3D Object shape reconstruction combining a single depth image and visual tactile sensors

Marc Badosa Samsó
Master in Computer Vision
Universidade de Santiago de Compostela, Spain
marc.badosa@rai.usc.es

***Abstract*—Full knowledge of an object's 3D shape is crucial for robotic manipulation tasks. However, obtaining this information in uncontrolled environments can be challenging. Additionally, robot interaction often obscures vision. High-resolution visuo-tactile sensors can provide dense local information about the object's shape. Nonetheless, working with visuo-tactile sensors has inherent limitations due to their small sensing area. Therefore, in some cases, the shape representation must be approximated without visual inspection. This work implements a methodology to achieve high-fidelity object shape reconstruction by extensively using a single RGB-D image view. The methodology estimates the symmetry axis plane for reflection and then adds different tactile measurements of the non-visual part using visuo-tactile sensors. The information from the depth camera and visuo-tactile sensors is processed and combined objectively to obtain the most accurate representation of the object's shape. The 3D mesh is then reconstructed using traditional methods, such as ball pivoting and Poisson surface reconstruction algorithms. Code available at: https://github.com/MBadosa/TFM**

***Keywords*—** 3D object reconstruction, Visuo-tactile sensors, Gelsight, Symmetry, Point clouds

I. Introduction

Object manipulation has become increasingly popular in recent years in the research world. Robots can perform a range of tasks, such as grasping, manipulating, and assembling, by having complete knowledge of the shape of the 3D object they are interacting with. To achieve this, robots must generate a representative 3D model of the object in real-time and on the fly while manipulating it.

To achieve this, the problem is typically addressed using normal or depth vision methods [1]. However, these methods may not always be accurate enough due to various challenges, such as partial object observation, object occlusion, poor visibility conditions caused by incorrect illumination, a limited range of view, or even object transparency. Sometimes, robots may not be able to perform tasks correctly due to incomplete knowledge of the object.

When considering how humans perceive the shape of an object, it is evident that we combine visual information from our eyes with tactile information from our hands [2]. This allows us to gain both approximate global knowledge and more precise, detailed information about specific areas. Therefore, the research community is currently trending towards this approach. As a result, recent advancements in tactile sensors combined with vision ([3]–[6]) have emerged. The GelSight Mini [6] is a modern and powerful tool that utilizes a layer of deformable gel, various light sources, and a specially calibrated camera to capture detailed images of surfaces it comes into contact with.

Efficiently fusing visual information from cameras with information from visuo-tactile sensors is a major challenge. The visual information is typically more extensive than that from the sensors due to their smaller contact area. Optimal adaptation of both pieces of information is necessary for successful fusion. When designing a feasible method, it is not possible to perform tactile measurements of all unknown areas of the object. This is due to the limited surface area of the visuo-tactile sensors, which would make the process time-consuming. Therefore, it is necessary to predict some of the unknown parts.

It is important to consider that the camera view provides a significant amount of information about the object's shape due to its high resolution. Additionally, it is worth noting that everyday objects often exhibit symmetry, either partial or complete. Consequently, the ability to view the front side enables us to estimate the appearance of the non-visible side. This can later be complemented by visuo-tactile measurements from sensors to add information or correct predictions.

This work proposes a framework for obtaining a 3D reconstruction of an object model using a single frontal image captured with an RGB-D camera and a set of tactile measurements made with the GelSight Mini sensor. To estimate the non-visible part of the object, a symmetry axis is calculated based on the frontal image captured by the camera. A series of tactile measurements are then taken using the sensor in the non-visible part. The obtained tactile information is processed using deep learning techniques to generate a high-resolution point cloud. This point cloud can be used to add or remove points from the reflected part.

After obtaining the complete point cloud, we use two well-known mathematical calculus-based algorithms, namely the Ball Pivoting algorithm and Poisson surface reconstruction, to perform 3D reconstruction of the object. This document will describe both methods. Finally, an evaluation is conducted using the YCB dataset.

The paper's specific contributions are:

- **Extensive use of the front image of the RGB-D camera:** The majority of works in this area focus on using either RGB-D images or full tactile sensor images. Some even add the depth image without further processing. However, we go beyond this by using a single RGB-D image to predict the symmetry axis of the item and mirror the depth capture. This is because most daily life items tend to be symmetric.
- **Optimal combination of visual and tactile information:** The aim is to optimally combine visual and tactile information by removing redundant or inaccurate points from the data obtained from the depth camera and visuo-tactile sensors. This will result in a higher-fidelity point cloud that can be reconstructed into a 3D object.
- **Evaluation using ground truth objects from the YCB dataset:** A comparison was made between the proposed method and the ground truth objects from the YCB dataset.

II. Related work

Tactile sensors have been extensively researched within the scientific community to replicate human-like tactile sensing abilities. This has led to the emergence of new and innovative hardware on the market, with vision-based tactile sensors being the most significant breakthroughs ([3]–[6]). These sensors have a common feature: they use a gel pad to make contact with objects, which is similar to

human finger skin. The gel pad is combined with an integrated camera that monitors the pad's deformation. By carefully calibrating the gel pad and the camera, the resulting tactile images can be analyzed to accurately determine the corresponding contact points, forces, and other useful information.

Of the various vision-based tactile sensors currently in use, GelSight is notable for its advanced and sophisticated technology and design. However, its capabilities have not been extensively explored due to its limited time on the market. Vision-based tactile sensors have a distinct advantage as they provide high-density, multi-degree-of-freedom (DOF) force sensing capabilities. These capabilities are encapsulated by using an image, which allows for the use of a wide range of computer vision techniques. Additionally, the extracted images can be used with neural network applications.

Due to the absence of high-resolution tactile sensors, researchers have typically employed multi-sensor approaches ([7]–[13]), utilizing depth cameras to capture the majority of global information and tactile sensors to supplement the missing shape information.

The majority of shape information is obtained directly from the depth camera, with tactile information typically used only as a refinement step. However, this approach has limitations due to occlusions. Combining both sensors provides a more reliable and complete reconstruction. For instance, Smith et al. ([14], [15]) proposed a methodology that primarily uses tactile data to predict local mesh deformation, with camera vision serving only to fill in gaps. Xu et al. [7] recently reconstructed deformable and non-deformable objects using visual and tactile information while manipulating them with a hand. They also provided an open-source simulator for performing tactile measurements.

Initially, most works in this area focused on active tactile exploration solely for object recognition without considering the required number of tactile samples, which ultimately increased the system's overall interaction time ([16]–[18]). Subsequently, other researchers [19] suggested acquiring haptic exploration strategies through a reinforcement learning framework applied to a recurrent attention model. However, their research solely concentrated on object classification and not reconstruction.

For surface reconstruction, previous studies have primarily relied on Gaussian processes ([20]–[22]), which provide natural uncertainty estimates to direct the next point of exploration. To enhance this process, Matsubara and Shibata [23] took into account both uncertainty and travel cost when selecting spots to touch using graph-based path planning. Another study, conducted by Bierbaum et al. [24], employed exploration through a dynamic potential field approach to generate point cloud prediction. However, both studies utilised deterministic strategies. Lastly, Smith et al. [15] proposed a fully data-driven solution in which actions are selected using policies trained and evaluated over large object datasets by fusing visual information with high-resolution tactile signals extracted from a simulation.

A new research branch involves exploring an unknown object by sliding the sensor along it, rather than relying solely on simple touches. For instance, Zhaot et al. [25] used only tactile sensors, without a depth camera, to predict the pose and reconstruct the object. The tactile sensor was slid across the face of the object of interest. Chen et al. [26] recently combined a single-depth image with four tactile fingers arranged like a hand's fingers to reconstruct the shape of an object's unseen part. They achieved this by sliding the sensors along the object's face.

Detecting the symmetry of a 3D object can be applied to various tasks ([27]–[30]). Current research can be classified into different groups based on the type of symmetry being sought: exact [31] or approximate symmetry [32], local [33] or global [34] symmetry and extrinsic ([31], [35]–[38]) or intrinsic ([39], [40]) symmetry. However, previous studies have encountered difficulties in predicting 3D symmetry when dealing with incomplete shape information about an object. Therefore, the research community has focused on finding a solution to this problem. The most relevant and recent studies are those that aim to predict the 3D symmetry plane of incomplete objects using a single-view RGB-D image ([41], [42]). Xi et al. ([42]) developed an end-to-end pipeline using deep neural networks to predict reflection and rotational symmetries of 3D objects in a single RGB-D image. The pipeline also outputs the symmetric counterpart for each 3D point based on the predicted symmetry.

