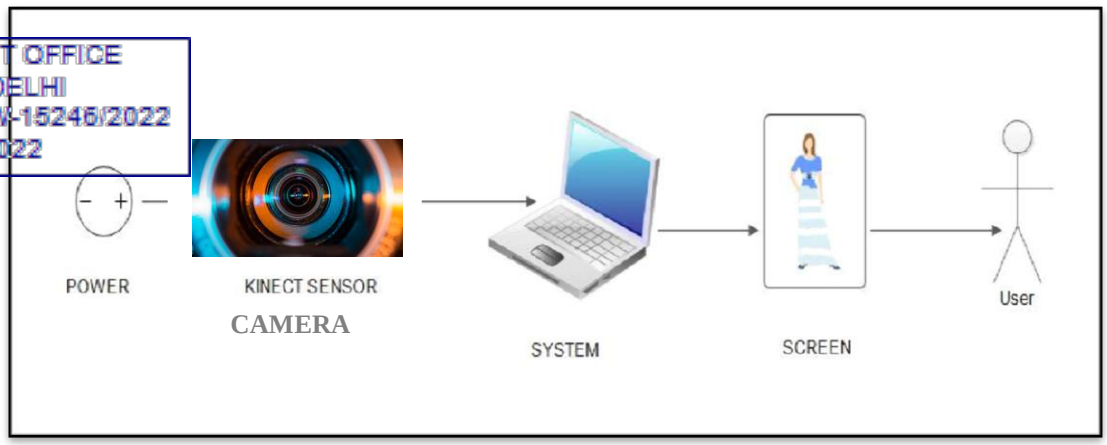
Hardware Requirements:

- **LCD Monitor**
- **Processor**: Intel i3 or above
- **Ram**: 4GB
- **Speed**: 1.8 Ghz

❖ Architectural Diagram

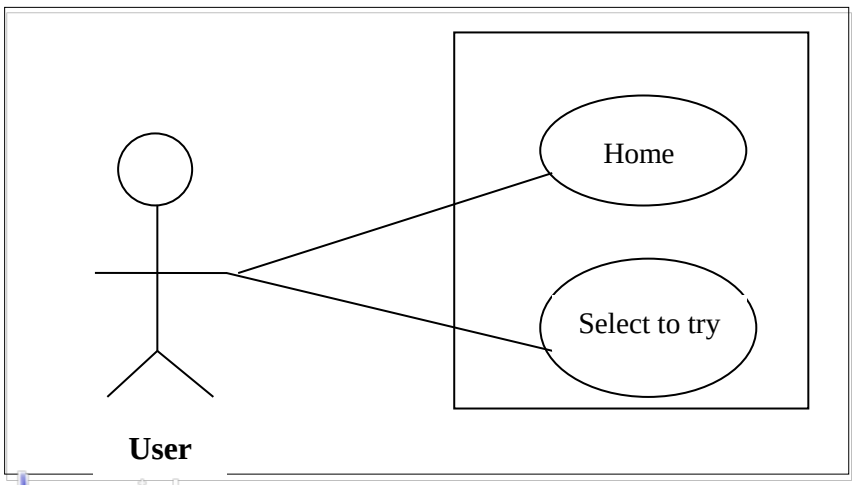
The design process of identifying subsystems and establishing the framework for a subsystem control and communication is called architectural design. It is a creative process where one tries to establish a system organization that will satisfy the functional and non-functional requirements of the system. It represents the structure of data and program components. It represents a set of abstraction that enables software engineers to describe architecture in predictable ways. The product of the architectural design process is an architectural design document which includes graphical representations of the system along with associated descriptive text.

The architectural design process is concerned with the establishment of a structural framework. It defines the major components of a system and communication between those components. Figure gives an architectural overview of the project.



