Vision-based detection and grasping by a mobile manipulator

José Ferreira da Silva
jose.ferreira.silva@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

February 1st 2021

Abstract

Robotics is poised to have a transformative impact in a variety of new markets and on various human social aspects. These include robot applications in disaster response, healthcare, domestic tasks, transport, space, manufacturing, and construction. However, there is a gap between current reality in robotic capabilities and the requirements of potential applications. One of the requirements is the use of vision to find and manipulate objects in the environment. The present work aims to shorten the gap in vision-based tasks by tackling the challenge of picking up brick shaped objects with a mobile manipulator endowed with vision. Previous attempts of performing similar tasks have resorted to single shot approaches which have proven successful in the past, in stationary applications such as the ones proposed by the Amazon Robotics Challenge. However, in mobile applications where the environment's state is constantly changing, the employment of a single-shot method would force the system to perform multiple stops for considerable amounts of time to record the changes in world in order to estimate correctly the pose of the desired object. Consequently, this leads to inefficient grasping strategies which require a great amount of time to complete. This thesis aims to develop a different approach, by employing an iterative multi-shot pose estimation method, constantly estimating the desired object's pose at a high frequency, implemented into a suitable grasping strategy, which is capable to perform the detection, approaching and grasping in a faster and more dynamically while maintaining the reliability and accuracy results of the previous single-shot approaches.
Keywords: Mobile Robots, Robotic Manipulators, Object Detection, Shape Fitting, Kalman Filter, Point Clouds, Pose Estimation

1. Introduction

This thesis aims to tackle the mentioned challenge, developing an accurate and time-efficient approach, through the combination of classical robotics methods, that is capable of locating and manipulating the different bricks through RGBD images using a 6DoF robotic arm mounted on top of a UGV, and evaluate its performance.

Even though the proposed thesis is inspired by challenge 2 of the MBZIRC competition and was tested in a similar environment, following the guidelines of the competition, the end goal of this work is not limited by the competition itself, but instead to serve as a base architecture for more complex problems that require object manipulation.

The original challenge requires a team of UAVs and an UGV to autonomously locate, pick and assemble a set of brick-shaped objects organized in three piles, each pile with a specific color and object dimensions. These objects are manipulated by the UGV to construct a pre-specified structure collaboratively with the UAVs. However, as neither the collaborative and brick placement aspects of the challenge are addressed in the current work, an adapted version will be addressed.

The following report will follow:

- **Literature Review**: A brief overview on previous attempts of tackling the grasping problem, as well as the used methods with the respective strengths an weaknesses.
- **Pose Computation**: Design and implementation of the multi-shot, closed-loop method to estimate the brick's pose.
- **Grasping strategy**: Design and implementation of an approaching and grasping strategy that exploits the proprieties of the proposed pose computation method.
- **Results Discussion**: Results obtained from

both pose computation method and grasping strategy through a simulation environment.

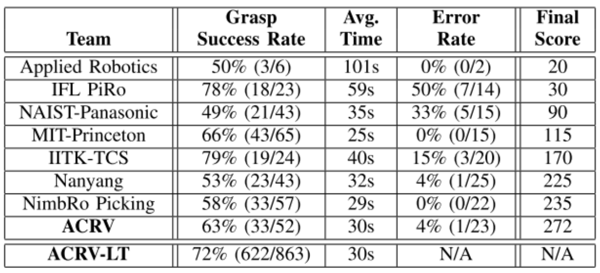
- **Conclusions**: Main achievements and limitations as well as future improvements.

2. Literature Review

In recent years, there has been increasing interest around tasks that require picking operation [1][2][3]. However, the gap between the capabilities of the previously available methods and the requirements of the real-world tasks that involve the picking motion is still very significant.

To shorten this gap, organizations try to fuel the interest in said tasks by inviting robotics groups to solve a set of challenges that include in some manner the action of picking up a certain object. Said challenges include the previously mentioned MBZIRC[4] which motivated the present thesis, but also the ARC[5]. The afford mentioned one is a fine case study for the task at hand since teams are required to pick-up unstructured objects in a some-what unstructured environment (warehouse shelves).

