Keyboard Projection And Detection

Filipe Gonçalves, 98083, 60%
Paulo Pereira, 98430, 40%

TABLE OF CONTENTS

01

The Idea

What we wanted to accomplish

02

The Solution

What was possible to build in such a short time

03

Difficulties

Obstacles we hit and workarounds

04

What’s next ?

Work to be done next as the project never stops

01

The Idea

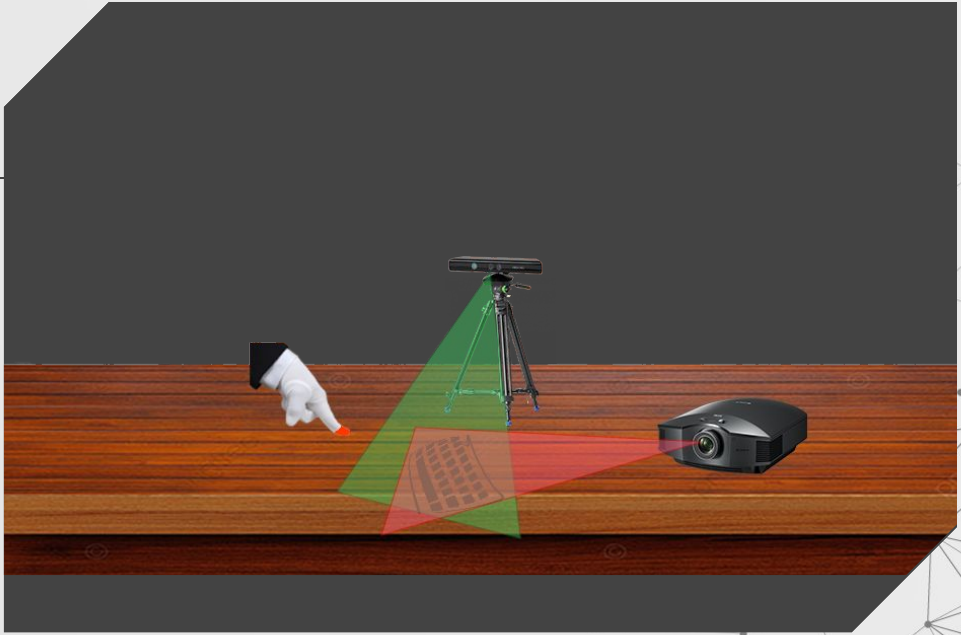
What we intended to build

The Idea

Project the keyboard onto the table

Kinect Camera to detect depth and record the keyboard

Glove with red point for finger detection

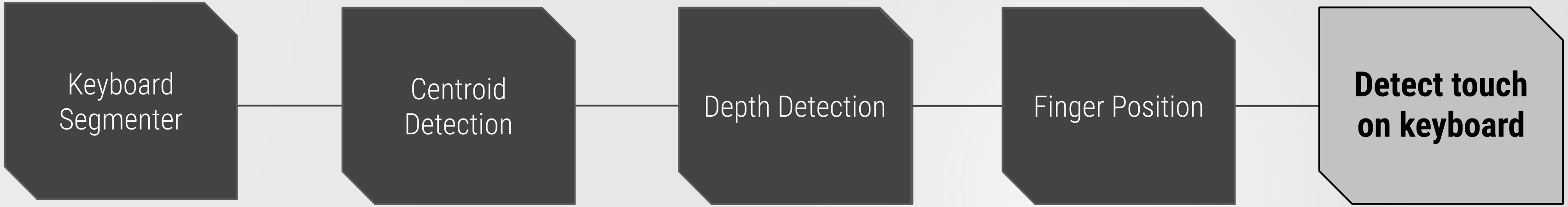


02

The Solution