Basic Architectural diagram of Virtual Mirror

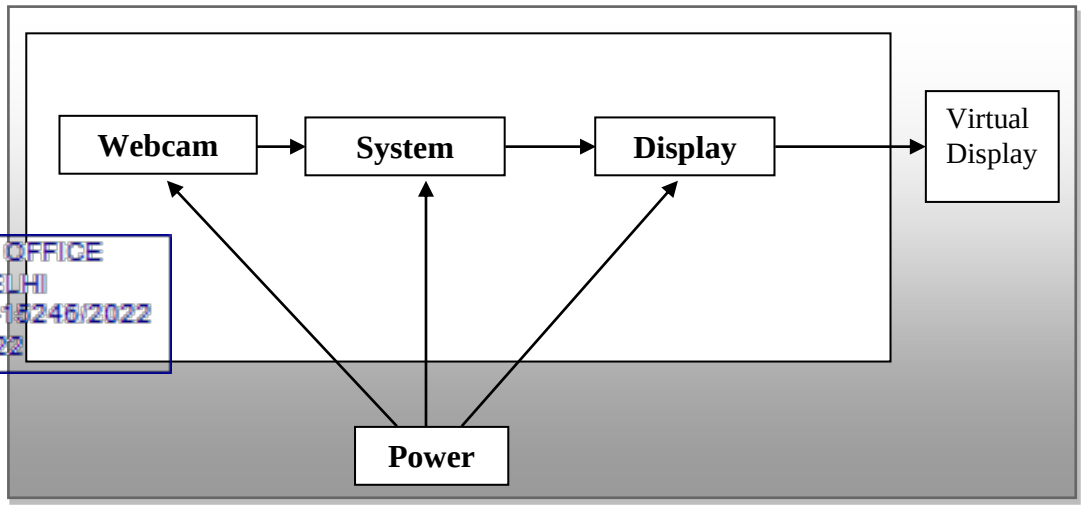
❖ Use Case Diagram



Use case for Virtual Mirror

❖ Modular Design Diagram

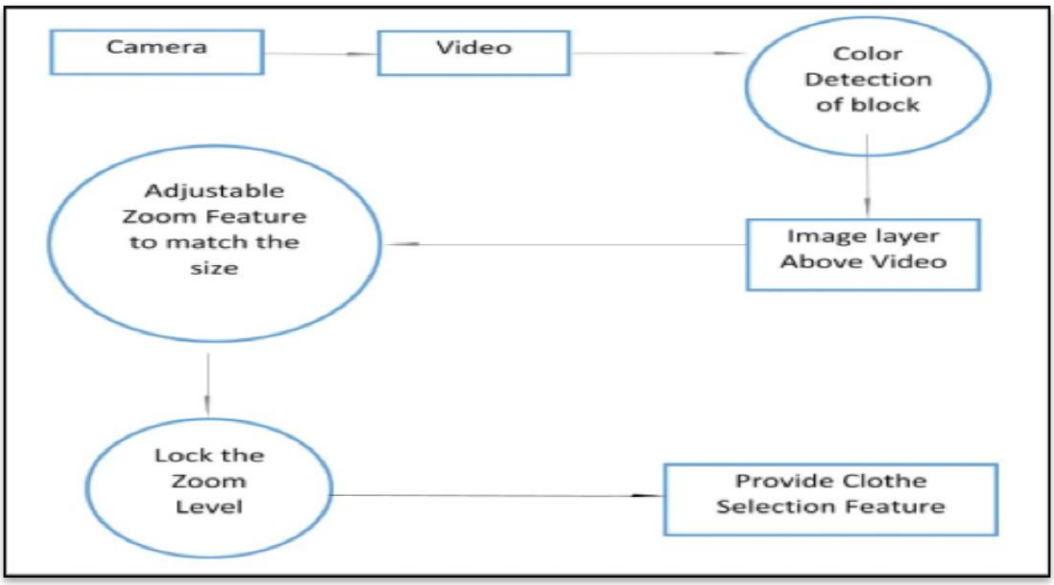
Modular design is a design approach that subdivides a system into smaller parts called modules, which can be independently created and then used in different systems. A modular system can be characterized by functional partitioning into discrete scalable, reusable modules, rigorous use of well-defined modular interfaces, and making use of industry standard for interfaces. Figure provides the modular diagram for the Virtual Mirror.



Modular Design diagram for virtual mirror

❖ Data Flow Diagram

The Data Flow Diagram (DFD) is the graphical representation of the flow of data between the various processes in the system. Each actor is involved in triggering various events that lead to the data transmission among various components. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFD's can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. Figure shows the dataflow diagram of the Virtual Mirror.



Dataflow diagram of the Virtual Mirror

❖ **Video Block**

This is the first process to try Virtual Dressing Room. A Webcam is used to capture video. The Video from the camera is processed and some UI is integrated with the same video screen where users could try the outfit.

