Alma Mater Studiorum – Università di Bologna

Corso di laurea magistrale in

BIOMEDICAL ENGINEERING

Tesi di Laurea in

AGEING AND REHABILITATION ENGINEERING

TITOLO

Landmarks recognition and body reconstruction through an OpenCV AI Kit device

CANDIDATO

Daniele Paolucci

RELATORE

Lorenzo Chiari

CORRELATORE

Moreno D'Amico

Appello 14/03/24

Sessione Anno Accademico 2022/2023

Contents

INTRODUCTION

The process of data acquisition is the basis of biomedicine. It is often time-demanding and requires the subject to perform tasks in a lab and to wear sensors. Applying these sensors to the subject requires time, trained personnel, and a laboratory or specialised location. One example is human movement's reconstruction through stereo-photogrammetry. This method is considered the gold standard of movement analysis but requires more than one specialised camera and some time to prepare the subject for the acquisition (markers placement). Although it is an exceptionally reliable method, it presents some limitations: it is time-consuming, it is not available to everybody, it requires a lab, and it must be performed in the presence of qualified personnel.

To improve data collection, we must look for reliable instrumentation that can be placed everywhere, does not require much time to set up, and can be quickly started by the user himself.

Given the spread of device cameras capable of registering 3D information, the asserting of the AI, and the general desire for real-time stream of information processing, oak-d devices seem to answer all this demand. We are talking about a low price, compact camera, easy to program and high-performance devices that are ideal for carrying around and experimenting with.

This project aims to prove that a marker-less 3D landmark acquisition through the exploitation of an end-to-end pipeline, the improvement of already-coded programs from GitHub, and the development of a calibration protocol is not only possible but also valuable.

In particular, the goals of this thesis are:

- Establish a calibration protocol for the cameras.
- Obtain intrinsic and extrinsic parameters.
- Acquiring and processing a point cloud that represents and adheres to the subject figure.

- Find the best work conditions for an end-to-end pipeline.
- Camera’s parameters tuning.
- Register and process the landmarks’ displacement.
- Explore and exploit the camera’s features to enhance the performance of all the points above.

More generally, one of the objectives is to highlight the device's functionality, strong suits and limitations. Moreover, gathering all the information about the device and integrating them with field tests can give future users a better overview of the oak-d’s capabilities.

Lastly, this paper intends to propose a new method to morph the point cloud acquired into a human mesh for a better reconstruction and simulation of missing points.

1. DEVICES

1.1 Hardware overview

Two types of devices were used:

- 1 OAK-D: The baseboard has three on-board cameras that implement stereo and RGB vision and are piped directly into the OAK SoM (System on Module) for depth and AI processing. The data is then output to a host via USB 3.1 Gen1 (Type-C). This OAK camera uses a USB-C cable for communication and power. It supports both USB2 and USB3 (5 GB or 10 GB).
  Price= $249
  The Main features are:
  - OAK-D is 4 TOPS of processing power (1.4 TOPS for AI-RVC2 (robotic vision)
  - Run any AI model, even custom-architecture/built ones - models need to be converted.
  - Encoding: H.264, H.265, MJPEG - 4K/30FPS, 1080P/60FPS
  - Computer vision (CV): wrap/de-wrap, resize, crop via Image Manipulation node, edge detection, feature tracking. You can also run custom CV functions.
  - Stereo depth perception with filtering, post-processing, RGB-depth alignment, and high configurability
  - Object tracking: 2D and 3D tracking with ObjectTracker node.
  - Integrated BNO085, a 9-axis IMU

Figure 1 The OAK-D device.

Camera Specs	**Colour camera**	**Stereo pair**
Sensor	IMX378 (PY004 AF, PY052 FF)	OV9282 (PY003)
DFOV / HFOV / VFOV	81° / 69° / 55°	89° / 80° / 55°
Resolution	12MP (4056×3040)	1MP (1280×800)
Focus	**AF: 8 cm – ∞ or FF: 50 cm – ∞**	**FF: 19.6 cm – ∞**
Max Frame rate	60 FPS	120 FPS
F-number	1.8 ±5%	2.0 ±5%

Lens size	1/2.3 inch	1/4 inch
Effective Focal Length	4.81 mm	2.35 mm
Pixel size	1.55 µm x 1.55 µm	3 µm x 3 µm

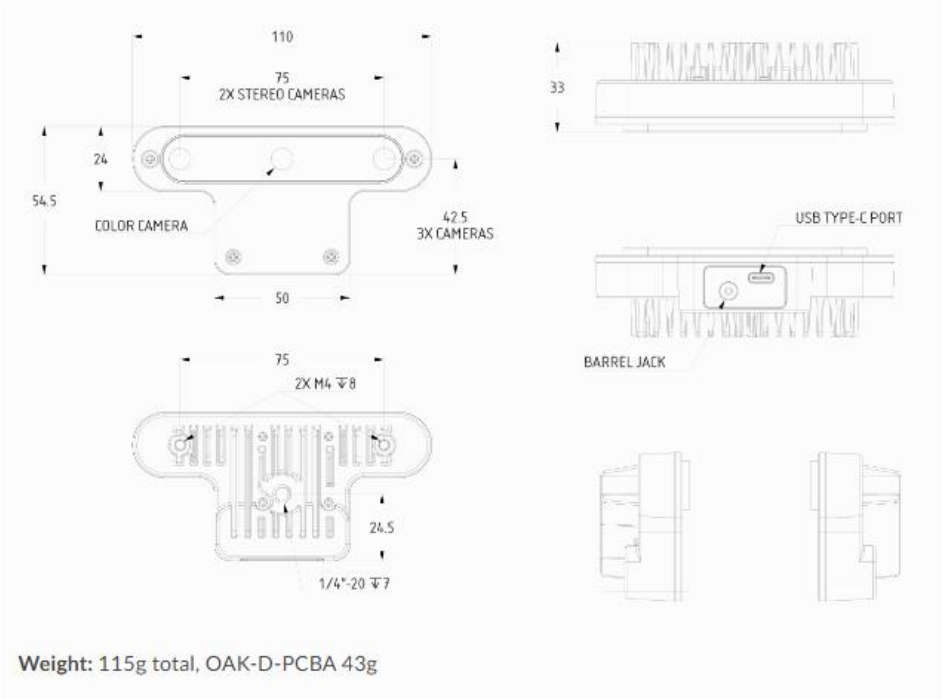

Figure 2 OAK-D device layout and dimensions

This OAK camera has a baseline of 7.5 cm - the distance between the left and the right stereo camera. Minimal and maximal depth perception (MinZ and Max) depends on camera FOV, resolution, and baseline.

- Ideal depth range: 70 cm - 12m
- MinZ: ~20 cm (400P and extended), ~35 cm (400P OR 800P, extended), ~70 cm (800P)

- Depth accuracy:
    - 0.7m - 4m: below 1% absolute depth error
    - 4m - 7m: below 2% absolute depth error
    - 7m - 12m: below 3% absolute depth error
- Operative temperature between -20° and 50°.

Most of the power is consumed by the RVC2, so the power consumption mostly depends on the workload of the VPU:

- Base consumption + camera streaming: 2.5W - 3W
- AI subsystem consumption: Up to 1W
- Stereo depth pipeline subsystem: Up to 0.5W
- Video Encoder subsystem: Up to 0.5W

The total power consumption can be up to ~5W if you are using all the features at 100% at the same time. To reduce the power consumption, you can reduce FPS of the whole pipeline - that way, subsystems will not be utilized at 100% and will consume less power. In this way, it is not necessary to plug the power supply cable since USB supplies enough power.

- 2 OAK-D-LITE: same Spatial AI functionality as OAK-D with a smaller weight and form factor. It supports the same USB as OAK-D.
  Price: $149
  The key features are:
  - 4 TOPS of processing power (1.4 TOPS for AI - RVC2 NN Performance)
  - Run any AI model, even custom-architecture/built ones - models need to be converted.

- Encoding: H.264, H.265, MJPEG - 4K/1080P
- Computer vision: wrap/de-wrap, resize, crop via ImageManip node, edge detection, feature tracking. You can also run custom CV functions.
- Stereo depth perception with filtering, post-processing, RGB-depth alignment, and high configurability
- Object tracking: 2D and 3D tracking with ObjectTracker node.

Figure 3 OAKD- lite device

- Width: 91 mm
- Height: 28 mm
- Length: 17.5 mm
- Baseline: 75 mm
- Weight: 61 g

Camera Specs	**Colour camera**	**Stereo pair**
Sensor	IMX214 (PY047 AF, PY062 FF)	OV7251 (PY013)
DFOV / HFOV / VFOV	81° / 69° / 54°	86° / 73° / 58°

Camera Specs	Colour camera	Stereo pair
Resolution	13MP (4208×3120)	480P (640×480)
Focus	AF: 8 cm – ∞ **OR** FF: 50 cm – ∞	Fixed-Focus 6.5 cm – ∞
Max Frame rate	35 FPS	120 FPS
F-number	2.2 ± 5%	2.0 ± 5%
Lens size	1/3.1 inch	1/7 inch
Effective Focal Length	3.37 mm	1.3 mm
Pixel size	1.12 µm x 1.12 µm	3 µm x 3 µm

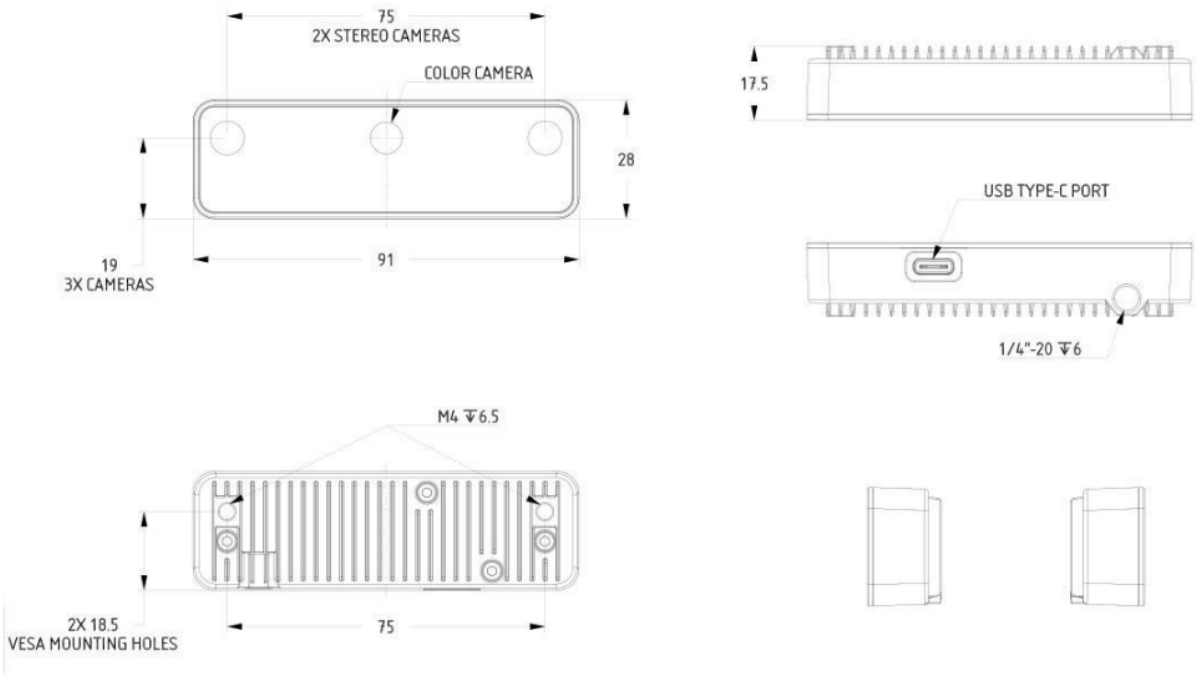

Figure 4 OAK-D lite device layout and dimensions

This OAK camera has a baseline of 7.5 cm - the distance between the left and the right stereo camera. Minimal and maximal depth perception (MinZ and Max) depends on camera FOV, resolution, and baseline.