III. Material and Methods

An overall structure of all the steps explained below can be seen in Figure 1.

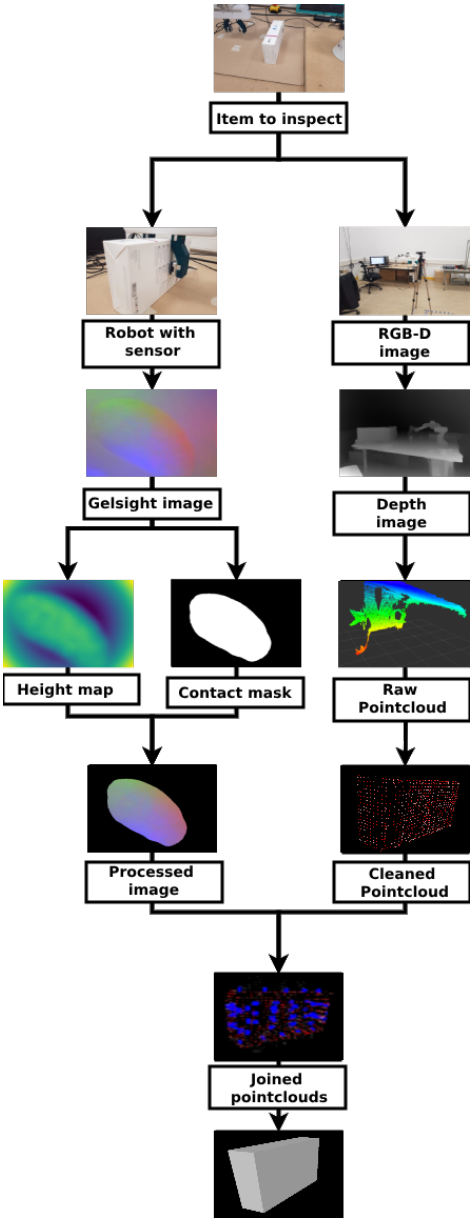

Fig. 1: General diagram of the proposed method. Left branch: analysis using the visuo-tactile sensor, from taking the image to obtaining the height map and contact mask to finally keeping only the contact point height map. Right branch: RGB-D camera data process, from gathering the image to the process of the point cloud. At the end, both parsed point clouds are merged before reconstructing the object.

The ultimate aim of this project is to reconstruct a 3D mesh model of an object by combining information from various sensors. To achieve this, the methodology involves capturing a depth image of the selected object using a calibrated depth camera, followed by taking measurements of the non-visible parts using GelSight Mini sensors. After capturing all the images, proceed to process them to extract the important information, which in this case is the point clouds.

After processing all the images, the aim is to use the point cloud extracted from the depth camera in conjunction with deep learning to infer the symmetrical axis of the object. Once the axis has been predicted, the point cloud is mirrored along it to achieve a complete object representation. The tactile measurement point clouds are then added to the total representation, resulting in a complete mesh with additional point clouds on the mirrored side. Once the mirrored point clouds that are on the same height axis as the tactile measurements have been removed, all the point cloud information is gathered. The object is then reconstructed to obtain a 3D mesh, which theoretically provides a more precise representation of the initial object.

A. Datasets

For this work, two generic datasets were used for evaluation, and a further custom dataset was constructed using real objects:

Yale-CMU-Berkeley dataset (YCB) [43]: The dataset includes mesh models, RGB, RGB-D and point cloud images of over 80 objects. The physical objects are also available through the YCB benchmarking project. The data is collected by two state-of-the-art systems: UC Berkeley's scanning rig and the Google scanner. The UC Berkeley scanning rig data provides meshes generated with Poisson reconstruction, meshes generated with volumetric range image integration, textured versions of both meshes, Kinbody files for using the meshes with OpenRAVE, 600 high-resolution RGB images, 600 RGB-D images, and 600 point cloud images for each object. The Google scanner data provides 3 meshes at different resolutions (16k, 64k, and 512k polygons), textured versions of each mesh, and Kinbody files for using the meshes with OpenRAVE.

YCB-Sight dataset [12]: This dataset contains ground truth meshes from the YCB dataset, GelSight sensor images from the interaction, sensor poses and a depth map from a camera. All objects in the dataset have different geometries, such as curved, rectangular and complex. This dataset is divided into two parts: GelSight-object interactions using Taxim, an example-based tactile simulator. It simulates 60 uniformly distributed sensor positions on each object, normal to the local surface of the mesh. It renders a depth map from the perspective of an overlooking camera using Pyrender [44]. Finally, zero mean Gaussian noise is added to the tactile point clouds, sensor poses and depth maps. The simulated data subset contains information from 30 objects in the YCB dataset. The authors simulated 60 uniformly distributed sensor poses on each object, normal to the local surface of the mesh. They rendered a depth map from the perspective of an overlooking camera using PyRender. They also added zero-mean Gaussian noise to the tactile point clouds, sensor poses and depth maps to make the data look more realistic. The real-life data subset contains information from six objects in the YCB dataset. The authors used a UR5e 6-DoF robotic arm with the GelSight sensor mounted on a WSG50 parallel gripper. The depth map was captured using a fixed-pose, calibrated Azure Kinect, approximately 1 meter from the object.

Real life items: For the sake of simplicity, the objects used in the real experiment were simple geometric figures, such as a cube, a rectangular prism and a pyramid. These objects were manually modeled using Blender [45], a free and open-source 3D creation toolkit. The 3D printed objects used can be seen in Fig. 2. Once the 3D model was ready, the objects were 3D printed in the lab. With the ability to 3D print the objects, we were able to test them with any object shape we could think of.

The material used to 3D print the objects, PLA, was light and had a high reflection rate. We simply decided to cover it with black tape so that the light from the RGB-D camera would not be reflected and the point cloud of the object could be easily extracted with more detail.

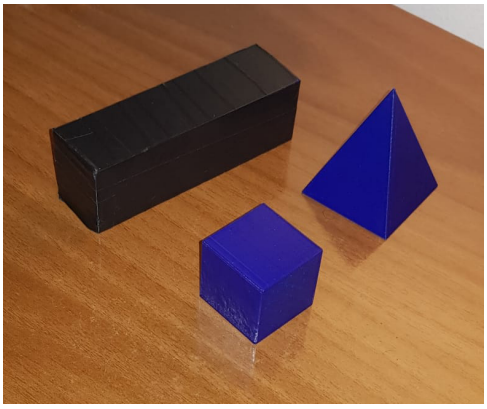

Fig. 2: Several 3D printed objects belonging to the real life items dataset.

In order to manipulate and get data from real-life objects, the robotic arm Franka Panda Emika from Franka Robotics [46] was used. Regarding the camera used, it is the Astra Orbbec 3D, an RGB-D camera from Astra optimized for long-range use cases, making it ideal for interactive systems, entertainment, retail, and robotics.

More details of the arm and camera used in the experiment can be found in the appendix A.

B. Extraction of point clouds from the object from an RGB-D image.

A partial frontal view of the object is required and, as mentioned, this is obtained using a depth camera that provides RGB-D information. The perfect scenario would be to place the object in a closed box so that the only information present in the image is the object, but unfortunately this is not the case, so the image contains information about the background, the table, etc.. Therefore, once the image has been captured, it needs to be cleaned and processed in order to obtain the final point cloud that contains only the information about the object that is useful for reconstruction. Also, as can be seen in more detail in the appendix to this document, the camera must be stationary in one place in order to extract and calculate various parameters that are also needed to obtain additional information, such as the location. Once the setup has been correctly set up for the start of the experiment, the image can be taken.

This subsection explains all four blocks from the right branch of Fig. 1. So let's start by explaining how the initial setup was put together. First, after checking the camera's minimum and maximum working distances, we marked a spot on the floor where we would place the tripod and leave it untouched. Then it was time to place the subject; for this we simply drew a spot on the table so that we knew where to place the subject, always in the same place.

Once the object and the camera were in place, it was time to calibrate the camera for both intrinsic and extrinsic parameters in order to extract the most suitable image; this calibration process can be found in the Appendix. Having calibrated the camera, the next step was to acquire and store a depth image of the object. To do this, RVIZ was used in conjunction with ROS, which provided the ability to not only take an image, but to directly visualize and store the point cloud of the setup that the camera was looking at. So it was just a matter of storing the point cloud for later processing. The results of cleaning the point cloud can be seen in Fig. 3. After creating a file with the extension ".pcd" and the point cloud of all the environments, including the object, Python was used to process it.