❖ **Color Detection Block**

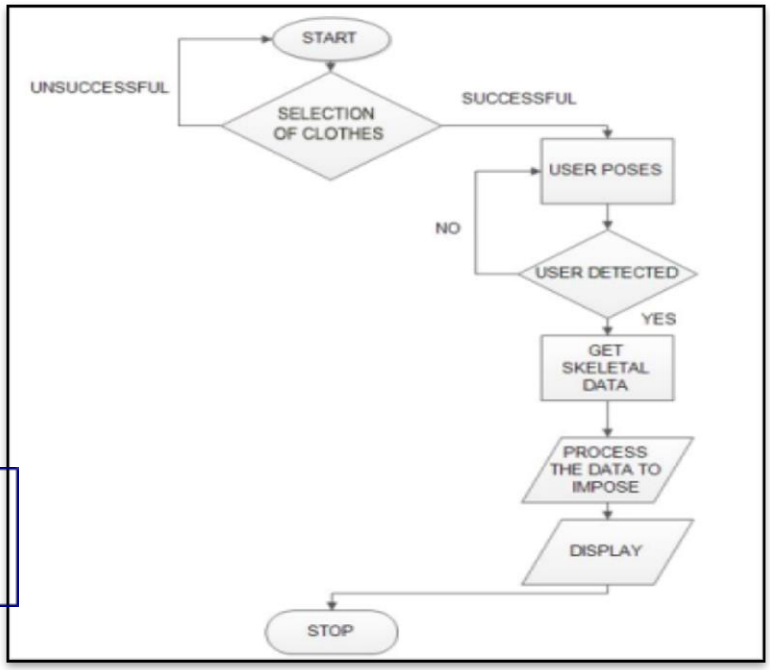
To enhance the adjustable zoom feature, a solid colored block is taken by the user and moved back and forth. The color of the block is detected and processed and the outfit is placed at the portion of the colored block. The image of the outfit is placed above the video and the zoom level can be adjusted moving the colored block.

Control Flow Diagram

❖ **Complete System Flow Diagram**

System Flow Diagram is basically a graphical and sequential representation of the major steps involved in a systematic process. A System Flow Diagram shows what kind of

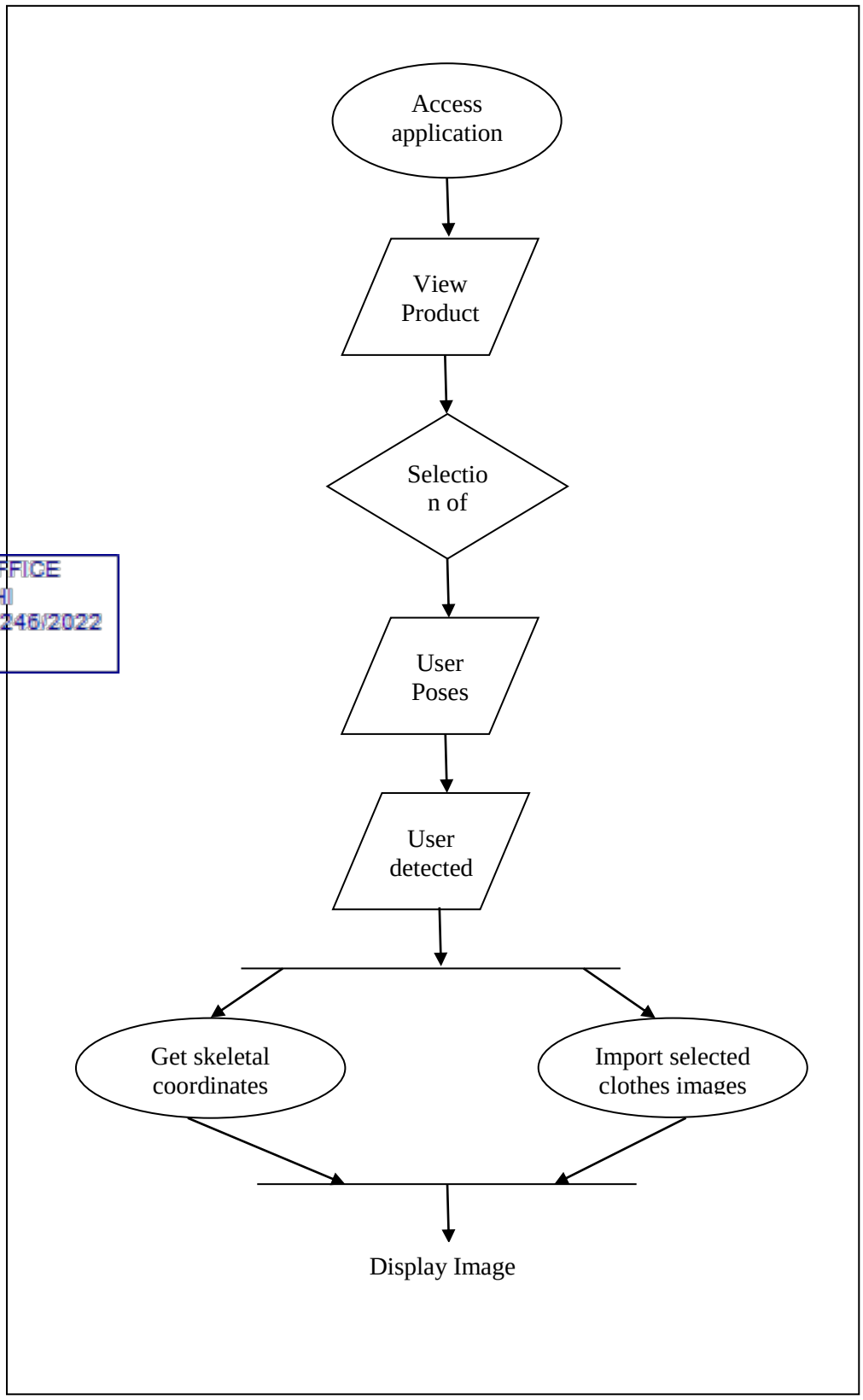
information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. Figure shows the Control Flow Diagram for the Virtual Mirror.