- Ideal depth range: 40 cm - 8m
- MinZ: ~20 cm (480P, extended), ~35 cm (480P)
- Depth accuracy (See 480P, 75 mm baseline distance OAKs for details):
    - 40 cm - 3m: below 2% absolute depth error
    - 3m - 6m: below 4% absolute depth error
    - 6m - 8m: below 6% absolute depth error
- Operative temperature between -20° and 50°

Most of the power is consumed by the RVC2, so the power consumption mostly depends on the workload of the VPU:

- Base consumption + camera streaming: 2.5W - 3W
- AI subsystem consumption: Up to 1W
- Stereo depth pipeline subsystem: Up to 0.5W
- Video Encoder subsystem: Up to 0.5W

The total power consumption can be up to ~5W if you are using all the features at 100% at the same time. To reduce the power consumption, you can reduce FPS of the whole pipeline - that way, subsystems will not be utilized at 100% and will consume less power.

The difference between OAKD AND OAKD-LITE are:

- Mono cameras have lower resolution (640×480 instead of 1280×800)
- There is no power jack, as most users just use the USB-C for power delivery, which provides 900mA at 5V and is enough for most use cases. However, some functions (e.g., inference, video encoding) can lead to

large current spikes, so there is a chance that hosts like RPi will not be able to provide enough power. In that case, you should use a Y-adapter.

- Robotics Vision Core 2 (RVC2) chip-down design, instead of connecting the OAK-SoM to the baseboard
- Sensor connectors are shorter and take up less space.

All the information above and the datasheet are available with all the data sheets in the Luxonis® web page [12].

2. CALIBRATION

Camera calibration is a fundamental task in computer vision and is crucial in various applications such as 3D reconstruction, object tracking, augmented reality, and image analysis. Accurate calibration ensures precise measurements and reliable analysis by correcting distortions and estimating intrinsic and extrinsic camera parameters. Each camera records the scene in its own 2D image plane exploiting the pinhole camera mode, then in consideration of each objective's distortion, the "disparity" algorithm computes the third dimension. In this chapter, we are going to discuss the importance of calibration and its techniques. It is important to distinguish the "formal calibration": starting from the intrinsic parameter and tuning the external one to reduce the measurement error, and the "device parameter calibration" is due to better acquiring the body and its landmarks (see parameter optimization chapter). "Formal calibration" stands for a procedure that starts from known parameters (intrinsic and extrinsic parameter), that are given for or extracted from the device camera and known distances between objects.

2.1 Intrinsic and extrinsic parameters

Intrinsic parameters are:

- Focal length, expressed in pixels.
- Coordinate of the point of intersection between the optical axis and the image plane. (pinhole)
- Distance between the two cameras on the board.

The first two parameters are obtained through these lines of codes through which the developer let the user access the cameras intrinsic:

intrinsic=calib data.set Camera Intrinsics (dai. CameraBoardSocket.RIGHT)

intrinsic=calib data.set Camera Intrinsics (dai. CameraBoardSocket.LEFT)

The output of these two functions is a 3×3 matrix where the elements [1,1] and [2,2] are the focal length expressed in pixels, while the elements [1,3] and [2,3] are the coordinates of the point of intersection.

Let's throw some the definition:

Focal length:" The distance between the centre of a lens or curved mirror and its focus."

Optical axis:" The straight line passing through the geometrical centre of a lens and joining the two centres of curvature of its surfaces."

Image plane:" The plane that contains the object's projected image and lies beyond the back focal plane."

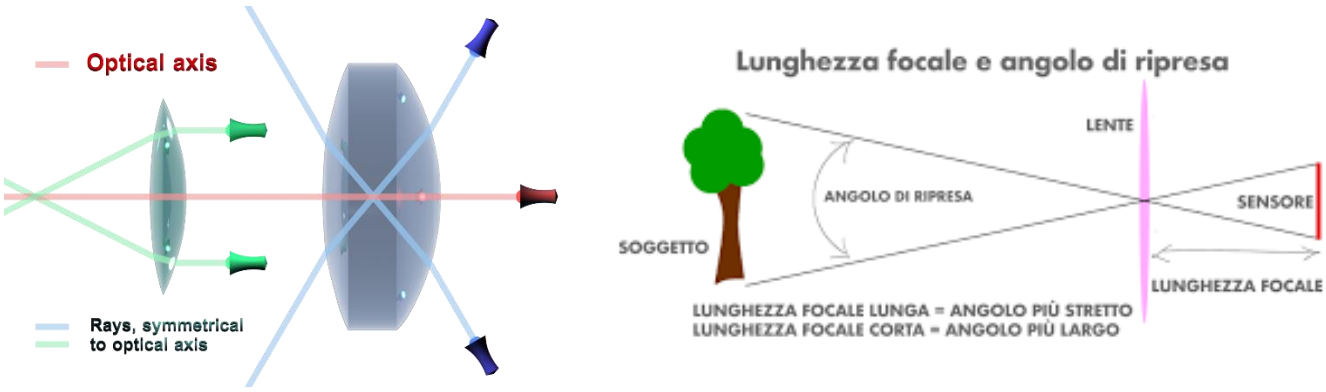

Figure 5 Optical axis and focal length

2.2 Pinhole model

The camera obscura, or pinhole image, is a natural optical phenomenon, at first theorised by the Arab physicist Ibn al-Haytham and then implemented by Giambattista Della Porta in his Magia Naturalis in 1558.

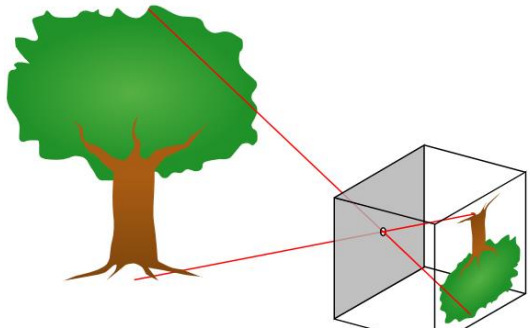

Figure 6 Pinhole model

This method is amazingly simple: the reflection of the object goes into a small hole in a box. A lens positioned in the hole projects an image on a surface.

So, let's talk about the pinhole camera model.

What we want to do is: take a point in 3D and transform it into 2D coordinates. For example, the tree in the figure above is in 3D, but the image is projected in 2D. Let's look at the formulas:

$$E : \mathbb{P}^3 \rightarrow \mathbb{P}^2$$

$$P' = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Projection matrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Point P' is the point in 3D coordinates, f is the focal distance, and x, y, z are the 3D coordinates. So, we have a matrix 3×4 that multiplies a 4-element column vector, and as a result, we have a 3×1 vector.

As we can see, z_p remains unchanged, and we can introduce c_x and c_y, which are the coordinates of the optical axis's intersection point with the image plane (pinhole).

$$p' = \begin{bmatrix} fx_p + c_x z_p \\ fy_p + c_y z_p \\ z_p \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

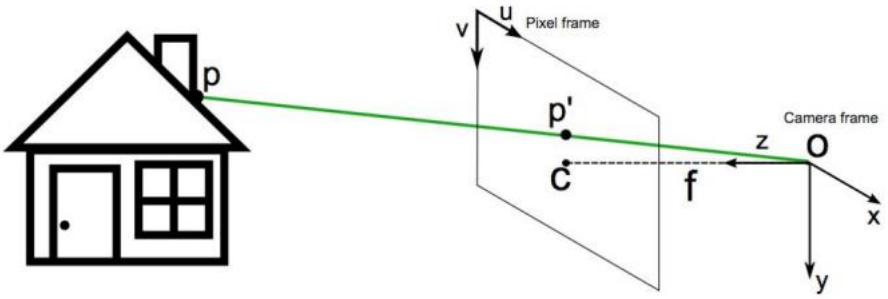

Figure 7 Projection of a 3d point in a 2d plane.

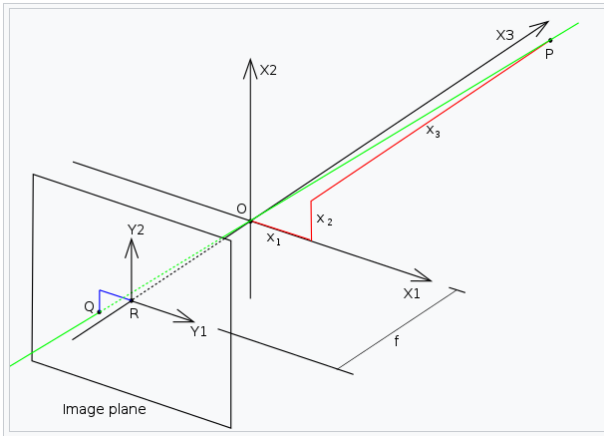

Figure 8 Image plane, centre of the image, focal length representation.

So, there is a point P at *a* x_1 x_2 x_3 coordinate, where x_1 x_2, x_3 is the coordinate system's axis while in O, the origin of the coordinate system, there is the pinhole. Plus, since $f>0$ there is an image plane in the back, which contains Y_1 and Y_2 axis, which are parallel to X_1 and X_2. The origin R is called the centre of the image. The point Q is in relation to the coordinate system Y_1, Y_2. f is the focal length and is the coordinate of R along the negative direction of X_3.