Using the Open3D library [47], all points not belonging to the object were removed and only the object's points were saved and stored for later use. More details of the processing pipeline can be found in the Github repository.

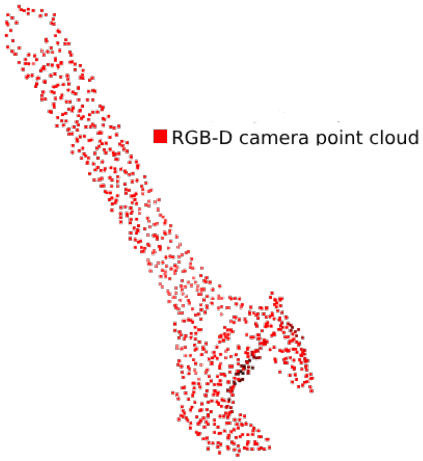

Fig. 3: A cleaned depth view point cloud was obtained after getting the depth image, transforming it into a point cloud, and cleaning the point cloud.

C. Extraction of images from the object by means of a GelSight visuo-tactile sensor.

Since the experimental setup was controlled, all distances from the camera to the object were known, so in order to extract images from the GelSight sensor, it was as simple as preparing a Python script to capture the image that the sensor provided and only store the image when the sensor was in contact with the object, and then manually annotate the position and rotation of the sensor with respect to the camera when the image was taken. This subsection explains the first two blocks of the left branch of Fig. 1.

The GelSight Mini sensor (shown in Fig. 4) from Gelsight [48] is a revolutionary type of tactile sensor due to its unique design. Its main element is a gel film used as a sensing element. This gel film deforms in response to external forces, creating a reflection of the shape and texture of the object being touched. A high-resolution camera is positioned above the gel to capture the deformation, along with three different LED light sources positioned to provide perfect illumination of the gel film, allowing the sensor to reconstruct a 3D representation of the object with excellent accuracy.

Fig. 4: GelSight Mini sensor used in this work.

The Gelsight Mini sensors can be plugged directly into a computer, allowing images to be extracted from them, which is easily done using Python. In addition to the ability to interact directly with specific code, Gelsight also provides an easy-to-use web demo for users to get a first look at the sensor's capabilities.

As explained earlier, there are several ways to get an image of the GelSight Mini sensors, but mainly two: using the web demo provided by GelSight, or by developing Python code. For this project, the Python option was used, as Python can also work with ROS, allowing it to be integrated with the Franka Panda Emika robotic arm from Franka Robotics [46] to extract the necessary metadata from the image, such as location information. As the sensor is connected via a USB connector, just like a webcam, it is possible to use a library such as OpenCV [49] to directly capture the image, as if it were any other type of device.

Training deep learning algorithms, such as those used in the development of this work, requires a lot of data. Therefore, in order to train the depth and contact mask estimators, it is essential to be able to capture a large number of images with the GelSight sensor and also extract the metadata required for later post-processing, such as the touch location. The Taxim simulator was used for this purpose [50]. The Taxim Simulator is a realistic and fast simulation model for the GelSight vision-based tactile sensor. It provides an example-based method for simulating the GelSight sensors. Taxim simulates the optical response to deformation using a polynomial look-up table (see Fig. 5. This table maps the contact geometries to the pixel intensity sampled by the embedded camera. It uses a linear displacement relationship and the superposition principle to simulate the movement of the surface marker caused by the stretching of the elastomer surface.

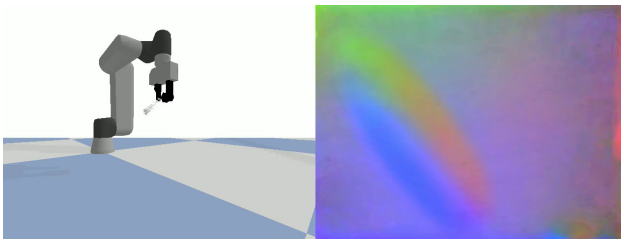

Fig. 5: Taxim simulator. Left: robotic arm grasping an object while equipped with GelSight sensors. Right: image captured from the GelSight sensor while grasping.

In addition to the image from the GelSight sensor, Taxim also allows the user to obtain the metadata of the contact point, which, as explained, is very useful to know the exact position and rotation of the contact image, which is essential in the post-processing part of the work.

D. Recovery of contact mask and depth map from GelSight images.

Visuo-tactile sensors use images as the data collection mechanism. The GelSight sensor, as mentioned above, captures an image of the deformed gel pad in contact with the object surface. The important information for this project that needs to be extracted from these sensor images is that only those pixels of the image that are in contact with the object surface can be used to predict critical information, such as the depth map.

We will use the following nomenclature to represent the information that needs to be extracted from the image:

$$\textbf{Image} \rightarrow I$$

$$\textbf{Contact Mask} \rightarrow CM$$

$$\textbf{Height Map} \rightarrow HM$$

$$\textbf{I} \rightarrow CM, HM$$

Currently, the GelSight Mini sensors use a standard lookup table method [51] to extract this information from the images. This is provided by the developers of the sensor. This lookup table method uses mathematics to map the 2D images directly into 3D point clouds. There are some approaches in the research community to compute height maps from simple 2D images using a supervised approach that takes advantage of deep learning capabilities. Following this idea of predicting the height map from a simple image, we propose to use a modified implementation of [52] to compute CM and HM from the 2D sensor image. The proposed architecture has two distinct parts. The first is the encoder, which uses a ResNet-50 architecture, and the second part is the decoder, which implements up-sampling blocks to output a final image of the input size and information. The final output is then up-scaled to the desired image size of 640x480. As the image has been up-scaled from a latent space to the output size, some small artifacts have appeared, so a simple Gaussian filter has been applied at the end of the network to blur these artefacts and achieve better results.

Since it is necessary to compute both CM and HM, two identical models with the same network architectures were trained using the same approach. Using tactile images from the GelSight sensor as input, one of the networks was tasked with producing predictions for HM as shown in Fig.6 up, while the other, with the same architecture, was trained to predict CM as shown in Fig. 6 down. This subsection explains the two parallel blocks in third position and the fourth block from the right branch of Fig. 1.

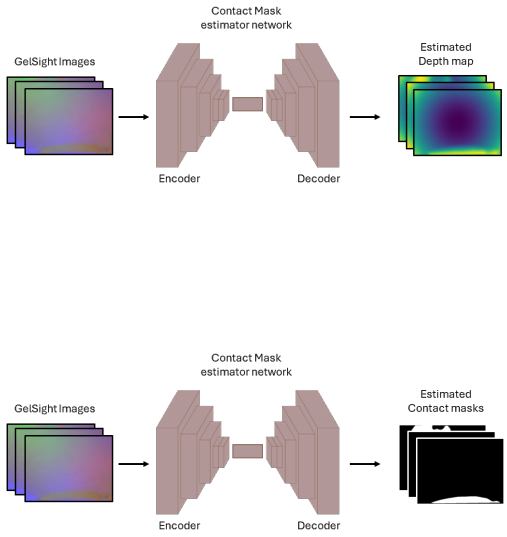

Fig. 6: Up: Height map (HM) estimator. Down: Contact mask (CM) estimator.

Data from the YCB-Sight dataset was used to train both models. This dataset contains 30 household objects from the YCB dataset, which were used to generate 60 images of each object using the Taxim simulator. The data split used for training was 21 items from training, 6 for validation and 3 for testing. As mentioned, each item contained 60 images, giving a total of 1,260 images for training, 360 for validation and 180 for testing. Once both networks had been trained and evaluated, all the data was used to train the models from scratch in order to have as much information as possible for the models to learn, so a total of 1,830 input images were used in the final training stage.

E. Reflection of the input frontal view of the object to have a higher level of information about it.

Most of the objects we interact with are completely or partially symmetrical, so if we could predict the axis of symmetry from a partial view of an object, we could simply mirror it to take the invisible back part and thus get an almost perfect representation of the object's shape. Two problems arise from this intuition, the first is how to determine this symmetry axis using only information from one part of the object, which is not an easy task to perform since the information about the object is sometimes sparse. The other problem is that we assume that the object is 100% symmetric and that the predicted symmetry axis is accurate. To minimize the possible error, we can use a robotic arm equipped with visuo-tactile sensors to locally recover information about the non-visible part of the object by exploring it and storing the contact images. With these images, after processing and extracting the point cloud information, we can modify the predicted point cloud structure to end up with a higher fidelity point cloud of the object and then finally reconstruct the 3D mesh from it.