What was accomplished in the short time span that was given

Implementation



Keyboard Segmenter

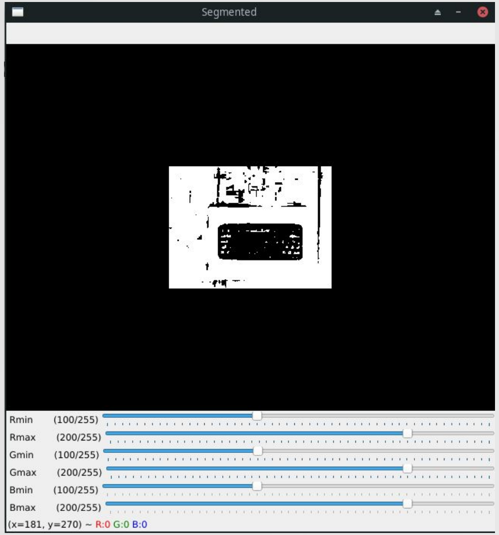
Centroid Detection

Depth Detection

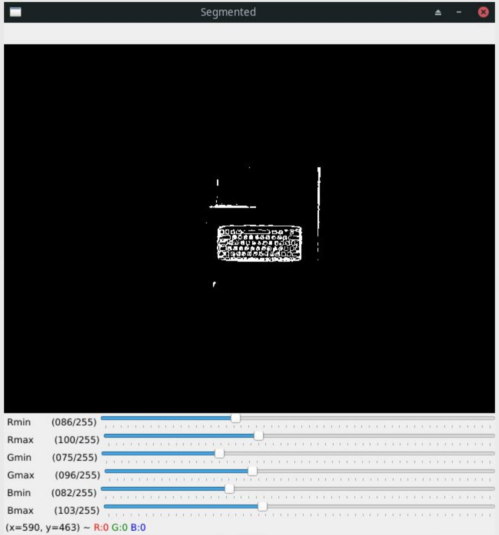
Finger Position

Detect touch on keyboard

Keyboard Segmenter



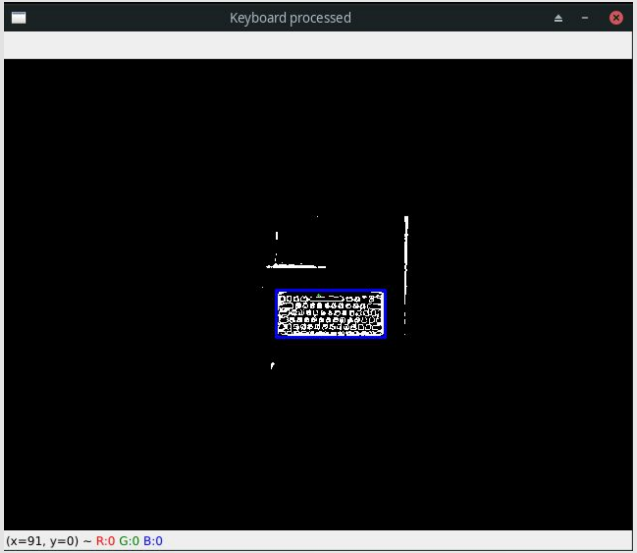
```
Rmin   (100/255)
Rmax   (200/255)
Gmin   (100/255)
Gmax   (200/255)
Bmin   (100/255)
Bmax   (200/255)
(x=181, y=270) ~ R:0 G:0 B:0
```



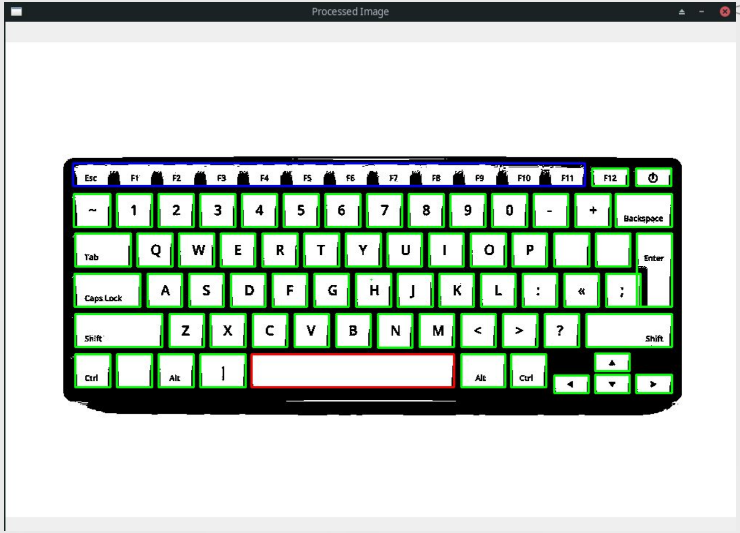
```
Rmin   (086/255)
Rmax   (100/255)
Gmin   (075/255)
Gmax   (096/255)
Bmin   (082/255)
Bmax   (103/255)
(x=590, y=463) ~ R:0 G:0 B:0
```