So, being y_1 and y_2, the coordinates of point Q in the Y_1-Y_2 coordinate system, we want to link Q to P.

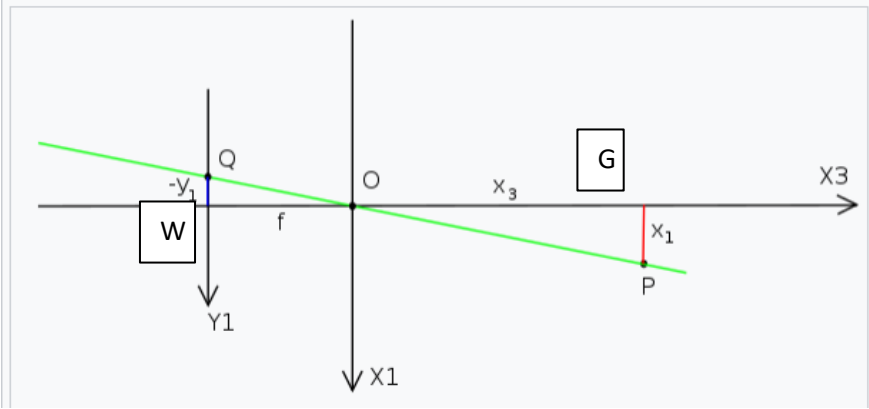

Figure 9 Catheti representation from side view

From another view, we see how there is a construction of two similar triangles where the catheti of the *QOW* triangle are -Y_1 and *f*, while the triangle *OPG* has x_1 and x_3 as catheti.

So, the following equations hold:

$$\frac{-y_1}{f} = \frac{x_1}{x_3} \qquad \frac{-y_2}{f} = \frac{x_2}{x_3}$$

that can be summarized as:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = -\frac{f}{x_3}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

In conclusion, we have established a relationship between the coordinates in the image plane and the principal coordinate system, and we can see that the point's coordinates are flipped due to the minus sign.

2.3 Disparity

Our goal is to find a way to assign 3D coordinates to an object in a 3D space. It is impossible to do so with one camera, but as the word "stereo-photogrammetry" suggests, we need at least two cameras.

As aforementioned, we have 2 cameras and so two image planes. We want to establish the distance between a certain point and a global reference system. Leaving aside the reference system for a moment and focusing solely on the distance, how can we do it?

Looking at the difference of the coordinates of the same point in the image planes of the two cameras, as referred to in the image below.

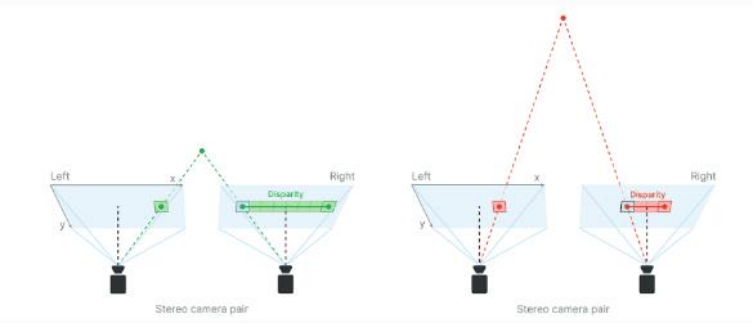

Figure 10 Disparity schematization

Disparity refers to the distance between two corresponding points in a stereo pair's left and right image. When calculating the disparity, each pixel in the disparity map gets assigned a confidence value from 0 to 255 by the stereo-matching algorithm as:

- 0 - maximum confidence that it holds a valid value
- 255 - minimum confidence, so there is more chance that the value is incorrect

Disparity and depth are inversely related. As disparity decreases, depth increases exponentially depending on baseline and focal length. Meaning, if the disparity value is close to zero, then a minor change in disparity generates a substantial change in depth. Similarly, if the disparity value is big, then substantial changes in disparity do not lead to a substantial change in depth. By considering this fact, depth can be calculated using this formula:

$$\text{depth} = \text{focal length in pixels} * \text{baseline} / \text{disparity in pixels}$$

2.4 Distortion

The pinhole model has a big problem: it does not take into account the camera distortion.

The figure below presents the three types of optical distortion.

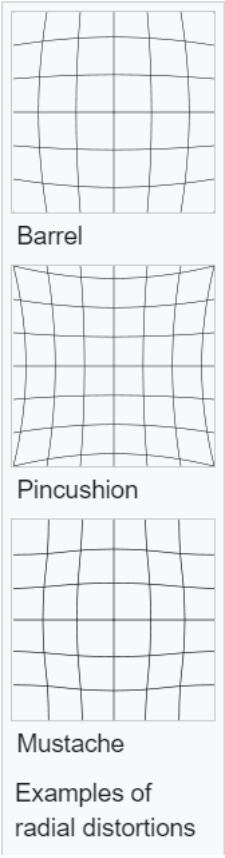

Figure 11 Types of optic distortion

The optical distortion is a deviation from rectilinear projection, a projection in which straight lines in a scene remain straight in an image.

In most cases it is due to defects in the camera's objective.

The pinhole camera model describes the image projection as a linear operator when working in projective spaces. Lens distortion produces a non-linear displacement of points after their projection.

Usually, it is a good approximation to model the lens distortion with the polynomial radial distortion model:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = (1+\mathbf{k_1}r^2+\mathbf{k_2}r^4+\mathbf{k_3}r^6)\begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2\mathbf{p_1}x_d y_d + \mathbf{p_2}(r^2+2x_d^2) \\ \mathbf{p_1}(r^2+2y_d^2) + 2p_2 x_d y_d \end{bmatrix}$$

point location in retina plane (unitary f) if the pinhole camera were perfect

Distorted location in retinal plane

$$r^2 = x_d^2 + y_d^2$$

Where k_i and p_i are distortion parameters. So, we have to find a way to calculate those parameters and represent optical distortion.

2.5) Charuco boards

"Charuco" is a word pun that comes from the union between the words "chessboard" and "Aruco."

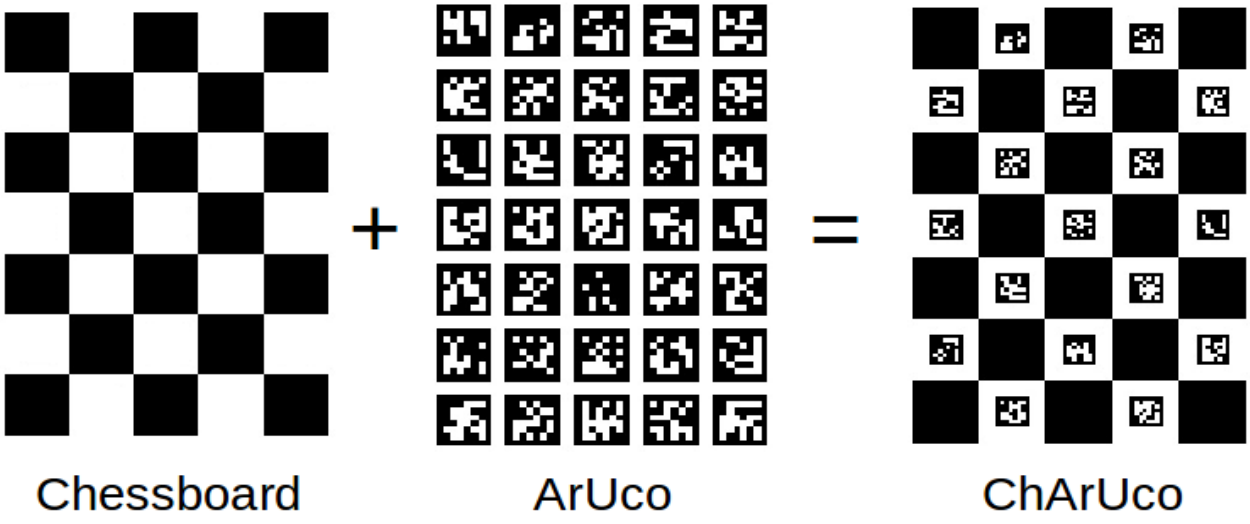

Figure 12 Charuco board composition.

Aruco markers and boards are extremely useful due to their fast detection and their versatility. However, one of the problems of Aruco markers is that the accuracy of their corner positions is not too high, even after applying subpixel refinement.

On the contrary, the corners of chessboard patterns can be refined more accurately since two black squares surround each corner. However, finding a chessboard pattern is not as versatile as finding an Aruco board: it has to be completely visible, and occlusions are not permitted.

A ChArUco board tries to combine the benefits of these two approaches.

The ArUco part is used to interpolate the position of the chessboard corners so that it is versatile enough to be used as a marker board since it allows occlusions or partial views. Moreover, since the interpolated corners belong to a chessboard, they are fully accurate in terms of subpixel accuracy.

When high precision is necessary, such as in camera calibration, Charuco boards are a better option than standard Aruco boards.

2.6 Parameters calibration

Starting from known distances, such as the side of the black square in the Charuco board, we can calculate the distortion parameter and calibrate the cameras. The script calinbrate.py which can be found in the depthai repository, is executed through the command line:

python3 calibrate.py -s [side of the square] -brd [directory of the .json file]

the inputs are the length of the square's side expressed in cm; the location where is stored the .json file corresponding to our device. The .json file is also present in the depthai repository under the "resources" folder.

For a correct calibration procedure, it is necessary to print the charuco board on a flat screen or use a paper sheet glued on a hard and flat surface. Then, measure the side of the black square on the charuco board with a ruler The square's length must be at least of 2.2 cm, although the program does not work.

The user must frame the charuco board from various angles and "take picture" of the charuco board by pressing the space tab.

The protocol used was:

1) Measure the length of the box printed on the screen.
2) Start the program with the correct inputs.
3) Frame the charuco board.
4) Press the space tab to take a picture.
5) Move the camera.
6) Repeat from point 3 from various angulations.
7) Press the s key to start the auto-calibration.

I suggest taking at least 3 pictures from each angulation chosen and having a total of 39 pictures. The device has three cameras, so each time we take a picture, we have 3 separate frames captured. My protocol reckons on 3 pictures from the left, 3 picture from the right, 3 from the top and 3 from the bottom view and one with the charuco board at the centre of the view that engulf the whole frame.