This section therefore provides information on how the symmetry axis of a partially viewed object is predicted and used to reflect the existing point cloud of the object. To accomplish such a task, an extensive review of the current state of the art was conducted. We decided to choose SymetryNet [53], an open source project with code available so that integration would be much easier, as the methodology of how the symmetry axis is predicted is not the aim of this project. SymmetryNet (see scheme in Fig. 7) is a 2020 project that aims to learn to predict the reflection and rotation symmetries of 3D shapes from single-view RGB-D images. The goal of this research is to directly train a neural network to predict the symmetry axis of an object using only a depth image as input. The main problem I encountered is that it does not generalize; it only works decently with known objects, which is not the perfect scenario. It was therefore, only useful in certain cases. Objects that were similar to the training set got a good prediction by referring to the symmetry axis, thus allowing the possibility of mirroring the visible point cloud on the back, but most of the objects that were completely unknown could not be mirrored correctly due to a bad symmetry axis prediction.

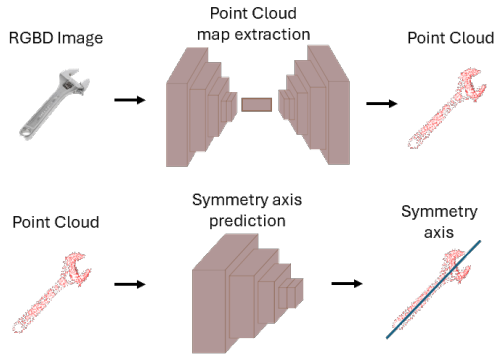

Fig. 7: Scheme of SymmetryNet [53]. An open-source neural network architecture to predict symmetric axis from only one depth image.

Once the symmetry axis is obtained, since we have the partial point cloud, it is as simple as moving all the points to the other side of the axis, maintaining the distance from the original point to the symmetry axis, but in the opposite direction. This process was

also done using Python. Basically, using the Open3D library, we are able to create a 3D space on which we can place the known point cloud, draw the symmetry axis for visual analysis and, using a simple algorithm that can be found in the appendix, mirror the input point cloud.

F. Addition of point cloud from tactile measurements.

Once the visible point cloud has been mirrored and we have a complete representation of the shape of the object, it is time to integrate the point clouds from the tactile measurements into the playground. This subsection explains the penultimate block from 1. As we said, when we took the tactile images we stored information about the rotation and location of those measurements, so it is as simple as aligning the processed point cloud from each measurement with those notations, so we end up with a complete point cloud with some extra patches of point clouds from the tactile measurements as seen in Fig. 8.

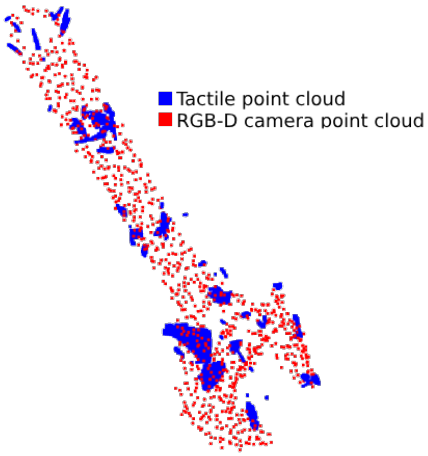

Fig. 8: Depth view plus tactile point clouds. This image shows the result from joining the point cloud extracted from the single RBG-D image and the measurements taken with the GelSight sensor.

G. Elimination of the points from the point cloud that are on the same axis as the point clouds of the tactile sensors.

At this stage, we have all the point cloud information collected on the same object, but some of it is redundant or incorrect. As mentioned in the previous sections, we have predicted what the shape of the object would be in the non-visible part, but now with the tactile information of that non-visible part, we can modify this prediction so that it is more accurate. To do this, we basically need to calculate the centre of the object and all the vectors from each point in the direction of that center. The idea is to use this vector information from the tactile measurements and remove any point that is not in the tactile image that passes through this imaginary line, thus deleting only those points of the mirrored point cloud that, as predicted, do not exist in the shape of the real object. The resulting points to be removed after doing this procedure can be seen in Fig. 9 and Fig. 10 shows the complete cloud.

H. Reconstruction of the object mesh from the point cloud.

Once all the point clouds have been processed and gathered together as seen in Fig. 10 its time to 3D reconstruct the object. There are various approaches to reconstructing an object's mesh from a set of point clouds, from more traditional methods based on pure mathematics to more innovative and emerging methods that use neural layers to reconstruct the meshes. This subsection, explains the last block from Fig. 1. Since the reconstruction algorithm is not the focus of this work, we decided to use mainly traditional methods. Two of them were used: Ball Pivoting and Poisson Surface Reconstruction. The Ball Pivoting Algorithm (BPA) [54] is a surface reconstruction method related to alpha shapes. Intuitively, think of a 3D ball with a given radius that we drop on the point cloud. If it hits any three points (and does not fall through those three points), it creates triangles as illustrated in Fig. 11. Then the algorithm starts to rotate around the edges of the existing triangles, and every time it hits three points that the ball does not pass through, we create another triangle.

The Poisson surface reconstruction method [55] solves a regularized optimization problem to obtain a smooth surface as illustrated in Fig. 12. This method produces non-smooth results because the points of the point cloud are also the vertices of the resulting triangle mesh without any modification.

Fig. 9: Points to remove from the mirrored point cloud since they are redundant due the new points which are more precise from the GelSight sensors.

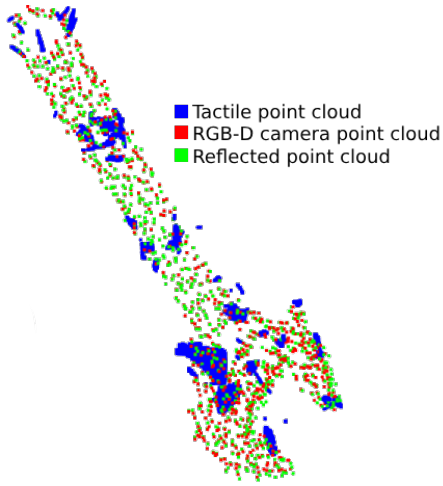

Fig. 10: Complete point cloud containing all the data from GelSight sensors, depth camera and mirrored point clouds. This is the last step before performing the mesh reconstruction.

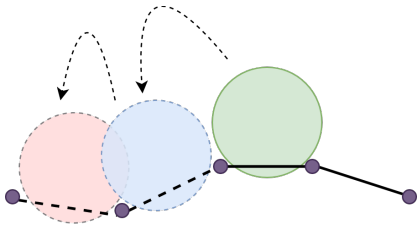

Fig. 11: Schematic representation of Ball Pivoting reconstruction Algorithm (BPA) [54].

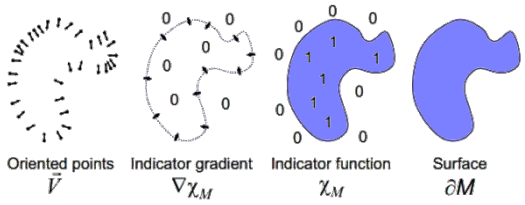

Fig. 12: Schematic representation of the Poisson Surface Reconstruction (PSR) algorithm [55].

IV. Results.

This section contains the results obtained both for the GelSight image depth map and contact mask estimator and the final reconstruction of the object after obtaining the cleaned point cloud. The first subsection explains the first part, making use of the YCB dataset explained in section III-A. The second subsection contains the results obtained for the mesh reconstruction.

A. Evaluation of the GelSight image depth map and the contact mask estimator.

The height maps obtained by the models were compared with the standard look-up-table procedure, which basically maps 2D tactile images onto gradients of local shape and uses fast Poisson integration to generate the resulting height map using RMSE (root mean squared error) as the numerical metric. The contact mask predictions are compared to the contact masks generated by intensity thresholding of contact and non-contact images using the Jaccard index (IoU) and the cube similarity, also known as the F1 score, which will be described next:

RMSE (Root mean squared error) [56]: is one of the most commonly used metrics in regression models. It is used to measure the average size of the errors between the predicted and true values. The smaller the value of RMSE, the better; it indicates that the model is performing better as it represents a smaller prediction error on average. It is important to note that the RMSE is sensitive to outliers, as it can be heavily influenced by large errors.

