ENSC180 - Group Project Report

1. Introduction

One of the main problems with online shopping is that the products can not be checked directly. Customers always need to wait until the product arrives before they can try it, and so while ordering, it is impossible for them to know if it will be suitable for them. To solve this issue, we thought of developing an application that allows its users to try the products before buying them. Our objective is to produce a realistic simulation of how a garment or accessory looks on the user, which could be achieved using augmented reality. For the application's prototype, we decided to focus on fitting a wristband on the user's wrist. The application is divided into three main phases. The first one is the hand recognition stage, where the hand area is segmented from the background. Then, that area is used as the anchor for the localization stage, where the main points of the hand are located and used to track both the hand and the wrist. Finally, necessary metrics like position and scaling factor will be computed from the main points and used to place the wristband correctly and display it to the user.

2. Background

In the last decade, virtual reality is one of the fields that have grown with an astonishing speed and impacted several industries. As one of the consequences of this rapid growth, the term augmented reality was created to tackle the need of applying digital objects to real environments. In recent years, research on augmented reality has been making huge progress, and several applications have been developed to take advantage of this technology. Social networks such as

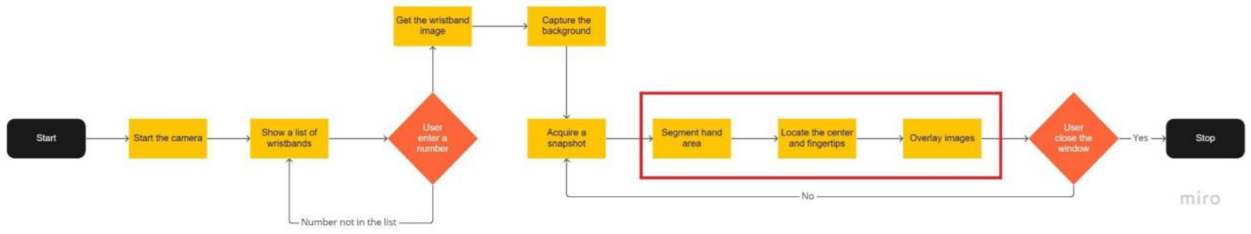
Facebook and Snapchat have been using augmented reality in their camera function to enrich their users' experience.

For a long time, there have been several attempts to localize face landmarks and use them for augmented reality applications. The accuracies of the state-of-the-art methods are remarkably high and the number of landmarks detected can be up to more than 100. As a consequence, several models have been exported for productional uses and a large number of applications have been developed using those models as the core. However, the research on hand landmarks localization has not reached that level yet. So far only the model Mediapipe of Google performs well enough for commercial uses. In Matlab, there are several approaches to segment hands from an image background and detect the gesture. Nevertheless, only an extremely few of them try to extract hand landmarks from the image, and almost none of the approaches is used in a situation where they have to process several images acquired from a continuous camera stream in a short amount of time.

Our objective is to create an augmented reality application to try wristbands in Matlab, which has never been done before. In addition to combining different methods to locate hand landmarks and testing them with continuous camera images, a new method is needed to calculate the position to place the wristband, as well as its rotation and scaling factors.

3. Methodology

As illustrated in the flowchart, our application comprises three major stages: hand recognition, hand landmarks localization and images overlaying. After the camera stream is initialized, the user chooses a wristband among those that are plotted at the beginning and it will be stored to be used in the overlaying stage. After that, snapshots are acquired, processed and plotted repeatedly using a loop. In each iteration, three stages are applied to the acquired snapshot and result in a processed image with the wristband in the correct position, with the correct rotation and scaling factors. As the images are processed almost instantly and the resulted images are displayed repeatedly, the user can have an experience similar to trying wristbands in real-time.



3.1. Hand Recognition

To segment the hand from the background, we tried three different methods: colours extraction, Viola-Jones (or Haar Cascade) algorithm and background subtraction. While HSV colours extraction is the most effective method when segmenting a single image, it is not suitable to be used for our application because it depends significantly on the lighting conditions and performs poorly if the hand is moving. The Viola-Jones technique is the easiest one to implement if the Haar Cascade files are available and it does not require a specific lighting condition. However, all the implementations using different Haar Cascade files that we found were unstable if the hand was not closed and could not detect the hand most of the time if the contrast between the hand and the background is not much. The background subtraction method relies greatly on a pre-acquired background image and requires that the background must remain the same even when hands are included in the image. This does not pose a problem since the user most probably stays in one place and only moves their hand while using our application.