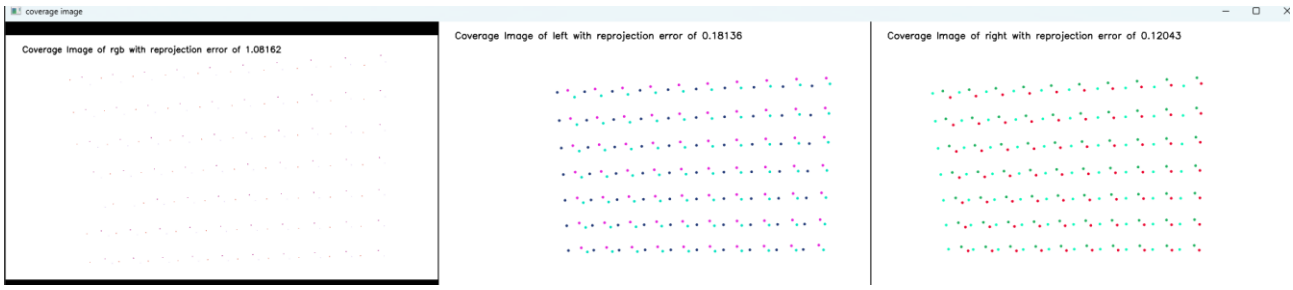

Figure 13 Output grill made of point.

Every time a picture is taken, a grill made of points is shown. Each point represents the vertex of the squares on the board.

After pressing “s,” the device calibration starts. This is a completely autonomous procedure. It is based on the Zhang, starting from a known distance (the side of the square). For every picture took, the error is calculated and printed on the screen.

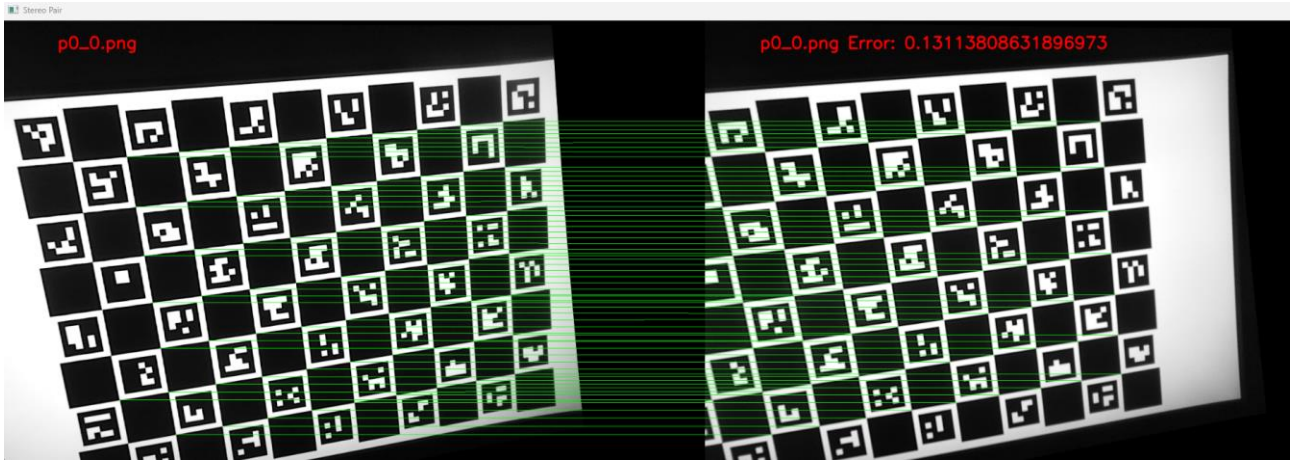

Figure 14 Visual output of the calibration procedure

If the calibration was successful the message: “EEPROM CALIBRATED” is shown to the user, and the calibration parameters are overwritten on the eproon(user modifiable rom)), otherwise, if the error is considered too high, the device aborts the procedure informing us that the calibration was failed.

This ends the calibration phase.

3. POINTCLOUD(PCL) ACQUISITION

One major aspect of this technology is the fact that we work “on” the whole subject, and there are no actual markers, so the whole body is under analysis through the point cloud recessing. This comes in handy if we want to reconstruct the body surface (see post-processing PCL chapter). A point cloud is a discrete set of data points in a given space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z). All the cameras present a field of view, but the information that comes from them is bidimensional. We are interested in the reconstruction of a 3D PCL, but how is it possible?

3.1 PCL in oak-d

All the devices we are using present a set of 3 cameras: one colour camera and two black and white cameras. The 3D reconstruction is performed mostly (see multiple device chapter) by the black and white cameras. Each camera presents a FOV. The point cloud is fulfilled by the intersection of the 2 cameras’ FOV. When an object or a point is visible by the 2 cameras, the disparity is computed and so the distance of such object from the camera is recorded.

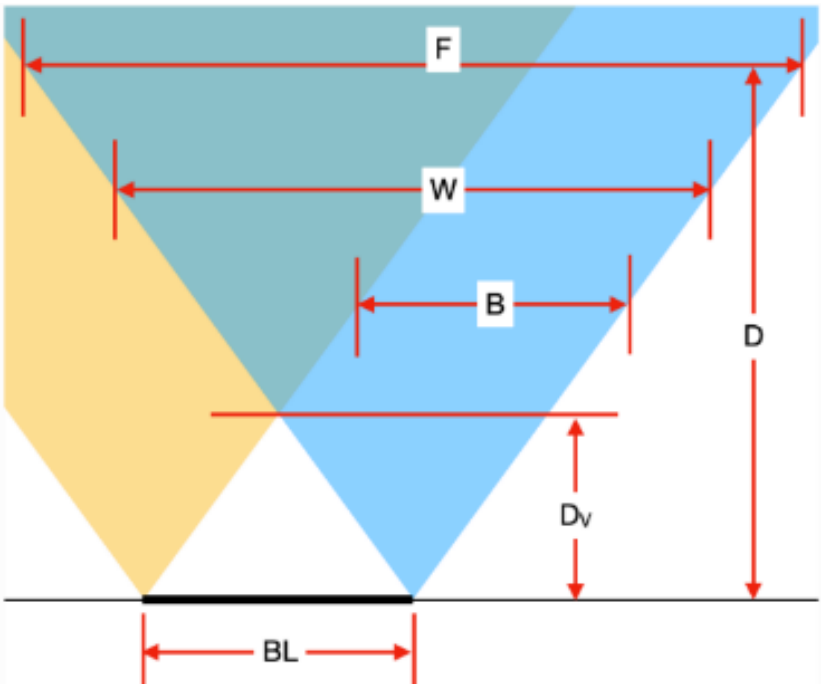

Figure 15 Cameras FOV

As the figure above shows, there are dark spots in the point cloud. Since depth is calculated from disparity, which requires the pixels to overlap, there is inherently a vertical band on the left side of the left mono camera and on the right side of the right mono camera, where depth cannot be calculated, since it is seen by only 1 camera. Plus, close to the camera, the two cons of vision don't overlap so for a short range Dv, the point cloud is not available. The formulas to calculate such spaces are:

$$F = 2 * D * tan(HFOV/2)$$

$$Dv = (BL/2) * tan(90 - HFOV/2)$$

So, for B we have:

$B = 2 * Dv * tan\ (HFOV/2) * W / F$

$B = 2 * Dv * tan\ (HFOV/2) * W / (2 * D * tan\ (HFOV/2))$

$B = W * Dv / D$

where HFOV is the Horizontal Field of view and BL the length of the baseline. So, the shorter the BL, the closer we can acquire the PCL.

The baseline length determines another aspect of the PCL clarity.

If the depth results for close-in objects look weird, this is likely because they are below the minimum depth-perception distance of the device.

To calculate this minimum depth-perception distance, we use the depth formula and choose the maximum value for the disparity_in_pixels parameter (keeping in mind that it is inversely related to minimum depth-perception, so the maximum value will yield the smallest result).

$$MinZ=focalLength*baseline/95$$

For example, OAK-D has a baseline of 7.5 cm, focal_length_in_pixels of 882.5 pixels, and the default maximum value for disparity_in_pixels is 95. By using the depth formula, we get:

min_distance = *focal length in pixels * baseline / disparity in pixels = 882.5 * 7.5 cm / 95 = 69.67 cm or roughly 70 cm.*

This parameter does not affect our type of acquisition that much, since it is important to have the whole body in the image frame, we operate at a longer distance (2 to 3 m).

The maximum depth perception distance depends on the accuracy of the depth perception.

$$\text{depth}[\text{mm}] = f_x[\text{px}] \cdot \frac{\text{baseline}[\text{mm}]}{\text{disparity}[\text{px}]}$$

Looking at the depth formula above, we can see that either a larger baseline distance or a larger focal length will result in further depth at the same disparity. In these cases, the depth accuracy will be higher.

Focal length is the distance between the camera lens and the image sensor. The larger the focal length, the narrower the FOV.

So, to get long-range depth perception, we can increase the baseline distance and/or decrease the FOV.

The formula used to calculate max distance is an approximation, but as follows:

$$Dm = (baseline/2) * tan ((90 - HFOV / HPixels) *pi/180)$$

Where HPixels is the height of the frame expressed in pixels.

So, using this formula for existing models, the theoretical max distance for OAK-D (7.5 cm baseline) is:

$$Dm = (7.5/2) * tan ((90 - 71.9/1280) *pi/180) = 3825.03 cm = 38.25 meters$$

If greater precision for long-range measurements is required, consider enabling Subpixel Disparity or using a larger baseline distance between mono cameras.

3.2 Subpixel method

Subpixel mode improves the precision and is especially useful for long-range measurements. It also helps for a better estimation of surface normal.

Besides the integer disparity output, the Stereo engine is programmed to dump the memory cost volume, that is 96 levels (disparities) per pixel, then software interpolation is done on Shave, resulting in a final disparity with 3 fractional bits, resulting in significantly more granular depth steps (8 additional steps between the integer-pixel depth steps), and also theoretically, longer-distance depth viewing - as the maximum depth is no longer limited by a feature being a full integer pixel-step

apart, but rather 1/8 of a pixel. In this mode, stereo cameras perform 94 depth steps * 8 subpixel depth steps + 2 (min/max values) = 754 depth steps

Moreover, adding subpixel fractional bits, at most 5, shifts the limit cap to 3010 layers according to the formula:

$$unique\_values=94*2^{subpixel\_bits+2}$$

This method enhances the depth of the point cloud to the detriment of the disparity.

4. LANDMARK RECOGNITION

As said, our camera does not follow any actual marker. The landmark recognition is performed by an AI algorithm and an end-to-end pipeline that could run a specific Python script on the embedded camera's CPU/GPU or on the computer’s GPU. Our goal is to have a fully automated and autonomous program that can identify a local reference system and give some visual feedback to the user. Moreover, through the pointcloud reconstruction, we have further information about body surface reconstruction.

4.1 Blazepose

BlazePose (Full Body) is a pose detection model developed by Google that can compute (x, y, z) coordinates of 33 skeleton key points, 16 more than Common Objects in Context (COCO).