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

where y_i represent the predicted values, x_i the ground truth value, and n the number of samples.

IoU (Intersection over Union) [57]: is a metric mainly used in image segmentation algorithms. It calculates the overlap between the predicted bounding box or segmented region and its ground truth values, providing a measure of how well the predicted mask or bounding box matches the actual target. The general idea of the IoU metric is shown in Fig. 13.

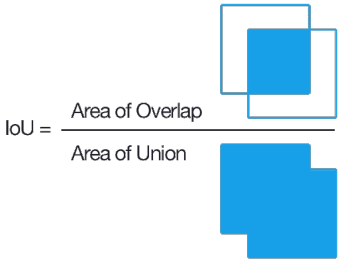

Fig. 13: Intersection over union scheme.

Fig. 14 shows the obtained results by our custom FCRN architecture, using box plots, with respect to the benchmarks on the YCB-Sight dataset. As explained, the contact mask is compared in terms of IoU while the height map is compared with pixel-wise RMSE.

For height map estimation, the custom FCRN outperforms the benchmark with an average RMSE of 0.1370 over the 30 objects with 60 images in each object from the YCB-Sight dataset, while the benchmark method obtained an average RMSE of 5.1430.

For the contact mask estimator, the average IoU obtained by the custom FCRN network is 0.3320, while the image thresholding method used as a baseline obtained results of 0.7740, which is lower.

B. Object reconstruction experiments.

In order to test the complete pipeline, several experiments were carried out using both real objects and existing datasets. It is important to mention that the majority of the experiments were carried out using the YCB-Sight dataset, as the data collection process was already completed, making it easier to start testing the pipeline right away. As a reminder, the objective of the pipeline is to obtain the 3D model of the object using different sensory data: a single image from an RGB-D camera and several images from the GelSight Mini visuo-tactile sensors. Therefore, the pipeline needs to be evaluated in terms of how accurate the final reconstructed model is compared to the ground truth. To perform such an evaluation, the Chamfer Distance [58] and Intersection Over Union (IoU) are used, which are among the most commonly used metrics for comparing object shape similarity.

Chamfer distance is a metric used to evaluate the similarity between two sets of points. Let's say we have two sets of points, **X** and **Y**, the Chamfer distance is the sum of the distances from each point in **X** to its nearest neighbor in **Y**, plus the sum of the distances from each point in **Y** to its nearest neighbor in **X**:

$$CD(S_1, S_2) = \frac{1}{|S_1|}\sum_{x \in S_1} \underset{y \in S_2}{min} \|x - y\|_2^2 + \frac{1}{|S_2|}\sum_{y \in S_2} \underset{x \in S_1}{min} \|x - y\|_2^2$$

Chamfer distance can be used in a variety of applications, including computer vision, robotics and computer graphics. It is particularly useful in point cloud registration, where the main objective is to perform an alignment between two or more point clouds. By minimizing this metric between the two sets of points, the best alignment between the two sets can be achieved.

1) Experiments with the YCB-Sight dataset.: The YCB-Sight dataset contains 30 objects from the YCB dataset. In order to evaluate the method, all 30 objects were reconstructed using all explained methods and then evaluated as explained. The reconstructed object mesh is compared with the ground truth objects of the YCB dataset. In order to visually see the numerical results obtained for all the metrics in the YCB dataset, Table I was created so that all the information is compacted and easily comparable.

Looking at Table I, the numerical results are not really good; the method worked, but not as well as expected. Certainly the results

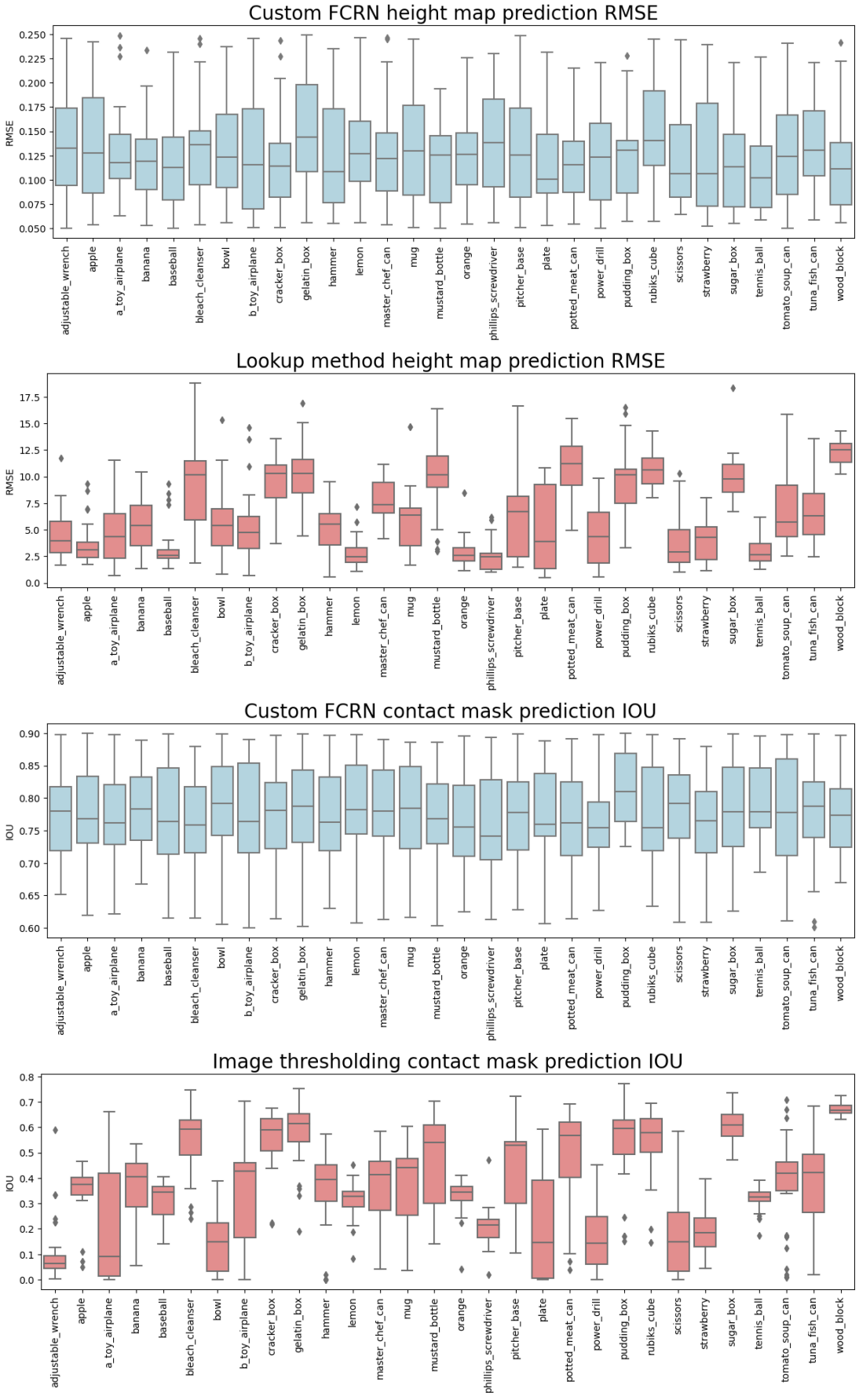

Fig. 14: RMS (1-2 rows) and IoU (3-4 rows) results in box plots of the baseline method and ours.

TABLE I: Results of 3D object reconstruction. The first column contains the Chamfer distance for each object, the smaller the value the better the result. The second and third columns contain the 3D IoU for both reconstruction methods used, Ball Pivoting Algorithm (BPA) and Poisson Surface Reconstruction (PSR), the higher the value the better the result.