- Mask with the range values selected
- Logical_not the mask in the image
- Threshold all values different than 0 to be 1
- Save the range to use in the main file

Centroid Detection

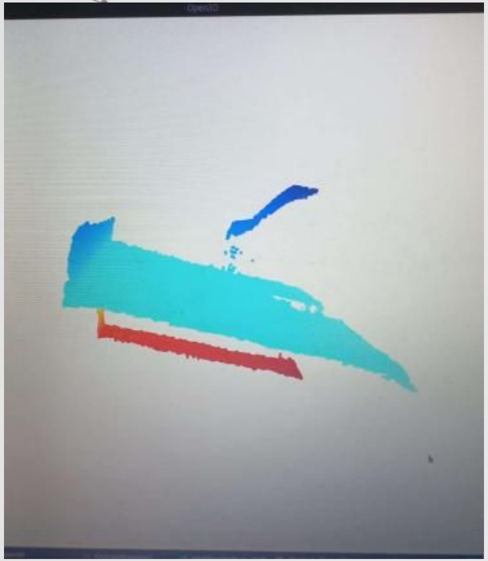


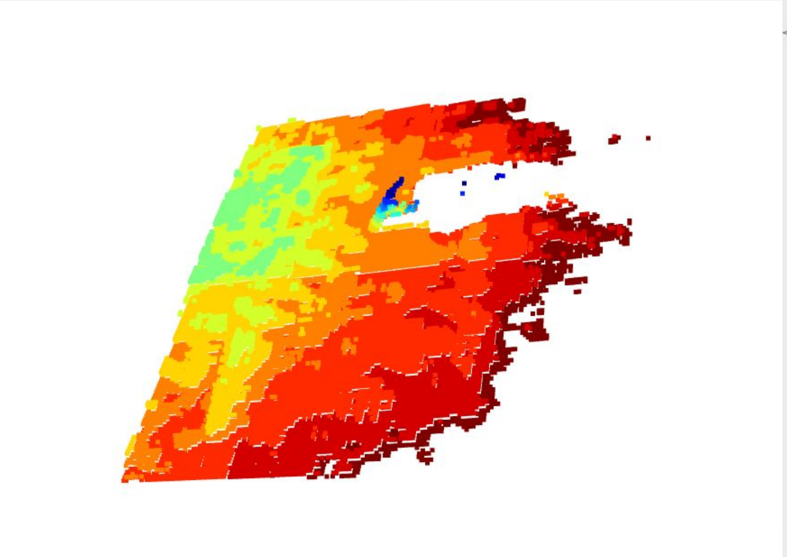
(x=91, y=0) ~ R:0 G:0 B:0


Processed Image
Esc F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
~ 1 2 3 4 5 6 7 8 9 0 - + Backspace
Tab Q W E R T Y U I O P Enter
Caps Lock A S D F G H J K L : « ;
Shift Z X C V B N M < > ? Shift
Ctrl Alt Alt Ctrl

- Mask the image to only look into a small rectangle in the middle of the image
- Process the image with the range values segmented before
- cv2.connectedComponentsWithStats to get the centroids
- Biggest centroid after the whole image is the keyboard