The background image is acquired immediately after the user chooses a wristband. At the beginning of each iteration, a snapshot is acquired and subtracted from the background. It is then converted to a grayscale image to find the threshold between the objects and the empty background. The result is converted to a binary image, and then the objects defined by the threshold that has fewer than 10000 pixels are removed. Erosion, dilation and median filtering are then applied to the binary image in that exact order to remove the remaining noises, and then the objects that have fewer than 10000 pixels are removed again.

3.2. Hand Landmarks Localization

After the hand area is acquired from the previous stage, the centroids and boundaries of the remaining objects are extracted. For each object, every point from the boundary set is used to find the cartesian coordinates of the local maximas. If a hand exists in the snapshot and no problem occurs in the previous steps, at this point, there should be only one object and one to five maximas, which is the hand and the detected fingertips.

3.3. Images Overlaying

If a hand is detected, the landmarks extracted in the previous stage will be used to calculate the position, rotation and scaling factors of the wristband that will be placed. The direction vector is computed by subtracting the third maxima from the centroid. The position to place the wristband is computed by multiplying the direction vector with a suitable factor and adding the result to the centroid. The distance between the third maxima and the centroid, which is the norm of the direction vector, is then divided by the standard distance measured when the hand fitted the wristband perfectly without rescaling to get the scaling factor. The angle between the direction vector and y-axis is computed using the arctangent formula. Coordinates of every pixel in the processed image that is not zero in all channels are extracted and the corresponding coordinates in the snapshot are replaced with them, with the center being the precalculated position.

4. Results

In our final version, all of the intended core functions were successfully demonstrated. The program was able to take snapshots from the camera stream, segment the hand area, locate important landmarks, calculate the factors needed to place the wristband, and place the wristband correctly. Although the result was still a little unstable, and it did not perform in most general cases as well as in situations where the lighting condition was perfect and there was a large contrast between the background and the hand, it was satisfactory enough for our purpose, which is to use as an augmented reality application to try wristbands.

5. Recommendations and Conclusions

In this work, we present a simple approach to the jewelry virtual try-on problem. Although the performance is to our expectations, further improvements can be made. For the hand segmentation part, we used a pure computer vision approach and thus, the result still somewhat depends on the background, the lighting conditions and the behaviours of the user. Our method of locating hand landmarks is still crude, hence it is only able to locate the center of the hand and the fingertips. Both parts can be improved remarkably by using a deep learning model to localize hand landmarks. Hand segmentation can be performed independently of environmental conditions, and several important landmarks on the fingers and the palm can be located accurately with minimum delay.

6. Appendices

6.1. Requirements

- Suitable support package for the webcam (Image Acquisition Toolbox Support Package for OS Generic Video Interface).
- Image Acquisition Toolbox.
- Signal Processing Toolbox.
- Image Processing Toolbox.

6.2. Images used in the demonstration

https://drive.google.com/drive/folders/1GniaItr3lsC0zNKO67GbxTYHX_4PoRkR?usp=sharing