Object	Chamfer distance (mm^2 x 10)	IoU BPA (*100)	IoU PSR (*100)
adjustable_wrench	0.3518	71.5185	71.5665
apple	0.5086	66.0864	64.3797
a_toy_airplane	**1.3765**	**40.2349**	**39.2959**
banana	0.5924	73.9246	74.6922
baseball	0.4384	70.3840	68.6098
bleach_cleanser	0.1332	68.3320	66.7522
bowl	0.4185	66.1856	67.5763
b_toy_airplane	0.9793	45.2062	46.6863
cracker_box	0.4873	71.8731	70.7660
gelatin_box	0.3919	68.9194	69.0716
hammer	0.7241	71.2413	69.2690
lemon	0.1366	64.3660	64.4303
master_chef_can	0.2355	67.3554	68.0025
mug	0.2201	62.2016	61.3412
mustard_bottle	0.2907	70.9708	69.2043
orange	0.2084	68.0842	66.4811
phillips_screwdriver	0.7700	70.7003	70.8076
pitcher_base	0.1928	62.9287	61.2214
plate	0.1922	69.9224	71.0908
potted_meat_can	0.4690	67.6902	65.7796
power_drill	1.1029	42.9701	43.9651
pudding_box	0.4649	64.6490	62.8894
rubiks_cube	**0.0600**	**74.6009**	**74.0001**
scissors	0.7023	70.0231	68.8181
strawberry	0.0842	63.842	62.4980
sugar_box	0.2195	63.1955	61.5153
tennis_ball	0.4897	72.8979	73.8733
tomato_soup_can	0.9330	48.6690	48.4742
tuna_fish_can	0.3508	70.5089	70.2337
wood_block	0.2496	64.4969	66.3424
Mean	**0.4593**	**64.7660**	**64.2878**

can be improved with more trial and error, especially in the way the reflection axis is calculated, i.e. the point cloud is mirrored, as it did not generalize perfectly for all the different object shapes.

Looking more closely at the table, we can see that the model with the best metrics is the Rubik's cube with a Chamfer distance of 0.06 and an IoU for both methods close to 74%. This is mainly because the shape of a Rubik's cube is really simple; it is basically a cube, which can be seen in Fig. 15. Such a simple shape makes it easy to calculate the reflection axis and thus predict the invisible part.

On the other hand, the worst results were obtained by the a_toy_airplane that can be seen in Fig. 16. The computed Chamfer distance is 1.3765, while the IOU for both methods is 40 on average. This low score is mainly due to its more complex shape; even though the object is symmetric, being able to compute efficiently a symmetrical axis is much more complex, thus the resulting point cloud is less complete and contains wrong points that are not part of the real figure.

Finally, after calculating the mean of all the metrics, we can

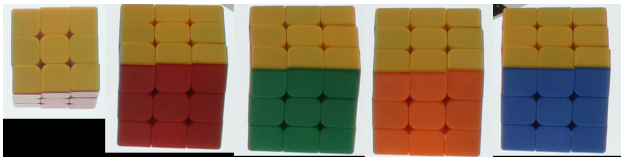

Fig. 15: Rubik's Cube object from the YCB dataset. Object that achieved the best results in all metrics.

Fig. 16: A toy aeroplane object from the YCB dataset. Item that scored lower on all metrics.

estimate values of 0.4593 for the Chamfer distance, 64.7660 for the IoU using the BPA algorithm, and finally 64.2878 for the PSR algorithm. Comparing the two reconstruction algorithms, the BPA performs better, but by a small amount, so it is clear that what lacks confidence is the point cloud acquisition, not the reconstruction algorithms used to obtain the final mesh of the object.

2) Experiments with real-life setup: All the procedures for extracting data from the real setup were prepared and the camera was calibrated using the method described in the Appendix. A program was written to extract images from the gel sight sensor so that the information from the sensors could also be taken. The only thing missing was the ability to automatically detect the position of the object and move the arm so that the touch was autonomous. This is a very time-consuming task, as the automatic manipulation of the robot is not a straightforward implementation.

To finish with the results section, trying to tackle this whole problem, which contains gathering data from different sensors, parsing them, creating new methods to extract and clean the data for the final objective of combining all the data for reconstructing a 3D object's mesh is hard to encapsulate only in the hours available for a final master thesis project.

Pictures of the real setup can be found in the Appendix.

V. Conclusions and Future Work

This work implements a methodology to combine information from a single-view RGB-D camera and several visuo-tactile sensors to reconstruct 3D models of real objects by mirroring the depth camera information and filling the rest with tactile measurements.

While a final reconstructed object was achieved, with the best results for basic shapes, the reconstruction results were less satisfactory than expected, as it was initially believed that by combining information from both sensory modalities a truly accurate 3D model could be achieved. The results were particularly poor for objects with more complex shapes, where the axis of symmetry was harder to find or simply absent. Finding the optimal symmetry axis was a key aspect of the implemented method as it relied heavily on its existence.

Despite the efforts made, it was not possible to carry out real experiments with the laboratory robots due to time constraints. I did not have enough time to perform experiments and validate the method with real objects. My intuition tells me that the results would be worse than with the online dataset, mainly due to the simulation of the real gap that is usually present in all kinds of robotic applications.

In the future, in order to improve the reconstruction algorithm, the key challenge would be to try to obtain a better performing model for calculating the symmetry axis from a figure. A model that has the

ability to generalize to a wider set of objects. Also, by implementing different algorithms or techniques to perform active exploration of the objects, the number of tactile contacts could be reduced while achieving the same or even higher accuracy, resulting in an overall reduction in time.

In conclusion, although the results were lower than expected, the overall impression is that the method could really work, making it promising for future development in obtaining high-fidelity 3D models of objects without relying solely on a static camera, which, as explained, has several limitations.

VI. Acknowledgments

Thanks to the scholarship program named "Concurso Público de Méritos, para la contratación de personal investigador que se inicie en la labor investigadora dentro de los correspondientes programas científicos de los centros singulares de la USC" that helped to develop the final master thesis. This scholarship started on January 17, 2023, and ended on October 31, 2023. The conditions that I had during this scholarship were helpful in facilitating the progress of this project, especially giving me free time to dedicate to the project while not having to work on other projects.

Also, thanks to both my supervisors, Juan Antonio Corrales Ramón and María José Carreira Nouche, for the support, guidance, and help that they have given me during all the development of the project and also for giving me the opportunity to work with them, which was a profitable experience in my professional career.

References

[1] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[2] Hannah Helbig and Marc Ernst. Optimal integration of shape information from vision and touch. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 179:595–606, 07 2007.

[3] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian H. Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, Dinesh Jayaraman, and Roberto Calandra. DIGIT: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *CoRR*, abs/2005.14679, 2020.

[4] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward H. Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. *CoRR*, abs/1803.00628, 2018.

[5] Shaoxiong Wang, Yu She, Branden Romero, and Edward H. Adelson. Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger. *CoRR*, abs/2106.08851, 2021.

[6] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12), 2017.

[7] Wenqiang Xu, Zhenjun Yu, Han Xue, Ruolin Ye, Siqiong Yao, and Cewu Lu. Visual-tactile sensing for in-hand object reconstruction, 2023.

[8] Jacob Varley, Chad DeChant, Adam Richardson, Avinash Nair, Joaquín Ruales, and Peter K. Allen. Shape completion enabled robotic grasping. *CoRR*, abs/1609.08546, 2016.

[9] Marten Björkman, Yasemin Bekiroglu, Virgile Högman, and Danica Kragic. Enhancing visual perception of shape through tactile glances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3180–3186, 2013.

[10] Gabriela Zarzar Gandler, Carl Henrik Ek, Mårten Björkman, Rustam Stolkin, and Yasemin Bekiroglu. Object shape estimation and modeling, based on sparse gaussian process implicit surfaces, combining visual data and tactile exploration. *Robotics and Autonomous Systems*, 126:103433, 2020.

[11] Jarmo Ilonen, Jeannette Bohg, and Ville Kyrki. Three-dimensional object reconstruction of symmetric objects by fusing visual and tactile sensing. *The International Journal of Robotics Research*, 33(2):321–341, October 2013.

[12] Sudharshan Suresh, Zilin Si, Joshua G. Mangelson, Wenzhen Yuan, and Michael Kaess. Efficient shape mapping through dense touch and vision. *CoRR*, abs/2109.09884, 2021.

[13] Shaoxiong Wang, Jiajun Wu, Xingyuan Sun, Wenzhen Yuan, William T. Freeman, Joshua B. Tenenbaum, and Edward H. Adelson. 3d shape perception from monocular vision, touch, and shape priors. *CoRR*, abs/1808.03247, 2018.