Control flow diagram for virtual mirror

Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is a flow chart to represent the flow form of one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagram deals with all types of flow control by using different elements like fork , join etc. Activity Diagram are also useful in formulating a use case by describing what action needs to take place and when they should occur. It describes complicated sequential algorithm and modeling applications with parallel processes. Activity diagrams should not give details about how the objects behave or how objects collaborate. The Figure shows the Activity Diagram for Virtual Mirror.



Activity diagram for virtual mirror.

Code

First 10 pages

DNNTest.py

```
# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import imutils
import time
import cv2
import os

#os.system("sudo modprobe bcm2835-v4l2")

'''CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]'''

CLASSES = ["", "", "", "", "",
    "", "", "", "", "", "", "",
    "", "", "", "person", "", "",
    "", "", ""]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe("MobileNetSSD_deploy.prototxt.txt",
"MobileNetSSD_deploy.caffemodel")

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
fps = FPS().start()

# loop over the frames from the video stream
count=0
while (True):
    frame = vs.read()
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),0.007843, (300, 300), 127.5)
    net.setInput(blob)
    detections = net.forward()
    #print(detections)

    over the detections
    np.arange(0, detections.shape[2]):
```

```
        confidence = detections[0,0,i,2]
        if(confidence > 0.2):
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            label = CLASSES[idx]
            cv2.rectangle(frame, (startX, startY), (endX, endY),COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx], 2)
            if(label=='person'):
                count+=1
                print("number of persons ",count)
    if(count == 0):
        print("No person detected")
    if(count == 1):
        print("1 person detected")
    if(count > 2):
        print("Additional person detected in the room.")
    # show the output frame
    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1) & 0xFF
    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

    fps.update()
    count = 0

fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

MySite.py

```
# import the necessary packages
from flask import Flask, render_template, redirect, url_for, request,session,Response
#from werkzeug import secure_filename
import os
import cv2
from supportFile import get_frame

sk(__name__)
```

```
app.secret_key = '1234'
app.config["CACHE_TYPE"] = "null"
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0

@app.route('/', methods=['GET', 'POST'])
def landing():
        return render_template('home.html')

@app.route('/home', methods=['GET', 'POST'])
def home():
        return render_template('home.html')

@app.route('/video', methods=['GET', 'POST'])
def video(id):
        return render_template('video.html')