BlazePose consists of two machine-learning models: a Detector and an Estimator. The Detector cuts out the human region from the input image, while the Estimator takes an image of the detected person as input and outputs the key points.

BlazePose outputs the 33 key points according to the following ordering convention.

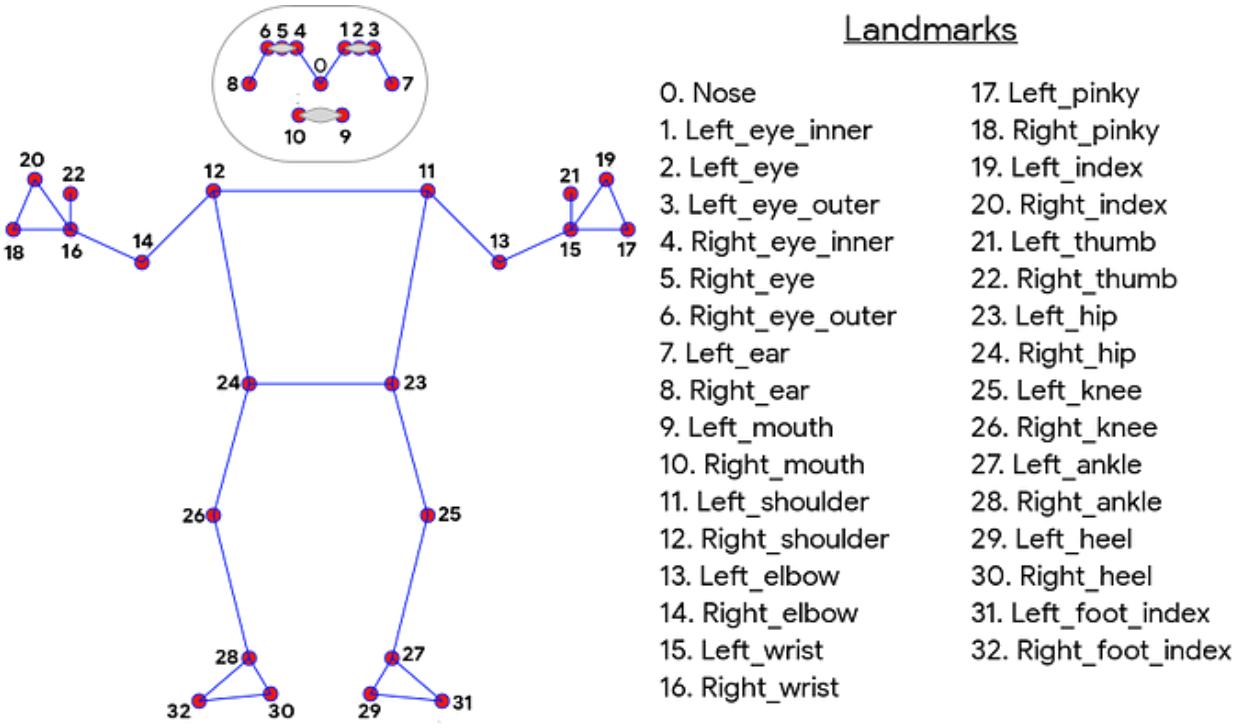

Figure 16 Blazepose stick diagram and landmarks.

The Detector is a Single-Shot Detector (SSD) based architecture. Given an input image, it outputs a bounding box and a confidence score. After the detection of the key points, they are drawn on the output video of the human body. Each key point is linked to an anatomic landmark and records the movement of that region in 3 coordinates.

There are two ways to use the Detector. The first is the box mode, where the bounding box is determined from its position (x, y) and frame size (w_{idth}, h_{eight}). The second one is the alignment mode, where the scale and angle are determined from (kp1x, kp1y) and (kp2x, kp2y), and bounding box including rotation can be predicted.

The first output of the Estimator are landmarks, the second one is flags. The landmarks are made of 165 elements for the (x, y, z, visibility, presence) for every 33 key points.

The z-values are based on the person's region of interest (ROI). The ROI is a small portion of the body placed between the hips or, if that region is not visible, between the shoulders. Only in the ROI the distance - the z coordinate - is calculated through

disparity. Although, all the key points have the z coordinate calculated through GHUM (3D human shape modelling pipeline).

Key points between the hips and the camera have z negative, and beyond the hips have z positive value.

The flags are visibility and presence; those are stored and are converted to probability by applying a sigmoid function. The visibility flag returns the probability of key points that exist in the frame and are not occluded by other objects. The presence one returns the probability of key points that exist in the frame. [1]

Then, the key points' coordinates are converted to a world reference system and stored in a ".csv" file. In fact, these are distances between the landmark and the ROI, so it is necessary to establish a relative reference system. An Aruco marker is used to do so.

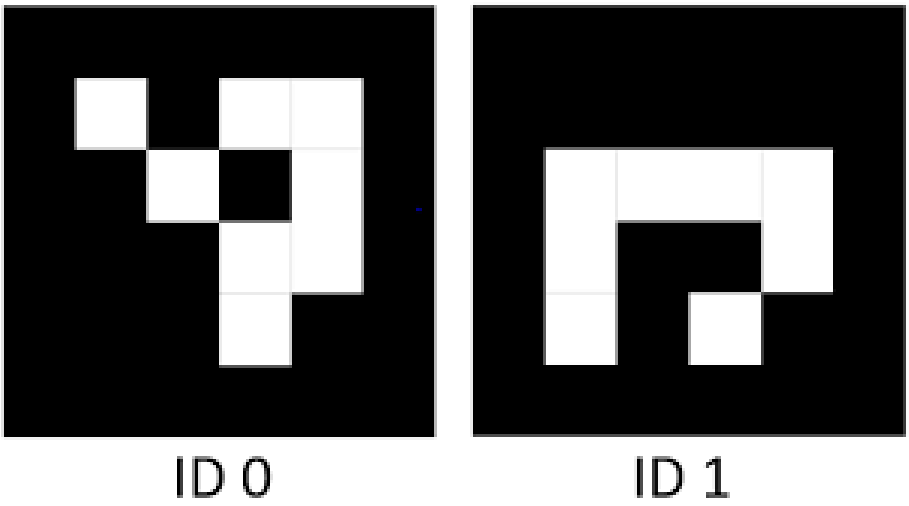

Figure 17 aruco marker 4×4

During the calibration phase, we had to choose the type of marker that composed the charuco board. If we used Aruco 4×4 marker during the calibration phase, the system recognizes only such type of markers, not, for example, aruco 5×5 markers. Once the marker is recognized, the reference system is drawn upon the marker itself, and rotation and translation vectors are calculated using the command "cv2.aruco. estimatePoseSingleMarker ()."

After this, the rotation matrix is computed with the Rodrigues' rotation formula and stored in a .txt file. The transition vector is stored, too. To give some feedback to the user, the angle between the side of the user and the left arm is shown. The figure below shows a real time acquisition where the reference system is drawn while the subject moves the arm. Note that even when the arm occludes a landmark, the stick diagram is still adherent to the body.

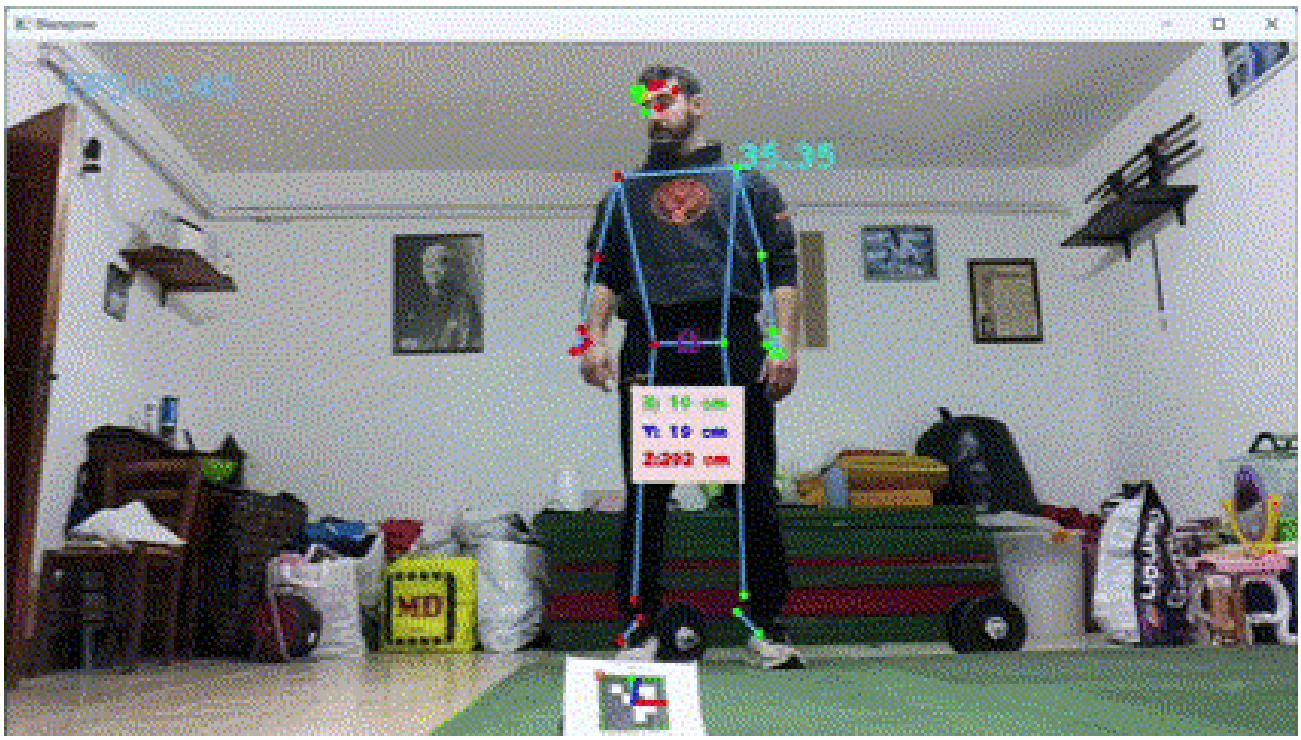

Figure 18 Blaze Pose running with local reference system and feedback to the user.

Here are reported the formulas to obtain the coordinates in the local reference system.