[14] Edward J. Smith, Roberto Calandra, Adriana Romero, Georgia Gkioxari, David Meger, Jitendra Malik, and Michal Drozdzal. 3d shape reconstruction from vision and touch. *CoRR*, abs/2007.03778, 2020.

[15] Edward J. Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero, and Michal Drozdzal. Active 3d shape reconstruction from vision and touch. *CoRR*, abs/2107.09584, 2021.

[16] Peter Allen and Ruzena Bajcsy. Object recognition using vision and touch. pages 1131–1137, 01 1985.

[17] Peter K. Allen. Integrating vision and touch for object recognition tasks. *The International Journal of Robotics Research*, 7(6):15–33, 1988.

[18] P.K. Allen and P. Michelman. Acquisition and interpretation of 3-d sensor data from touch. *IEEE Transactions on Robotics and Automation*, 6(4):397–404, 1990.

[19] Sascha Fleer, Alexandra Moringen, Roberta L. Klatzky, and Helge J. Ritter. Learning efficient haptic shape exploration with a rigid tactile sensor array. *CoRR*, abs/1902.07501, 2019.

[20] Nawid Jamali, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Active perception: Building objects' models using tactile exploration. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 179–185, 2016.

[21] Danny Driess, Peter Englert, and Marc Toussaint. Active learning with query paths for tactile object shape exploration. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 65–72, 2017.

[22] Zhengkun Yi, Roberto Calandra, Filipe Veiga, Herke van Hoof, Tucker Hermans, Yilei Zhang, and Jan Peters. Active tactile object exploration with gaussian processes. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930, 2016.

[23] Takamitsu Matsubara and Kotaro Shibata. Active tactile exploration with uncertainty and travel cost for fast shape estimation of unknown objects. *Robotics and Autonomous Systems*, 91:314–326, 2017.

[24] Alexander Bierbaum, Matthias Rambow, Tamim Asfour, and Rudiger Dillmann. A potential field approach to dexterous tactile exploration of unknown objects. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pages 360–366, 2008.

[25] Jialiang Zhao, Maria Bauza, and Edward H. Adelson. Fingerslam: Closed-loop unknown object localization and reconstruction from visuo-tactile feedback. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8033–8039, 2023.

[26] Yiting Chen, Ahmet Ercan Tekden, Marc Peter Deisenroth, and Yasemin Bekiroglu. Sliding touch-based exploration for modeling unknown object shape with multi-fingered hands, 2023.

[27] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 560–568, New York, NY, USA, 2006. Association for Computing Machinery.

[28] Joshua Podolak, Aleksey Golovinskiy, and Szymon Rusinkiewicz. Symmetry-enhanced remeshing of surfaces. In *Symposium on Geometry Processing*, July 2007.

[29] S. Thrun and B. Wegbreit. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1824–1831 Vol. 2, 2005.

[30] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J. Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. Graph.*, 29(4), jul 2010.

[31] Yaron Lipman, Xiaobai Chen, Ingrid Daubechies, and Thomas Funkhouser. Symmetry factored embedding and distance. *ACM Trans. Graph.*, 29(4), jul 2010.

[32] Mark Pauly, Niloy J. Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. Graph.*, 27(3):1–11, aug 2008.

[33] Natasha Gelfand and Leonidas J. Guibas. Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, page 214–223, New York, NY, USA, 2004. Association for Computing Machinery.

[34] Vladimir Kim, Yaron Lipman, Xiaobai Chen, and Thomas Funkhouser. Möbius transformations for global intrinsic symmetry analysis. *Computer Graphics Forum (Symposium on Geometry Processing)*, 29(5), July 2010.

[35] Lin Gao, Ling-Xiao Zhang, Hsien-Yu Meng, Yi-Hui Ren, Yu-Kun Lai, and Leif Kobbelt. PRS-net: Planar reflective symmetry detection net for 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3007–3018, jun 2021.

[36] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François X. Sillion. Accurate detection of symmetries in 3d shapes. *ACM Trans. Graph.*, 25(2):439–464, apr 2006.

[37] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[38] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. Symmetry detection using feature lines. *Computer Graphics Forum*, 28(2):697–706, April 2009.

[39] Kai Xu, Hao Zhang, Andrea Tagliasacchi, Ligang Liu, Guo Li, Min Meng, and Yueshan Xiong. Partial intrinsic reflectional symmetry of 3d shapes. 28(5):1–10, dec 2009.

[40] Dan Raviv, Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Full and partial symmetries of non-rigid shapes. *International Journal of Computer Vision*, 89(1):18–39, February 2010.

[41] Yifei Shi, Xin Xu, Junhua Xi, Xiaochang Hu, Dewen Hu, and Kai Xu. Learning to detect 3d symmetry from single-view rgb-d images with weak supervision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4882–4896, 2023.

[42] Yifei Shi, Junwen Huang, Hongjia Zhang, Xin Xu, Szymon Rusinkiewicz, and Kai Xu. Symmetrynet: Learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images, 2020.

[43] Ycb dataset. https://www.ycbbenchmarks.com/. Last accessed: 01-01-2024.

[44] Pyrender. https://pyrender.readthedocs.io/en/latest/. Last accessed: 01-01-2024.

[45] Blender. https://www.blender.org/. Last accessed: 01-01-2024.

[46] Franka Robotics. https://franka.de/. Last accessed: 01-01-2024.

[47] Open3d. https://www.open3d.org/. Last accessed: 29-10-2023.

[48] Gelsight. https://www.gelsight.com/. Last accessed: 01-01-2024.

[49] Opencv. https://opencv.org/. Last accessed: 29-10-2023.

[50] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 2022.

[51] Gelsight mini github repository. https://github.com/gelsightinc/gsrobotics. Last accessed: 01-01-2024.

[52] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks, 2016.

[53] Yifei Shi, Junwen Huang, Hongjia Zhang, Xin Xu, Szymon Rusinkiewicz, and Kai Xu. Symmetrynet: Learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images, 2020.

[54] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October 1999.

[55] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, 2006.

[56] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, June 2014.

[57] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression, 2019.

[58] Harry G. Barrow, Jay M. Tenenbaum, Robert C. Bolles, and Helen C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International Joint Conference on Artificial Intelligence*, 1977.

[59] Ros. https://www.ros.org/. Last accessed: 29-10-2023.

[60] Openni. https://github.com/OpenNI/OpenNI. Last accessed: 29-10-2023.

[61] Opencv camera calibration intrinsic. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Last accessed: 29-10-2023.

APPENDIX

A. Camera Calibration

Cameras are the sensors that capture images. The purpose of a camera is to take points in the real world and project them onto a 2D plane, which we call an image. To do this, the information has to go through various processes and transformations. One of the most important transformations is parameterization, or what we call intrinsic and extrinsic parameters. The operation of a camera is a fundamental task in computer vision applications such as object reconstruction, object tracking, augmented reality and many others. Accurate calibration ensures accurate measurements and, therefore, reliable information, which is obtained by correcting distortions and estimating intrinsic and extrinsic parameters.

Camera calibration is the process of determining specific camera parameters in order to perform the necessary operations to undistort the captured image. It means that we have all the information about the camera, such as parameters or coefficients, that are needed to determine an accurate relationship between a 3D point in the real world and its corresponding 2D projection in the image captured by this calibrated camera. The two most important parameters to calculate are, as mentioned, intrinsic or internal parameters and extrinsic or external parameters:

Intrinsic parameters specific to each camera allow mapping between pixel coordinates and camera coordinates in the image frame; these parameters include focal length, optical center and skew coefficient. The focal length and optical centers can be used to create a camera matrix, which can be used to remove distortion caused by the lenses of a particular camera. The camera matrix is unique to a particular camera, so once calculated it can be reused for other images taken by the same camera. It is expressed as a 3x3 matrix, as shown in Fig. 17.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates | Intrinsic properties (Optical Centre, scaling) | Extrinsic properties (Camera Rotation and translation) | 3D World Coordinates

Fig. 17: Matrices describing both the intrinsic and extrinsic parameters of a camera.

Extrinsic parameters describe the orientation and position of the camera. This refers to the rotation and translation of the camera with respect to some world coordinate systems. The extrinsic parameters can be defined as a translation matrix **T** and a rotation matrix **R**, as shown in Fig. 17.

To fully calibrate the camera, we need to know all three matrices, one for the intrinsic parameters and two for the extrinsic parameters, so we can complete Fig. 17 equation and apply it to each image captured by the camera.