@app.route('/input', methods=['GET', 'POST'])
def input():
        if request.method == 'POST':
                a = request.form['a']
                b = request.form['b']
                c = request.form['c']
                d = request.form['d']
                e = request.form['symptoms']
                print(a,b,c,d,e)
                return render_template('input.html',result=a+b+c+d+e)

        return render_template('input.html')



@app.route('/video_stream/<id>')
def video_stream(id):
   return Response(get_frame(id),mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/image', methods=['GET', 'POST'])
def image():
        if request.method == 'POST':
                if 'Upload' in request.form:
                        print('inside')
                        savepath = r'upload/'
                        photo = request.files['photo']

        photo.save(os.path.join(savepath,(secure_filename(photo.filename))))
                        image =
cv2.imread(os.path.join(savepath,secure_filename(photo.filename)))

                cv2.imwrite(os.path.join("static/images/","test_image.jpg"),image)
                        return render_template('image.html')
        return render_template('image.html')
```

```
# No caching at all for API endpoints.
@app.after_request
def add_header(response):
        # response.cache_control.no_store = True
        response.headers['Cache-Control'] = 'no-store, no-cache, must-revalidate, post-
check=0, pre-check=0, max-age=0'
        response.headers['Pragma'] = 'no-cache'
        response.headers['Expires'] = '-1'
        return response


if __name__ == '__main__':
        app.run(host='0.0.0.0', debug=True, threaded=True)
```

SupportFile.py

```
import time
import numpy as np
import imutils
import time
import cv2
import os

'''CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]'

CLASSES = ["", "", "", "", "",
    "", "", "", "", "", "", "",
    "", "", "", "person", "", "",
    "", "", ""]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe("MobileNetSSD_deploy.prototxt.txt",
"MobileNetSSD_deploy.caffemodel")



def get_frame(id):
    camera_port=0
    camera=cv2.VideoCapture(camera_port) #this makes a web cam object

    while True:
        ng = camera.read()
        ) = ir g.shape[:2]
```

```
        blob = cv2.dnn.blobFromImage(cv2.resize(img, (300, 300)),0.007843, (300, 300),
127.5)
        net.setInput(blob)
        detections = net.forward()
        #print(detections)

        # loop over the detections
        for i in np.arange(0, detections.shape[2]):
            confidence = detections[0,0,i,2]
            if(confidence > 0.2):
                idx = int(detections[0, 0, i, 1])
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")
                label = CLASSES[idx]
                ht = endY - startY
                bt = round(ht*0.42)
                tp = round(ht*0.18)
                if label == 'person':

                    s_img = cv2.imread("static/clothes/"+str(id)+".png", -1)

                    cv2.rectangle(img, (startX, startY+tp), (endX, endY-bt),COLORS[idx], 2)
                    y = startY - 15 if startY - 15 > 15 else startY + 15
                    cv2.putText(img, label, (startX, y),cv2.FONT_HERSHEY_SIMPLEX, 0.5,
COLORS[idx], 2)

                    x1 = startX
                    x2 = endX
                    y1 = startY+tp
                    y2 = endY-bt
                    s_img = cv2.resize(s_img, (x2-x1, y2-y1))
                    alpha_s = s_img[:, :, 3] / 255.0
                    alpha_l = 1.0 - alpha_s

                    print("s_img",s_img.shape)
                    print("img",img.shape)

                    print("s_alfa",s_img.shape)
                    print("img_alfa",img.shape)

                    for c in range(0, 3):
                        img[y1:y2, x1:x2, c] = (alpha_s * s_img[:, :, c] +
                                                alpha_l * img[y1:y2, x1:x2, c])

        imgencode=cv2.imencode('.jpg',img)[1]
        stringData=imgencode.tostring()
        yield (b'--frame\r\n'
               Content-Type: text/plain\r\n\r\n'+stringData+b'\r\n')

        era)
```

MobilenetSSD_Deploy.Prototttxt.txt

```
name: "MobileNet-SSD"
input: "data"
input_shape {
  dim: 1
  dim: 3
  dim: 300
  dim: 300
}
layer {
  name: "conv0"
  type: "Convolution"
  bottom: "data"
  top: "conv0"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 32
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv0/relu"
  type: "ReLU"
  bottom: "conv0"
  top: "conv0"
}
layer {
  name: "conv1/dw"
```