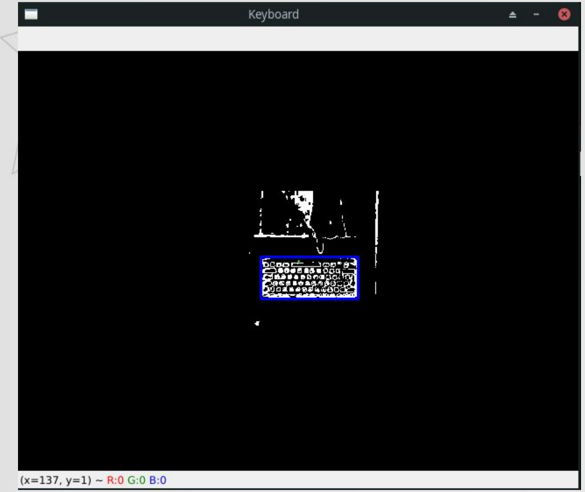
Depth Detection



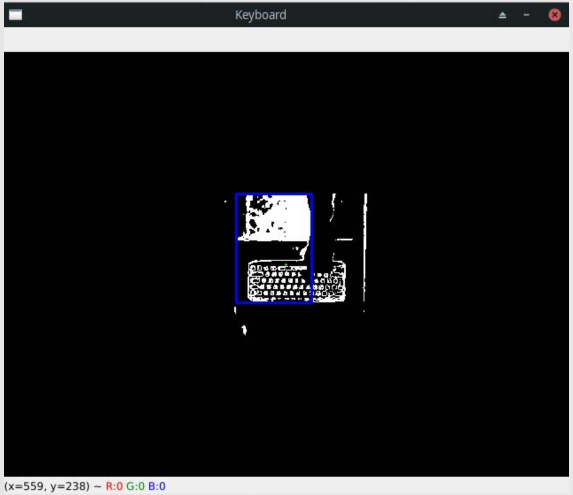


- Filtering point clouds to get the base of the board
- Detect when there is a big “shadow”, as the hand is starting to move to the keyboard
- Detect when the hand is close enough to the board to deduce touch

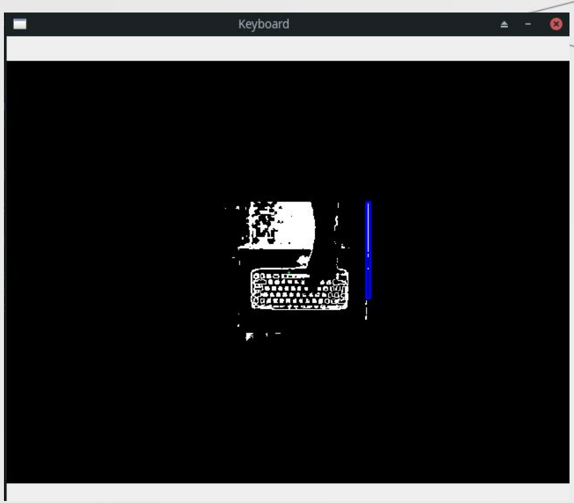
Finger Position



(x=137, y=1) ~ R:0 G:0 B:0



(x=559, y=238) ~ R:0 G:0 B:0



- Process the frame only if there was touching deduced
- Identify the biggest centroid
- If the keyboard calculated is different from the calibrated (points are different, as area is always different because of the segmenter), there was touch!

03

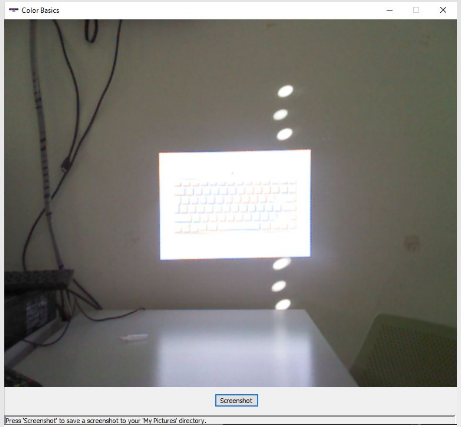
Difficulties

Obstacles we came across and how we worked around them

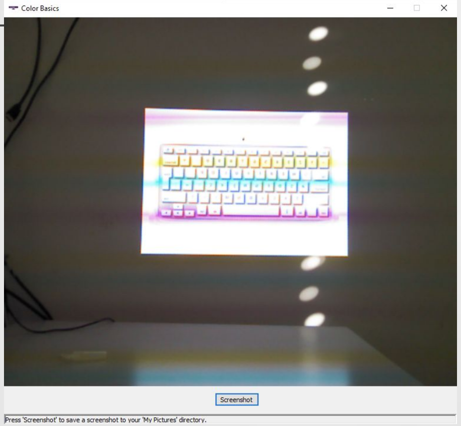
Difficulties

Refresh Rates

The projector and the Kinect camera both have different refresh rates and the video outputting from the camera had some color problems



Press 'Screenshot' to save a screenshot to your 'My Pictures' directory.