Next we will describe how the camera calibration process for the intrinsic parameters was carried out for this project. Since the camera was used with the ROS framework [59], specifically with a library called Open NI, which already has a pipeline for calibrating a camera, the methodology used by the OpenNI library [60] is basically the one implemented in OpenCV ([49], [61]), which can be explained as follows:

To calibrate the camera, a chessboard pattern with known dimensions, as shown in Fig. 18, must be printed to take at least 10 samples to compute the parameters. The OpenCV library [49] also contains the pattern with the known dimensions. The OpenCV function "findChessboardCorners" returns the corner points of the grid pattern; the detections can be seen in Fig. 19. Once the corner points are obtained, we can use the function "cornerSubPix" to increase the accuracy.

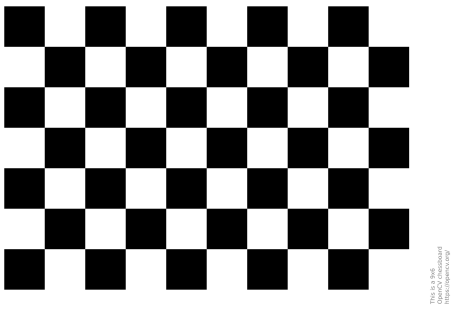

Fig. 18: Chessboard pattern used to calibrate the camera.

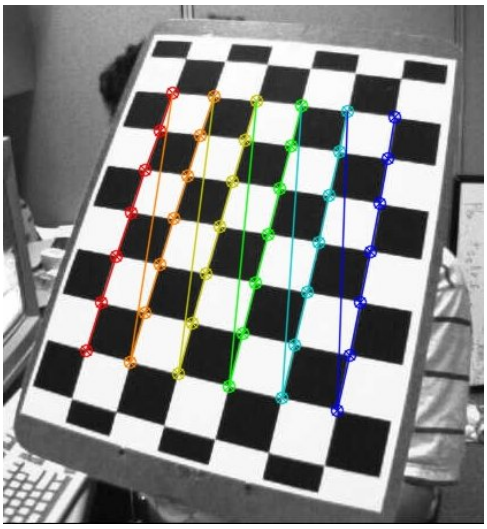

Fig. 19: Detected points in the chessboard pattern.

With the corner points obtained and refined from the ten samples, OpenCV provides a function to calibrate the intrinsic camera parameters called "calibrateCamera", which returns both intrinsic and extrinsic parameters of our camera.

Once we have both types of parameters, we can test them by undistorting an image using the function already implemented in OpenCV called "undistort". Since we are dealing with an RBGD camera, we need to know that the RGB and depth images are captured with different lenses and sensors; therefore, we need to calculate two different intrinsic parameters, one for each camera. The procedure for the depth camera is exactly the same, but to do this we need to occlude the IR emission lens, as can be seen in Fig. 20.

The following lines describe the robotic arm and camera used during the real life experiments in the laboratory.

The robotic arm used, as explained, is the Franka Panda Emika 21. This robotic arm is easy to interact with due to its intrinsic cooperative nature, which allows a human being to manually position it and interact with the gripper. More technical specifications can be seen in Table II.

The camera used, as said, is the Astra Orbbecc which has been designed to be highly compatible with existing OpenNI applications. The camera mounted on a tripod in the laboratory setup can be seen in Fig. 22. The Astra Orbbec 3D camera specifications can be found in Table III.

Fig. 20: RGB-D camera with IR blackout.

TABLE II: Key technical specifications of the Franka Panda Emika robotic arm used in the real life experiments.

Product Name	**Franka Panda Emika**
Axes	7
Payload	3.0 kg
Reach	850 mm
Repeatability	0.1mm
Weight	18 kg

Fig. 21: Robotic arm used in the laboratory.

TABLE III: Key technical specifications of the Astra Orbbec 3D. Camera used in the real life experiments.

Product Name	**Astra Orbbec 3D**
Range	0.6m - 8m
FOV	60°H x 49.5°V x 73°D
RGB Image Resolution	640 x 480 @ 30fps
Depth Image Resolution	640 x 480 @ 30fps
Size	165mm x 30mm x 40mm

Fig. 22: Camera setup used in the laboratory. The image contains the Astra Orbbec camera mounted on top of a tripod.

As described in section IV, the real experiments could not be performed due to the limited amount of time, even though everything was prepared. Therefore, some pictures of the setup were taken to be known in this document. The contents of Fig. 23 are images taken while trying to perform an experiment with a real object.

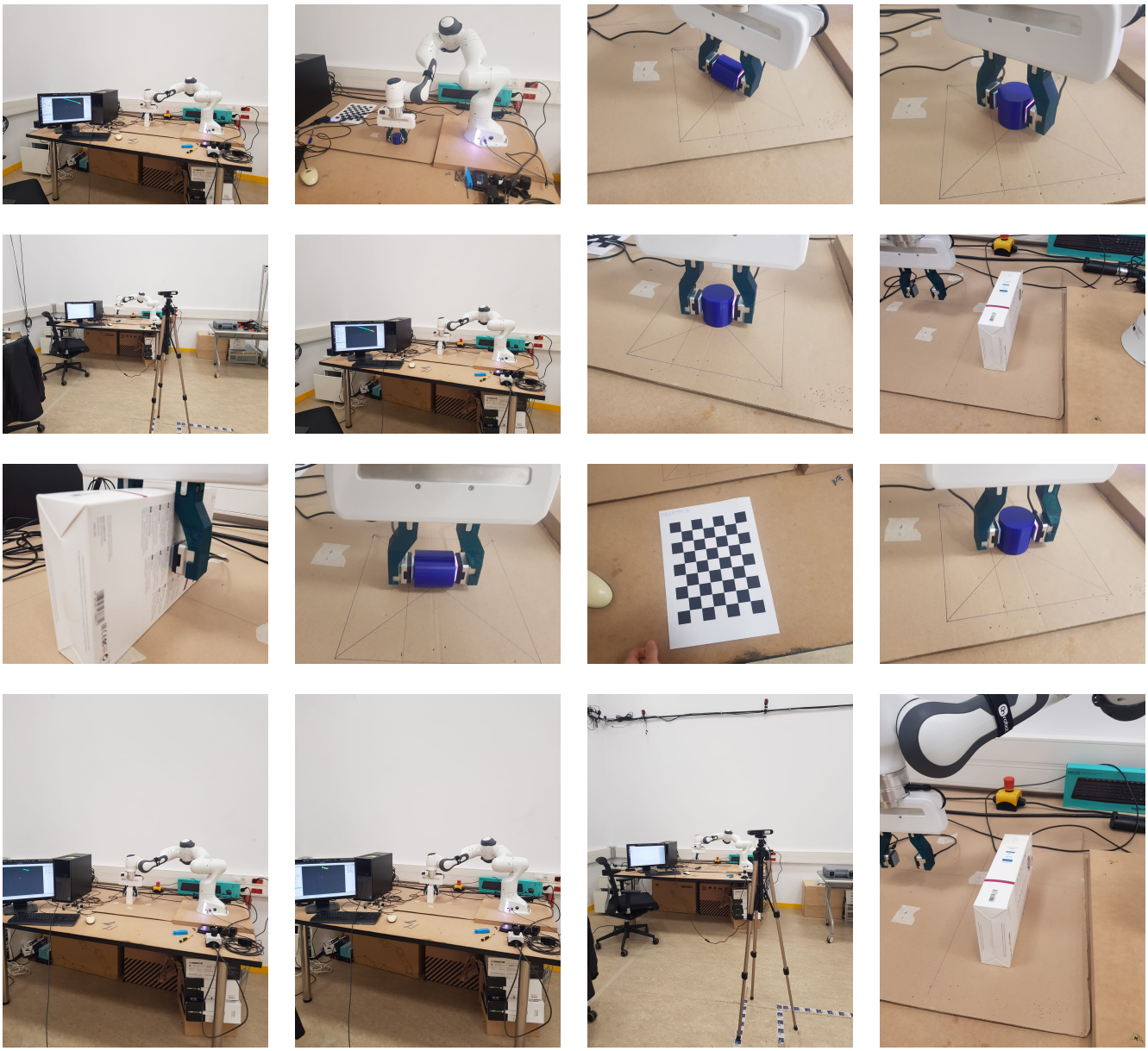

Fig. 23: Images taken of the real life setup in the laboratory while extracting real life data to perform experiments.