```
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 32
    pad: 1
    kernel_size: 3
    group: 32
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv1/dw/relu"
  type: "ReLU"
  bottom: "conv1/dw"
  top: "conv1/dw"
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "conv1/dw"
  top: "conv1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 64
    kernel_size: 1
    weight_filler {
      type: "msra"
```


ler {
'constant"

```
      value: 0.0
    }
  }
}
layer {
  name: "conv1/relu"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "conv2/dw"
  type: "Convolution"
  bottom: "conv1"
  top: "conv2/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 64
    pad: 1
    kernel_size: 3
    stride: 2
    group: 64
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv2/dw/relu"
  type: "ReLU"
  bottom: "conv2/dw"
  top: "conv2/dw"
}
layer {
  name: "conv2"
        onvolution"
        'conv2/dw"
        v2"
```

```
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv2/relu"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "conv3/dw"
  type: "Convolution"
  bottom: "conv2"
  top: "conv3/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    pad: 1
    kernel_size: 3
    group: 128
    engine: CAFFE
    weight_filler {
      type: "msra"
```



```
ler {
'constant"
```

```
    value: 0.0
    }
  }
}
layer {
  name: "conv3/dw/relu"
  type: "ReLU"
  bottom: "conv3/dw"
  top: "conv3/dw"
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "conv3/dw"
  top: "conv3"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv3/relu"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4/dw"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4/dw"
  param {
    : 1.0
    ult: 1.0
```

```
 order: 2
   order: 3
   order: 1
  }
}
layer {
  name: "conv14_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv14_2_mbox_conf_perm"
  top: "conv14_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv14_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv14_2"
  bottom: "data"
  top: "conv14_2_mbox_priorbox"
  prior_box_param {
    min_size: 150.0
    max_size: 195.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv15_2_mbox_loc"
  type: "Convolution"
  bottom: "conv15_2"
  top: "conv15_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
          nult: 0.0

          ion_param {
```

```
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv15_2_mbox_loc"
  top: "conv15_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv15_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv15_2_mbox_loc_perm"
  top: "conv15_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv15_2_mbox_conf"
  type: "Convolution"
  bottom: "conv15_2"
  top: "conv15_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
        size: 1
    _fille {
    'msra'
```

```
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv15_2_mbox_conf"
  top: "conv15_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv15_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv15_2_mbox_conf_perm"
  top: "conv15_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv15_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv15_2"
  bottom: "data"
  top: "conv15_2_mbox_priorbox"
  prior_box_param {
    min_size: 195.0
    max_size: 240.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
```


onv15_2_mbox_lcc"

```
  type: "Convolution"
  bottom: "conv16_2"
  top: "conv16_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv16_2_mbox_loc"
  top: "conv16_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv16_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv16_2_mbox_loc_perm"
  top: "conv16_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv16_2_mbox_conf"
  type: "Convolution"
        'conv16_2"
        v16_2_mbox_conf"
```

```
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv16_2_mbox_conf"
  top: "conv16_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv16_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv16_2_mbox_conf_perm"
  top: "conv16_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv16_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv16_2"
  bottom: "data"
  top: "conv16_2_mbox_priorbox"
  prior_box_param {
    :e: 240.0
    ze: 285.0
    ratio: 2.0
```

```
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv17_2_mbox_loc"
  type: "Convolution"
  bottom: "conv17_2"
  top: "conv17_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv17_2_mbox_loc"
  top: "conv17_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
```


onv17_2_mbox_loc_flat"
atten"

```
  bottom: "conv17_2_mbox_loc_perm"
  top: "conv17_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv17_2_mbox_conf"
  type: "Convolution"
  bottom: "conv17_2"
  top: "conv17_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv17_2_mbox_conf"
  top: "conv17_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv17_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv17_2_mbox_conf_perm"
          v17_2_mbox_conf_flat"
          aram {
```