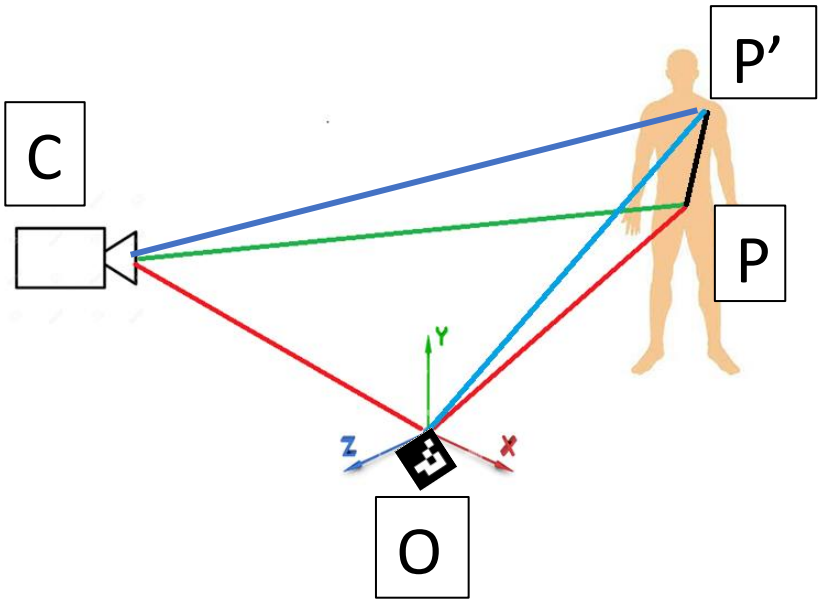

Figure 19 Schematization of BlazePose coordinate computation.

$CP = CO + OP$

$OP + PP' = OP'$

${}^{cam}R_r P = P_r$

$P_r + CO = CP$

$CP' = CO + OP'$

${}^{cam}R_r P' = P'_r$

4.2 Blazepose VS HOGCV

Leaving aside the body tracking, why is it preferable to use Blazepose instead of the HOGCV detector?

OpenCV features an implementation for an extremely fast human detection method called HOG (Histograms of Oriented Gradients). This method is trained to detect pedestrians, which are humans mostly standing up and fully visible. So, one should not expect it to work well in other cases. Moreover, this method is gradient-based, so it is expensive in terms of memory and GPU performance.

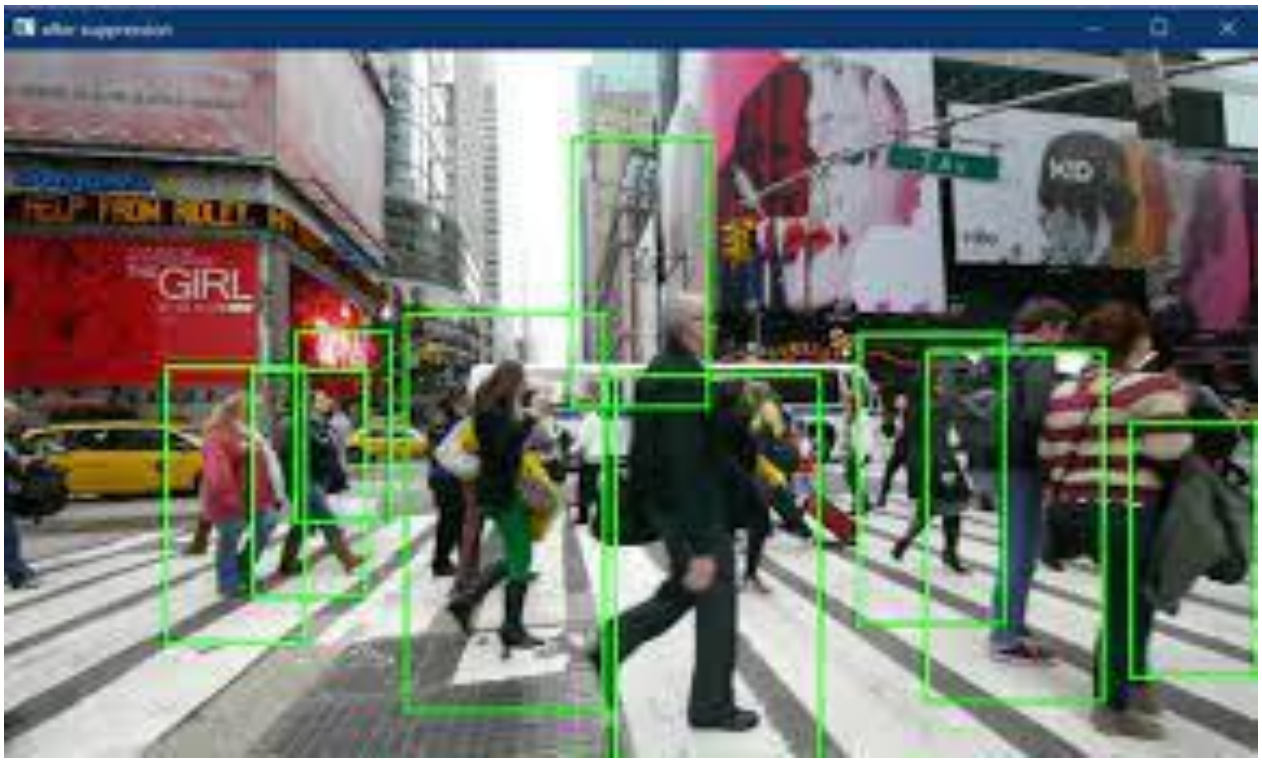

Figure 20 HOGCV output

As already explained, Blazepose is a machine learning-based algorithm that uses "MediaPipe Pose Landmarker" to detect landmarks of human bodies in an image or video.

Mediapipe uses Lightweight ML models all while preserving accuracy, domain-specific processing - including vision, text and audio, end-to-end acceleration across CPU and GPU, complex pipeline graph with multiple models and states, Cross-

platform deployment to Android, iOS, web, and bare metal. In conclusion, Blazepose is a more efficient and less memory-demanding machine learning-based algorithm.

Regarding body tracking, the HOG method does not have this feature. So, we proceed to compare Blazepose with Open Pose. Open Pose is a real-time multi-person human pose detection library that has shown, for the first time, the capability to jointly detect the human body, foot, hand, and facial key points on single images. OpenPose can detect a total of 17 body key points.

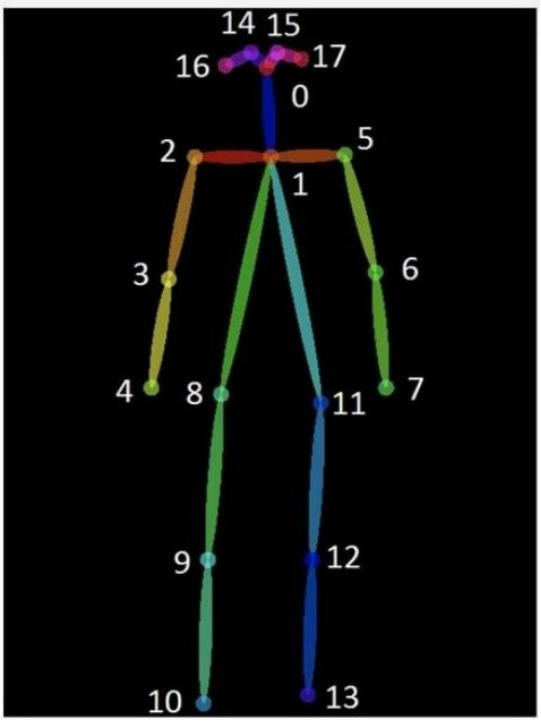

Figure 21 OpenPose stick diagram

As said before, this model presents 16 fewer key points, does not implement a real-time 3D evaluation, and is more memory-demanding. Although it is more precise and reliable. In fact, Openvivo is the “Gold standard” for markerless body tracking. [2], [2].

4.3 GHUM

Generative 3D Human Shape and Articulated Pose Models is a statistical, articulated 3D human shape modelling pipeline within a fully trainable, modular, deep learning framework. It uses over 60.000 diverse human configurations for training to provide a generic human model of different resolutions consisting of 10.168 vertices. Given a training set of human body scans, represented as an unstructured point cloud, where the number of points P varies, we learn a statistical human model $X(\alpha) \in R$ representing the variability of body shapes and natural deformation due to articulation. Since $X(\alpha) = M(\theta, \tilde{X}(\beta b), \Delta\tilde{X}(\theta), \Delta\tilde{X} f(\beta f), C(\tilde{X}), \omega)$ (1) where:

- $\tilde{X}(\beta b) \in R\ 3V$ is the identity-based rest shape in A pose, with βb a low-dimensional embedding vector encoding body shape variability.

- $\Delta\tilde{X} f(\beta f)$, is the facial expression at neutral head pose controlled by low-dimensional latent code βf;

- $c = C(\tilde{X}) \in R\ 3J$ are skeletal joint centres dependent on the body shape, $\theta \in R\ 3\times(J+1)$ is a vector of skeleton pose parameters consisting of 3 rotational DOFs in Euler angles for each joint, and 3 translational variables at the root, $\omega \in R\ V \times Ia$, pose-dependent corrective blend shapes $\Delta\tilde{X}(\theta)$

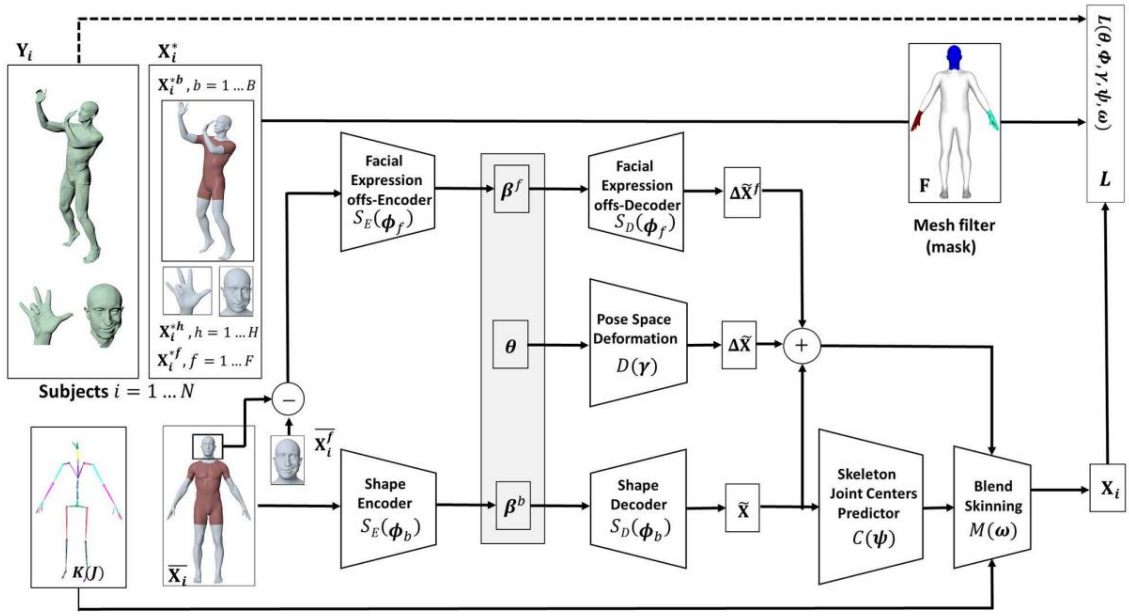

Figure 22 GHUM's block diagram.