The common denominator between teams[6][7][8][9] is a single-shot architecture where firstly the desired object is detected through the camera's color information, followed by the computation of the grasping pose through the depth information. These are, in general, not robust to small changes in the environment (**open-loop**) and also rely on a single measurement (**single-shot**) which increases the chance of failure and require the system to be fully stationary to obtain a quality measurement leading to inefficient use of time. To put this into perspective, the winning team[6] of the 2016's edition managed a fail rate of 25% (9 out of 12 items picked) and in the 2017's edition, the winning team[9] managed a failure rate of 27% (33 out of 52 items picked). However, both teams only managed an average picking up time of around 30s (Figure 1).



Team	Grasp Success Rate	Avg. Time	Error Rate	Final Score
Applied Robotics	50% (3/6)	101s	0% (0/2)	20
IFL PiRo	78% (18/23)	59s	50% (7/14)	30
NAIST-Panasonic	49% (21/43)	35s	33% (5/15)	90
MIT-Princeton	66% (43/65)	25s	0% (0/15)	115
IITK-TCS	79% (19/24)	40s	15% (3/20)	170
Nanyang	53% (23/43)	32s	4% (1/25)	225
NimbRo Picking	58% (33/57)	29s	0% (0/22)	235
ACRV	63% (33/52)	30s	4% (1/23)	272
ACRV-LT	72% (622/863)	30s	N/A	N/A

Figure 1: Results from the ARC 2017, from [9].

The present thesis is inspired by the approaches seen in the ARC, but will aim to improve the time efficiency of the afford-mentioned approaches, by implementing a similar architecture (object detection followed by grasping/pose estimation) ran multiple times over time (**multi-shot**) resorting to previous estimations to improve both accuracy and efficiency (**closed-loop**).

2.1. Object Detection

Object detection consists in detecting and locating instances of objects within an image. A single object may be desired to detect but in general multiple classes of objects are looked for in the image. These started getting serious attention at the beginning of the millennium with a series of machine learning approaches which firstly would extract features from images and then train a classifier that would classify the extracted features and point where the object would be in the image[10][11][12]. These showed good results in terms of classification accuracy but would lack in efficiency which hindered real-time applications.

More recent approaches make use of deep learning techniques, namely CNNs to perform detection. This kind of approaches can be divided into two groups, Multi-shot detectors[13][14] and single Single-shot detectors[15][16]. The first ones are known to be very accurate but efficiency remained an issue. Single-shot detectors, on the other hand, are much more efficient while sacrificing accuracy. Improvements in both approaches have leveled the playing field but single-shot approaches are currently state of art when it comes to object detection methods, wielding accuracy results of over 90%, same as the multi-shot techniques, while being able to perform over 40 detections per second.

Deep learning techniques, however, have a major disadvantage which is the critical need for a suitable data set with a large number of examples. This means a great deal amount of work is required gathering and labeling images which rules out the usage in a large number of problems such as this one.

However, in other traditional robotic applications [17][18][19], the use of color segmentation methods proved to be a reliable and fast approach to object detection, especially while working with objects with simple features such as the building bricks.

2.2. Grasping Computation

Grasping computation involves computing the optimal point of contact and angle of attack to approach an object to grasp it, resorting to Depth information and Color information in some cases. This is a problem that has been around for a long time and has attracted a lot of attention over the last couple of decades. There are three main categories where grasping computation techniques may fall, methods that require known 3D CAD Models[20][21][22][23], methods that do not require 3D CAD Models[24][25] and finally Deep Learning methods [26][27]. The latter two approaches try to find in the object's point cloud the best grasping point and its orientation. The non CAD methods can be interpreted as classical machine learning methods, where features such as the surface normals are extracted and classified, whereas Deep Learning methods, as the name suggests, resort to Deep Neural Networks (CNNs) to perform the grasping computation similarly to the Object Detection methods resorting to CNNs.

These two kinds of approaches are currently state of the art in terms of accuracy, achieving accuracy results of over 90%, but the non CAD methods often lack in efficiency whereas recent Deep Learning techniques can perform over 10 computations per second. Once again the disadvantage of said methods is the need for a suitable training set (especially in Deep learning techniques).

The methods that require a CAD Model, aren't as much a solution to the grasping computation problem, but instead for the point set registration problem, where two sets of points differ from a rigid transformation which is desired to estimate. These methods are very suitable when dealing with simple objects, such as the bricks, and therefore will be used in this thesis.