```
  }
 }
 layer {
  name: "conv17_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv17_2"
  bottom: "data"
  top: "conv17_2_mbox_priorbox"
  prior_box_param {
   min_size: 285.0
   max_size: 300.0
   aspect_ratio: 2.0
   aspect_ratio: 3.0
   flip: true
   clip: false
   variance: 0.1
   variance: 0.1
   variance: 0.2
   variance: 0.2
   offset: 0.5
  }
 }
 layer {
  name: "mbox_loc"
  type: "Concat"
  bottom: "conv11_mbox_loc_flat"
  bottom: "conv13_mbox_loc_flat"
  bottom: "conv14_2_mbox_loc_flat"
  bottom: "conv15_2_mbox_loc_flat"
  bottom: "conv16_2_mbox_loc_flat"
  bottom: "conv17_2_mbox_loc_flat"
  top: "mbox_loc"
  concat_param {
   axis: 1
  }
 }
 layer {
  name: "mbox_conf"
  type: "Concat"
  bottom: "conv11_mbox_conf_flat"
  bottom: "conv13_mbox_conf_flat"
  bottom: "conv14_2_mbox_conf_flat"
  bottom: "conv15_2_mbox_conf_flat"
  bottom: "conv16_2_mbox_conf_flat"
  bottom: "conv17_2_mbox_conf_flat"
  top: "mbox_conf"
  concat_param {
```

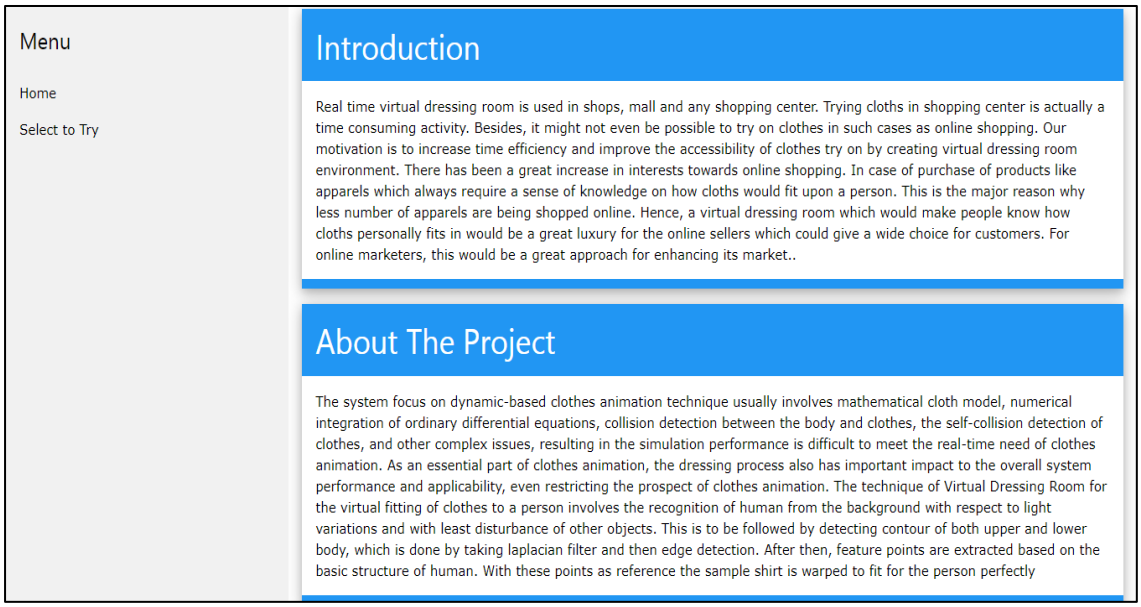
```
layer {
  name: "mbox_priorbox"
  type: "Concat"
  bottom: "conv11_mbox_priorbox"
  bottom: "conv13_mbox_priorbox"
  bottom: "conv14_2_mbox_priorbox"
  bottom: "conv15_2_mbox_priorbox"
  bottom: "conv16_2_mbox_priorbox"
  bottom: "conv17_2_mbox_priorbox"
  top: "mbox_priorbox"
  concat_param {
    axis: 2
  }
}
layer {
  name: "mbox_conf_reshape"
  type: "Reshape"
  bottom: "mbox_conf"
  top: "mbox_conf_reshape"
  reshape_param {
    shape {
      dim: 0
      dim: -1
      dim: 21
    }
  }
}
layer {
  name: "mbox_conf_softmax"
  type: "Softmax"
  bottom: "mbox_conf_reshape"
  top: "mbox_conf_softmax"
  softmax_param {
    axis: 2
  }
}
layer {
  name: "mbox_conf_flatten"
  type: "Flatten"
  bottom: "mbox_conf_softmax"
  top: "mbox_conf_flatten"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "detection_out"
    etectionOutput"
    'mbo _loc"
    'mbox_conf_flatten"
```