Press 'Screenshot' to save a screenshot to your 'My Pictures' directory.

Difficulties

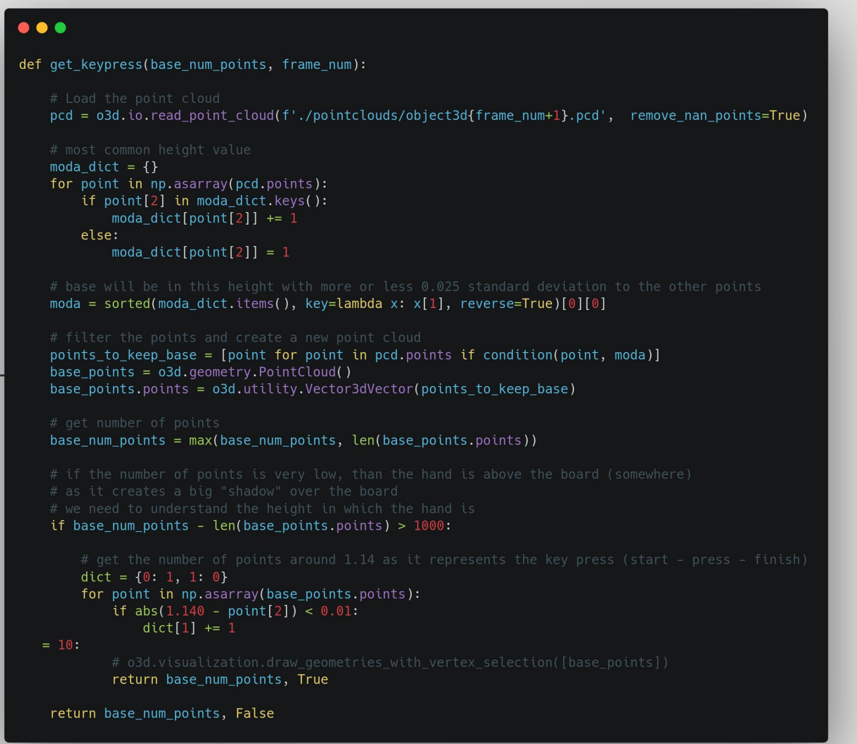
Camera Setup

For a static image we needed to have the kinect fixed in a point, and the easiest mode was to use the safety blocks of the kinect as tripe and record a keyboard in the ceiling

Difficulties

Contact detection

Filtering the point clouds, for the base board, as we don’t have that much knowledge on the subject, became quite difficult