A rather popular point set registration algorithm is the ICP[28][29], which has been around for over 20 years and has been used in countless robotics applications [30][31][32][33][34][35] to name a few.

3. Pose Computation

To be able to detect the brick, even with a moving camera, the previous single-shot, open-loop approaches need to be adjusted to be capable of iteratively detecting and tracking the moving brick. This calls for the need of a multi-shot, closed-loop detection method.

The proposed approach still requires the previously described pipeline of Object Detection followed by the Grasping Computation. However, the output from this pipeline is reused in the next iteration to improve the estimate and execution times.

3.1. Brick Detection

For this thesis, based on MBZIRC 2020 challenge 2 guidelines, the only red object in the environment will be in fact the brick. Therefore the brick can be detected by color thresholding the corresponding HSV image and grouping the extracted pixels together. One can assume that the largest object in the color segmentation output (i.e the one with the most pixels) will correspond to the brick.

The camera's color information comes already aligned with the point cloud, so, to retrieve the coordinates of the points belonging to the brick's surface it is only necessary to extract the 3D coordinates of the largest object's pixels, in the output of the color segmentation.

The previously described process to extract the K points in the brick's surface S_C $[3 \times K]$, is described by the block diagram in Figure 2.

3.2. Grasping Computation

According to Section 2, the chosen approach for the Grasping Computation module is a Shape Fitting method, namely the Iterative Closest Point. This approach requires a 3D Model of the brick to be generated.

The model generation will consist of firstly a selection of which faces to include in the 3D model (**face selecting**) and then determine the sampling interval between points in each face (**face sampling**). The first one is achieved by a simple threshold method whilst the second makes use of a heuristic function that predicts the spacing between points in the 3D Model based on the brick's pose. This procedure, to the extent of the author's knowledge, was designed in the present work.

Assuming a 2D version of the problem taking only into consideration the coordinates among the x and z axis, the number of points in each face depends on the brick's rotation over the y axis, β, and on the brick's position in the XZ plane, as shown in Figure 3.

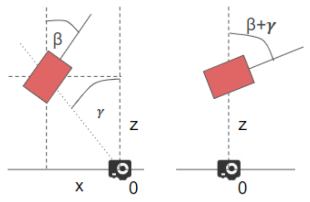


Figure 3: 2D Version of the problem to understand the dependency of the number of points with the brick's pose. Both cases appear similar through the camera's perspective.

Both face selecting and sampling will be a function of $\beta + \gamma$. If the brick's faces are numbered according to Figure 4,

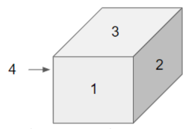


Figure 4: Numeration of brick's faces (irrelevant to enumerate further due to the brick's symmetry)

then, assuming that the brick is always below the

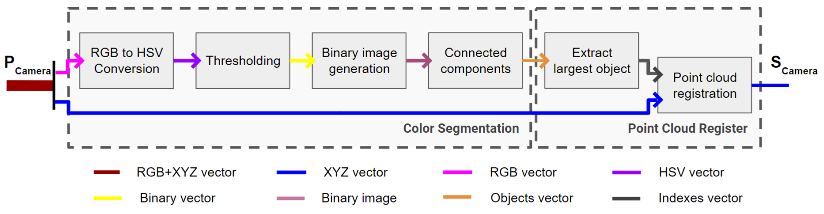


Figure 2: Brick detection pipeline.

camera's horizontal plane, **face selection** will follow

$$Mb_{Brick} = \begin{cases} \{Face1\ ;\ Face2;\ Face3\},\ if\ \beta+\gamma > 0 \\ \{Face1\ ;\ Face3;\ Face4\},\ if\ \beta+\gamma < 0 \end{cases} . \tag{1}$$

Regarding **face sampling**, there are a couple of options that could be followed but the simplest of them is to design an heuristic function that states the sampling interval in the side face and in the forward face as a function of $\beta+\gamma$. For the top face, a constant sampling interval is considered.

In Figure 5 are represented the functions that model the sampling interval as a function of the brick's pose corresponding to

$$F1(\beta+\gamma) = \left(3 - \frac{4|\beta+\gamma|}{\pi}\right)^{-1}$$

for face 1, and

$$F24(\beta+\gamma) = \left(1 + \frac{4|\beta+\gamma|}{\pi}\right)^{-1}$$

for faces 2 and 4.