6.3. Code

```
clear all; clc; close all;                    %clear stored variables, command windows, close any open windows
objects = imaqfind;                            %find video input objects in memory
delete(objects);


%config camera stream
vid=videoinput('winvideo',1);
set(vid,'ReturnedColorspace','rgb')
triggerconfig(vid, 'manual');
start(vid);


while true
    figure(1);
    for idx=1:6
        subplot(2, 3, idx);
        I=imread(string(idx) + ".png");

        M = repmat(all(~I,3),[1 1 3]); %mask black parts

        I(M) = 255; %turn them white

        imshow(I);

        title(idx);

    end

    sgtitle('Please pick a wristband');


    inp = input("Enter a number from 1-6, 0 to exit: ");

    while inp < 0 || inp > 6

        inp = input("Please enter a number from 1-6: ");

    end

    if inp == 0

    break

    end

    IM1=getsnapshot(vid);                                   %get snapshot from the webcam video and
store to IM1 variable


    orgwb = imread(string(inp) + ".png");

    wb=orgwb;

    fgh = figure(2);

    while true

        if ~ishghandle(fgh)
```

```
            break
        end
        IM2=getsnapshot(vid);                    %get snapshot of test image and store to
variable IM2



        IM3 = IM1 - IM2;                                  %subtract Backround from Image
        IM3 = rgb2gray(IM3);                               %Converts RGB to Gray
        lvl = graythresh(IM3);                            %find the threshold value using
Otsu's method for black and white

        IM3 = im2bw(IM3, lvl);                             %Converts image to BW, pixels
with value higher than threshold value is changed to 1, lower changed to 0
        IM3 = bwareaopen(IM3, 10000);
        IM3 = imfill(IM3,'holes');
        IM3 = imerode(IM3,strel('disk',15));                 %erode image
        IM3 = imdilate(IM3,strel('disk',20));                %dilate iamge
        IM3 = medfilt2(IM3, [5 5]);                          %median filtering
        IM3 = bwareaopen(IM3, 10000);                          %finds objects, noise or
regions with pixel area lower than 10,000 and removes them

        REG=regionprops(IM3,'all');                        %calculate the properties of
regions for objects found
```

```
    CEN = cat(1, REG.Centroid);                         %calculate Centroid
    [B, L, N, A] = bwboundaries(IM3,'noholes');                 %returns the number of
objects (N), adjacency matrix A, object boundaries B, nonnegative integers of contiguous
regions L

    for k =1:length(B)                                  %for the given object k
      BND = B{k};                                       %boundary set for object
      BNDx = BND(:,2);                                    %Boundary x coord
      BNDy = BND(:,1);                                    %Boundary y coord

      if (length(B) == 1)
        pkoffset = CEN(:,2);                             %Calculate peak offset point from
centroid
        [pks,locs] = findpeaks(-BNDy,'minpeakheight',-pkoffset);        %find peaks in the
boundary in y axis with a minimum height greater than the peak offset
        n_X = BNDx(locs);
      end

    end
    if length(CEN) > 1
      try
      centroids = round(CEN);
  %       start = centroids(1, :);
```

```
if exist('pks','var') && length(pks) > 3
    dvec = [n_X(3), -pks(3)] - centroids(1, :);
    start = dvec * -0.5 + centroids(1, :);

    distance = norm(dvec);
    wb = imresize(orgwb, distance / 264);

    %CosTheta = max(min(dot(dvec, [0, -1])/norm(dvec),1),-1);
    %angle = real(acosd(CosTheta));
    % a = atan2d(x1*y2-y1*x2,x1*x2+y1*y2);
    angle = atan2d(-dvec(1), -dvec(2));
    wb = imrotate(wb,angle,'bilinear','crop');
end
if ~exist('start','var')
    start = centroids(1, :);
end
ystart = start(2) - fix(size(wb, 1) / 2);
xstart = start(1) - fix(size(wb, 2) / 2);

rmask = find(wb(:, :, 1));
gmask = find(wb(:, :, 2));
bmask = find(wb(:, :, 3));
wbmask = union(rmask, gmask);
```

```
            wbmask = union(wbmask, bmask);

            xmask = fix(wbmask/size(wb, 1));

            ymask = rem(wbmask, size(wb, 1));




            catch

            end

        end

        if exist('ystart','var')

            try

            %IM2(ystart + ymask, xstart + xmask,:) = wb(ymask, xmask+1,:);

            for cc=0:2

                cmask = cc * size(IM2, 1) * size(IM2, 2) + (xstart + xmask) * size(IM2, 1) + ystart +
ymask;

                nwbmask = cc * size(wb, 1) * size(wb, 2) + wbmask;

                IM2(cmask) = wb(nwbmask);

            end

            catch

            end

        end

        figure(2);imshow(IM2);

    end

end
```

```
delete(vid);
```