Starting from a skeleton K in dependencies of the joint (J) we can estimate the pose and through the skinning procedure, the body model is linked to the stick diagram. Finally, from the body model, Blazepose predicts the z coordinate in relation to C, its joints centres, and computes the distance between ROI and each key point. As already mentioned, the shape is detected by the Detector, the pose by the Estimator [4]

4.4 Parameter optimization

As mentioned above, there is a list of parameters that the user could change to optimize the blaze pose algorithm.

- min_pose_detection_confidence= this is the minimum confidence score for the pose detection to be considered successful.
- min_pose_presence_confidence= this indicates the minimum confidence score of pose presence score in the pose landmark detection
- min_tracking_confidence= this parameter is the minimum confidence score for the pose tracking to be considered successful.

Each of these parameters has a value range between 0.0 and 1.0, where 0 is considered zero confidence and 1 is maximum confidence. The tuning of these parameters depends on our goals. If we are interested in the presence of the body and want to be sure of the presence of the body, it is suggested to set a high value for pose presence confidence and low value for tracking and pose detection. However, if we are interested in tracking movement's body, it is suggested to use lower value for pose presence and higher for detection and tracking. In accomplishing tasks such as walking, it is inevitable that key points appear and disappear from the frame due to occlusion by the body's parts. Blaze pose has the possibility to track the key points that are not visible within a certain confidence.

Which value to assign to the parameters really depends on the situation. Factors to take in account are:

- illumination: the room must be well illuminated, no matter from natural or artificial light, as long as there are no overshadowing phenomena [5]
- clothes: avoiding black or solid one block-colour clothing in general is recommended.
- distance: it is vital to have the whole body in the frame
- room furniture: avoid objects that resemble the human shape to avoid affections or occlusions of the ROI.

4.5 Post processing of the point cloud

Why do we need an algorithm that simulates a body model when we can acquire a 3D point cloud and establish the coordinates of each point?

Manly for 2 reasons:

- the typical aspect of a point cloud is this:

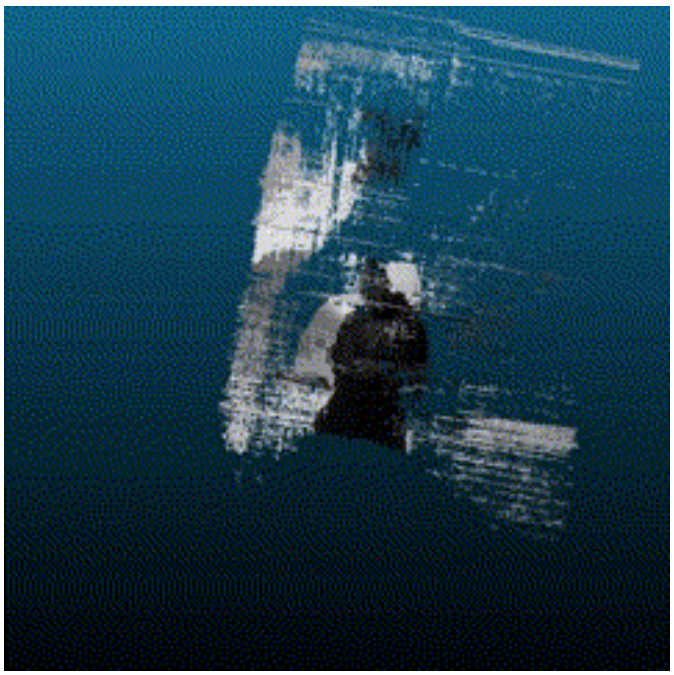

Figure 23 Unpolished pointcloud

The richness of outliers makes it practically impossible to have a reasonable number of points to establish the coordinates of each landmark. Nevertheless, this is a static pose, so all the key points are visible within the color camera frame, but during a task, like walking, the visibility of the points is not guaranteed.

- It is memory demanding:

Reconstructing the 3D coordinates through stereo is much more demanding than predicting them. So, the aspect of real-time reconstruction of Blazepose fails if we decide to use a “pure” stereo approach.

In fact, a polished point cloud is a key step to have a full 3D scan of the body to reconstruct the surface. Open3d is an open-source Python library that lets the user operate on point clouds.

There are different functions for the reconstruction and cropping of the PCL. It uses triangular meshes to reconstruct surfaces and overall works well to store and reconstruct the point cloud. The limit is in the crop. There are only two ways to crop:

- aligned bounding box (ABB).
- oriented bounding box (OBB).

Both bounding boxes encapsulate your geometry box. The same suggests ABB, which is aligned with the x, y, and z axes. The lines will always align with the global coordinate system if the user visualizes it. On the other hand, OBB has a rotation, which means it is not aligned with the coordinate system, but it could more tightly fit the geometry. Right now, the oriented bounding box is computed via PCA, which is far from optimal; hence, it does not produce the tightest possible fit.

Not even ABB is optimal for body surface reconstruction since the cropped point cloud maintains the “image background,” as we saw in the figure above.

To remove the background, it is necessary to implement a post-processing program that is gradient-based. For each point of the cloud, the least squares local plane is fitted to its k-nearest neighbours. The normal of each point is the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. After estimating the normal of each point, then we consider k nearest neighbours for the sample point and afterwards cluster the normal of those k nearest neighbours with the agglomerative technique.[6]

Algorithm 1 Agglomerative Clustering algorithm

procedure AGGLOMERATIVE–CLUSTERING
    Consider each element as an individual $Cluster$ at the first step
    for all the $Clusters$ **do**
        Find the angle between each $Clusters$
    end for
    Find the pair of clusters with minimum angle
    Merge two clusters with the minimum angle as in (1)
While $MinAngle < Threshold$
    Compute number of clusters
end procedure

Figure 24 Pseudocode for edges reconstruction

For the point cloud that we saw, the edges were extracted using k=5 and threshold value=0.2. The result is:

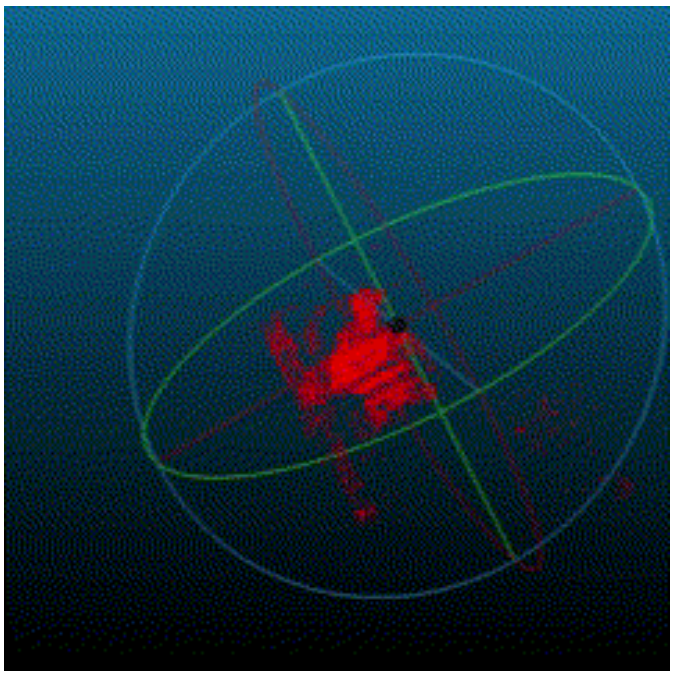

Figure 25 pointcloud edges extraction

Then, the user can filter the point cloud with statistical_outlier_removal. By doing so, one removes points that are further away from their neighbours compared to the average for the point cloud. It takes two input parameters: pcd. remove_statistical_outlier (nb_neighbors, std_ratio)

- nb_neighbors allows specifying how many neighbours are taken into account in order to calculate the average distance for a given point.
- std_ratio allows setting the threshold level based on the standard deviation of the average distances across the point cloud. The lower this number, the more aggressive the filter will be. The values used were neighbour =20000 and std=10^{-5}.

The point cloud polished resulting is:

Figure 26. On the right, a heatmap along the z axis is applied, where the blue represents the closer point and the red the further point from the camera. On the left, the pointcloud polished.

From this information is possible to extract a full 3D body scan through different acquisition of the same subject from different views or through the use of multiple devices (see multiple devices chapter).

5. FUNCTIONALITY

The fundamental part of data acquisition is that the actual data is reliable. Big amount of data can stand for nothing if they are not suited for processing. Low frame rate, bad calibration, and general substandard data quality can make the acquisition erroneous and possibly hurt the subject.

Live data management and elaboration are not the only choices we have. It is possible to work with pre-acquired data and then run the algorithms in post-processing. Moreover, a single device does not limit us, but we can use multiple cameras to improve the acquisition quality.

5.1 Encode

As seen in the table in the device chapter, the maximum operating frame rate of the 2 types of cameras are 35 fps for the oak-d-lite RGB camera and 60 fps for the oak-d RGB camera, while for the black and white cameras, we are well above 100 fps.

One problem with the body tracking algorithm is that the frame rate number drops when we operate in synchronicity between acquisition and data elaboration. Such a low value is unsuited for high-frequency movement like sprinting or running because it infringes the Shannon Theorem. Although the AI is guaranteed to work on two contiguous frames, a frame rate that does not respect the theorem leads to aliasing phenomena.

The encoding process prevents this. The camera can encode the RGB camera and both greyscale cameras at the same time. The RGB is set to 1920×1080 and the greyscale is set to 1280×720 each, all at 30FPS, for example. Each encoded video stream is transferred over XLINK and saved in a separate file. Pressing CTRL+C will stop the recording and then convert it, using ffmpeg, into an .mp4 file to make it playable. Note that ffmpeg will need to be installed and runnable for the conversion to mp4 to succeed. The results are then stored in three files: “colour,” “left,” “right.” In order, they represent: the video captured by RGB cam, the right and left cams. These videos can be further elaborated on by Blazepose, feeding them as input into the videos' directory.

As said, it is guaranteed that the algorithm works on all the frames since it checks both the timestamp and the total number of frames. If we need a large number of frames that exceed the capabilities of the RGB cam, it is possible to acquire the same scene at different frame rates. As said, the Black and White cameras have higher frame rate caps This is possible by changing the line of code:

device. getOutputQueue (name='ve1Out', maxSize=FPS)

where ve1Out is the stream name, and maxSize the frame rate.