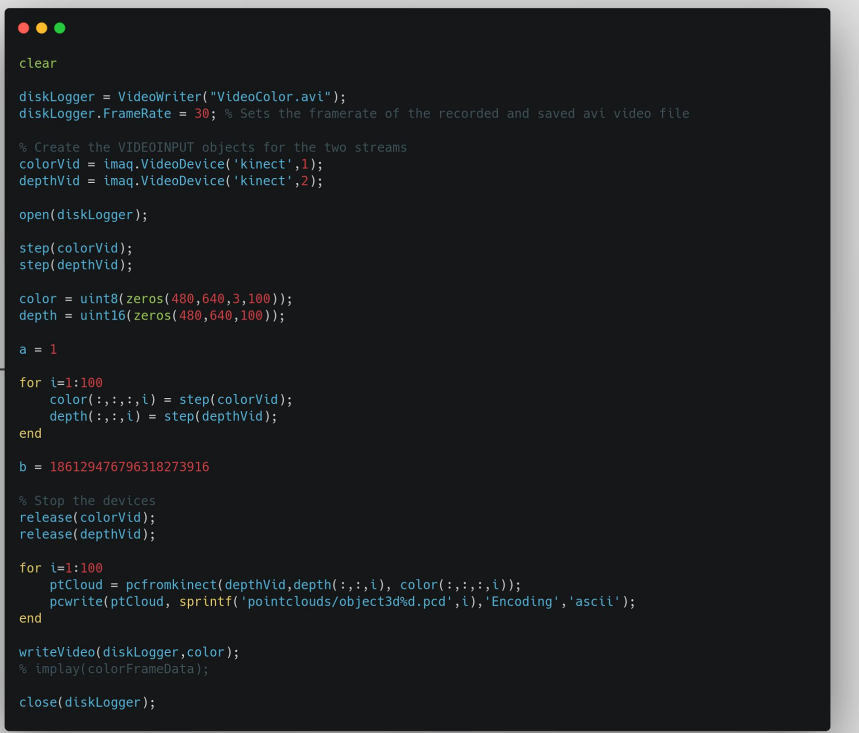


```
def get_keypress(base_num_points, frame_num):

    # Load the point cloud
    pcd = o3d.io.read_point_cloud(f'./pointclouds/object3d{frame_num+1}.pcd',  remove_nan_points=True)

    # most common height value
    moda_dict = {}
    for point in np.asarray(pcd.points):
        if point[2] in moda_dict.keys():
            moda_dict[point[2]] += 1
        else:
            moda_dict[point[2]] = 1

    # base will be in this height with more or less 0.025 standard deviation to the other points
    moda = sorted(moda_dict.items(), key=lambda x: x[1], reverse=True)[0][0]

    # filter the points and create a new point cloud
    points_to_keep_base = [point for point in pcd.points if condition(point, moda)]
    base_points = o3d.geometry.PointCloud()
    base_points.points = o3d.utility.Vector3dVector(points_to_keep_base)

    # get number of points
    base_num_points = max(base_num_points, len(base_points.points))

    # if the number of points is very low, than the hand is above the board (somewhere)
    # as it creates a big "shadow" over the board
    # we need to understand the height in which the hand is
    if base_num_points - len(base_points.points) > 1000:

        # get the number of points around 1.14 as it represents the key press (start - press - finish)
        dict = {0: 1, 1: 0}
        for point in np.asarray(base_points.points):
            if abs(1.140 - point[2]) < 0.01:
                dict[1] += 1
   = 10:
            # o3d.visualization.draw_geometries_with_vertex_selection([base_points])
            return base_num_points, True

    return base_num_points, False
```

Difficulties

Kinect Connection

The kinect camera only works in windows, and connect it to python is a difficult job, so we decided to record video and depth point clouds using Matlab and utilize them in the python code created prior



```
clear

diskLogger = VideoWriter("VideoColor.avi");
diskLogger.FrameRate = 30; % Sets the framerate of the recorded and saved avi video file

% Create the VIDEOINPUT objects for the two streams
colorVid = imaq.VideoDevice('kinect',1);
depthVid = imaq.VideoDevice('kinect',2);

open(diskLogger);

step(colorVid);
step(depthVid);

color = uint8(zeros(480,640,3,100));
depth = uint16(zeros(480,640,100));

a = 1

for i=1:100
    color(:,:,:,i) = step(colorVid);
    depth(:,:,i) = step(depthVid);
end

b = 186129476796318273916

% Stop the devices
release(colorVid);
release(depthVid);

for i=1:100
    ptCloud = pcfromkinect(depthVid,depth(:,:,i), color(:,:,:,i));
    pcwrite(ptCloud, sprintf('pointclouds/object3d%d.pcd',i),'Encoding','ascii');
end

writeVideo(diskLogger,color);
% implay(colorFrameData);

close(diskLogger);
```

04

What’s Next ?

Next steps to finish the idea

What’s Next

Camera

Better Kinect setup and video streaming directly to the code

Use the Projector to project the keyboard

Projector

Calibration

Improve the settings and detection ranges for the hand touching the board as well as the hand above the keyboard

THANKS

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.

Please keep this slide for attribution.