```
bottom: "mbox_priorbox"
top: "detection_out"
include {
  phase: TEST
}
detection_output_param {
  num_classes: 21
  share_location: true
  background_label_id: 0
  nms_param {
    nms_threshold: 0.45
    top_k: 100
  }
  code_type: CENTER_SIZE
  keep_top_k: 100
  confidence_threshold: 0.25
}
}
```

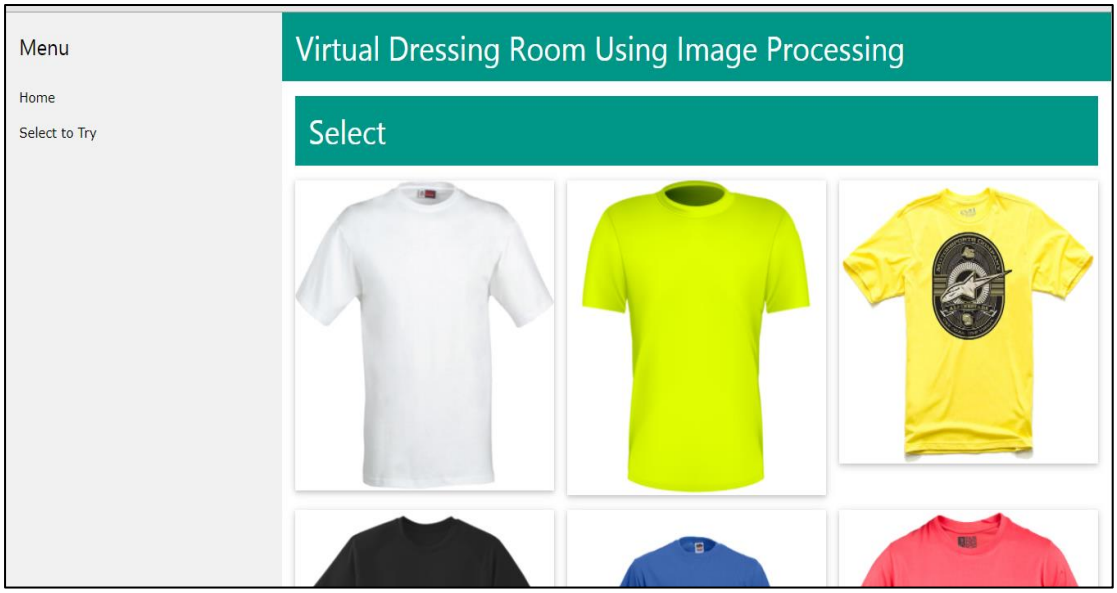

RESULT AND APPLICATION

❖ Screenshots of Project

Home page of Virtual Dressing Room



More about the system in the home page



Options of clothes to try on with.

3rd T-Shirt imposed on the user.

2nd T-Shirt imposed on the user.

CONCLUSION AND FUTURE SCOPE

Conclusion

A common problem faced by customers while shopping for clothes is the need to spend hours trying out a variety of clothes physically. This can be tiring and the time available might be short. The proposed solution to overcome this problem is the use of a Virtual Mirror that acts as a virtual trial room. It uses a camera and DNN(deep neural network) to detect things (objects other than body). Then node points of the human body are plotted and this data is then used to render the image of clothes over the user's body, thereby eliminating the need to physically try on the clothes and hence helping save time.

Thus with constant research and numerous iterations, a working prototype of the system was developed well enough to create a virtual room for the users to allow them to virtual tryout the selected outfit. In this study, a virtual mirror system is designed for the purpose of the clothing changing room. Our motivation here is to increase the time efficiency and improve the accessibility of clothes try-on by creating a virtual dressing room environment. The system contains two basic male and female models of which joints are over 200. They have the spectacular potential of performing human movements as well as facial expressions. For body size and height, we created a lot of model variations. The GUI of changing-room reads and interprets the data arrived from keyboard, mouse, webcam or Kinect input units and enables users to try garments and shoes on a created humanoid model.

Future Enhancements

The Virtual mirror project proposed here allows the customers to choose the clothes of their choice by using a tablet which is then identified by the mirror. This project can be improvised by including gesture recognition for selection of clothes, thereby eliminating the need of an external device to select clothes.

The Project works currently for male garments like T-shirt (full sleeve, half sleeve), Pants and Shorts. The Project can be improvised to include female garments also like one piece dresses, traditional dresses.