Then, during the conversion, we must specify the FPS at which the video is encoded.

The command line to write in the cmd is:

ffmpeg -framerate FPS -i pathvideo -c copy pathstored

where "pathvideo" is the path where the ".H264" or ".H265" video is stored and "pathstored" is the path where we want to store the mp4 converted video.

5.2 Multiple devices

A crucial step to a lab configuration is the possibility to frame the same scene from multiple POVs. The advantages of using multiple devices are:

- Better point cloud acquisition: using multiple devices leads to an acquisition of the point cloud that better resembles the human structure. We can use 2 cameras on both sides of the subject and scan almost the entire body. Of course, there are going to be some dark spots due to occlusion and part of the body out of the camera FOV.
- Better acquisition of the landmark: As said, it is impossible to have all the key points in the FOV of the camera while completing a task. Even though Blazepose has coefficients of reliability, presence, and visibility to predict the key points' trajectory eventually, it is better to have the key points well visible to the camera. In doing this, the risk of artefact and general errors decreases.

As said many times, we are looking for a real-time acquisition and rendering of the information, so the only way to acquire simultaneous images from different POVs is to use more cameras. It is good to specify that because there is the possibility of rotating the point cloud in a way that composes a volume, this procedure, of course, must be made in post-processing and needs the subject to be completely still.

The point cloud is used as a reference system for the global reference system of the camera, so the image is presented as "the cameras see it." This cannot be optimal if more devices participate in the acquisition since they are naturally placed in 2 distinct positions of the room, meaning that the cameras' placement must be registered: camera calibration. All that is needed is to execute the script calibration.py in the repository gen-2multiple-device and follow the instructions in the readme.txt.

This procedure uses Open CV's findChessboardCorners to find the chequerboard, and "solvePnP" to compute the translation and rotation from the camera to the chequerboard.

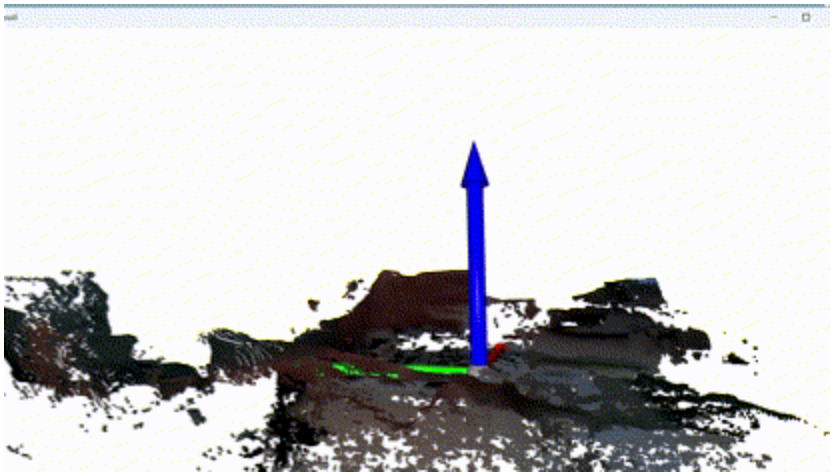

Figure 27. Hand movement 3D reconstruction. The reference system is built on top of a checkboard. The position of the board is computed in the calibration phase.

All we must do is frame a chessboard in the camera view, press "p" and wait for the system to write a 3D reference system on the board. Then, select the second device and repeat the same procedure. The cameras and the chessboard must occupy the same space during the whole calibration and point cloud processing.

In the same repository, the point cloud generator Phyton file is present. As above, read the readme.txt file before starting the acquisition. All this functionality does is unify and align the point cloud from the devices. The result is a wider 3D point cloud.

For clarity, all the devices that compute the point cloud are synchronised.

6. LIMITATION AND FUTURE DEVELOPMENT

6.1 Limitation

This technology is far from perfect. Its prominent limitation is consistency. This stems from its versatility as a data acquisition device. It works in every situation and in every condition as long as the user has a computer, a cable, and a camera. However, it is easy to make incorrect acquisitions if the operator does not know what is happening. Most common errors come from an inaccurate calibration, so it is important to follow the protocol. Other limitations are due to the *minZ* value. As said before, for our kind of acquisition the subject is positioned far from the zone subjected to distortion, but this is not the case while working with multiple devices. If we want to work in small volume, staying away from the area subjected to distortion is hard.

However, when working with longer distances, the operator must calibrate the system properly and ensure that the point cloud is present for the distance wanted. For example, the subpixel method and focus's value are suited for the type of acquisition.

Moreover, the camera method to calculate the distance, "disparity" precisely, does not perform well on flat surfaces such as walls. This happens because it is hard for black-and-white cameras to establish given points on a solid color and a flat surface where there is no reference. Although, new models are equipped with infrared cameras that work better in this situation.

Plus, the problems with point clouds that have already been discussed - visibility, polishing, and outlier detection - really depend on the background. So, making the subject stand out from the background is important.

Regarding BlazePose, as said, it is a fast-processing AI but has a flaw in clipping the stick diagram to the body, especially in the lower limbs. Moreover, it is a prediction algorithm, even though the creator states it is not, since it uses GHUM, a statistical human shape modelling pipeline.

6.2 Future development

Using the PCA method base is always risky. While working with the human body, we must consider soft tissue and body part deformation. Establishing a pose and making the model move accordingly is insufficient, like in GHUM or SMPL. SMPL is a realistic 3D model of the human body that is based on skinning and blending shapes and is learned from thousands of 3D body scans. Both methods present a real-time processing feature and a skinning procedure to link the stick diagram to the model. But is it worth sacrificing all this memory to elaborate a dummy and measure the landmark's displacement on a human body's surrogate?

Given that it is possible to have a 3D scan of the body, establish the subject's pose and implement a neural network, the direction to follow is to work with the actual human figure [11].

The idea is to adapt the point cloud in a human mesh as it happens in SMPL and GHUM, but the finality is to use the human model as a template and to overlap the point cloud on the mesh. The fitting of the point cloud on the mesh can be done by a neural network where the bias is the original mesh, the input is the landmark or point cloud from parts of the body (left leg, head, left forearm…) and the coefficients are a mix of the visibility and presence parameters and values on dependency of the anatomic part, like SMPL. The output is a set of points interpolated with the mesh preferred since there are three types of mesh models: male, female, and neutral gender.

Here is an example of a mesh:

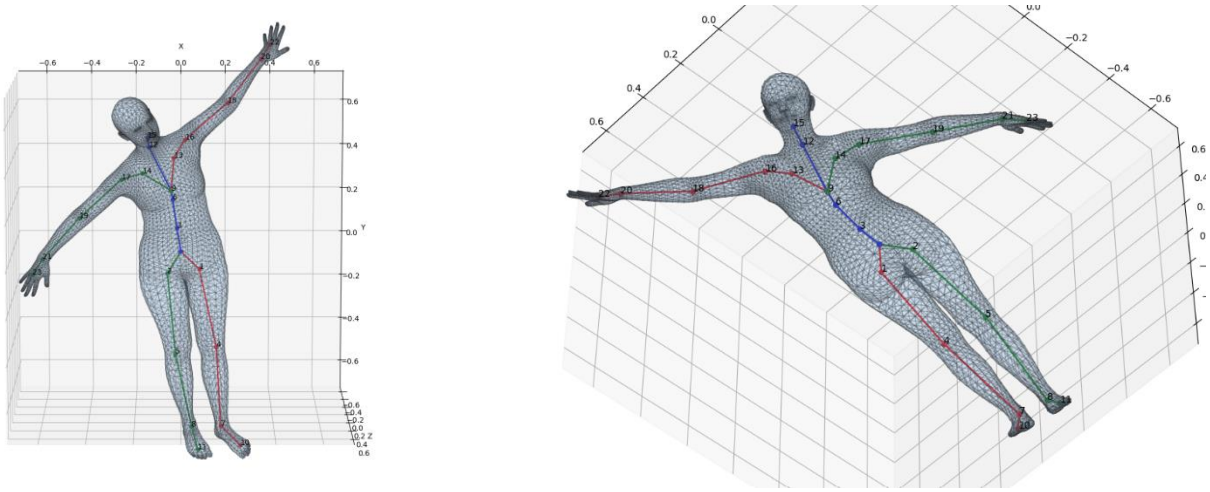

Figure 28 SMPL mesh example.

SMPL human body layer for PyTorchis is a differentiable PyTorch layer that deterministically maps from pose and shape parameters to human body joints and vertices. [7]

CONCLUSION

To summarise, the strong suits of this technology are accessibility, in both price range and availability, usability, and simplicity of working with and program. Its versatility makes the device suited for many applications; its fast-paced processing units make it perfect for a feedback system for rehabilitation purposes. In the research of biomedicine, particularly the study of human movement, systems that could give real-time feedback to the user are fundamental. Entertaining the subject during rehabilitation has been shown to increase both involvement and enhancement of the rehabilitation time and experience. As to this last point, gamification is the most effective way to keep the subject involved with the activity, but the devices' availability limits the effectiveness of this method. Implementing an RGB cam rehabilitation method is important, since everybody owns a smartphone capable of running rehabilitation programmes. BlazePose is only one of many MediaPipe method based, capable of running on embedded CPU like phones or oak-devices. Moreover, since the majority of smartphones have more than one camera, experimenting with multi-camera devices resembles the working conditions of modern phones really well. Although these devices didn't perform too well for diagnosis, this technology is quite new and further developments are expected.

REFERENCES

1 BlazePose: A 3D Pose Estimation Model David Cochard

2 Estimating Ground Reaction Forces from Two-Dimensional Pose Data: A Biomechanics-Based Comparison of AlphaPose, BlazePose, and OpenPose Marion Mundt

3 Comparing the Quality of Human Pose Estimation with BlazePose or OpenPose

4 GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models Hongyi Xu et al.

5 What Can We Learn from Depth Camera Sensor Noise? Azmi Haider and Hagit Hel-Or

6 Fast and Robust Edge Extraction in Unorganized Point Clouds, Dena Bazazian et al

7 SMPL: A Skinned Multi-Person Linear Model Matthew Loper

8 Computer Vision The Pinhole Camera Model

9Un algoritmo per la Camera Calibration basato sulla visione stereoscopica, David Melis

10 Automatic Posing of a Meshed Human Model Using Point Clouds Tamal K et al.

11 Graph-Based Point Tracker for 3D Object Tracking in Point Clouds Minseong Park et al.

12 Luxonis documentation page