These functions were dimensioned to model the observed variation in number of points of the captured point cloud with $\beta+\gamma$. This behaviour is defined by maximizing the number of points in Face number 1 whilst minimizing the number of points in Face 2/4 when the brick is presented facing forward ($\beta+\gamma = 0$). Vice versa, when the brick is presented sideways ($\beta+\gamma = \pm\frac{\pi}{2}$), with a similar sampling interval between faces when the brick is displayed diagonally ($\beta+\gamma = \pm\frac{\pi}{4}$)

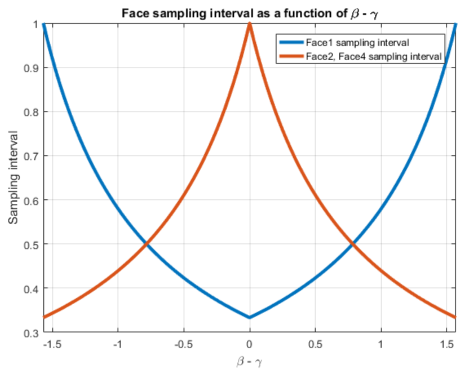


Figure 5: Sampling interval modifier functions for faces 1, 2 and 4.

The final sampling intervals of each face is a transformation of a base value st and the functions previously described, leading to

$$Sampling\ interval: \begin{cases} Face\ 1 = \frac{st}{(3-\frac{4|\beta+\gamma|}{\pi})} \\ Faces\ 2,4 = \frac{st}{(1+\frac{4|\beta+\gamma|}{\pi})} \\ Face\ 3 = st \end{cases} . \tag{2}$$

The generation of the 3D model can be described by the block diagram of Figure 6.

3.3. Closing the Loop

As mentioned, a Kalman Filter will be used to track the brick's movement and consequently enhance performance. There is no available information regarding on how the brick's state evolves through time and therefore state transitions will follow a first-order state model given by

$$\begin{cases} x_k = x_{k-1} + v_{k-1} \\ v_k = v_{v_k-1} + w \end{cases} , \tag{3}$$

where w is the white noise associated with the state transition, x_k is the system's state in time instant k and v_k corresponds to the system's velocity in time instant k ($\frac{dx_k}{k} = v_k$). Therefore, the state variables are the brick's linear and angular position

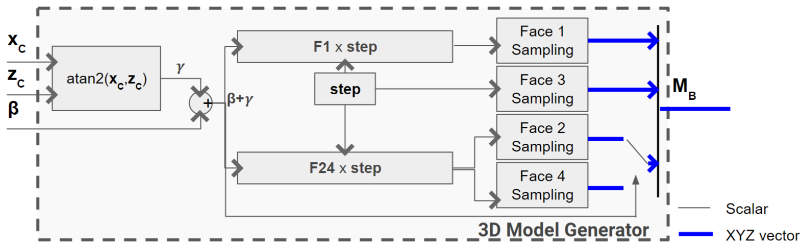


Figure 6: 3D Model generation.

as well as it's linear and angular velocities, in relation to the camera's frame, represented in the state vector x as

$$x = [X_{Camera}\ Y_{Camera}\ Z_{Camera}\ v_x\ v_y\ v_z\ \ldots$$
$$\ldots\ \alpha\ \beta\ \theta\ \omega_\alpha\ \omega_\beta\ \omega_\theta]^T. \quad (4)$$

With this, the filter's prediction step matrices are

$$A = \begin{bmatrix} 1&0&0&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&0&0&1&0&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix},$$

$$Q = diag([0\ 0\ 0\ \mathbf{0.05\ 0.05\ 0.05}\ 0\ 0\ 0\ \mathbf{0.01\ 0.1\ 0.01}]). \quad (5)$$

The update step usually makes use of measurements from the system's sensors to correct the prediction, which in this case corresponds to the Depth Camera. However, the Depth Camera does not output an explicit observation of the system's variables. On the other hand, the pipeline of Object Detection followed by the Grasping Computation, outputs an estimate of the brick's position and orientation. In light of this, the update step is described by the following matrices

$$H = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0 \end{bmatrix},$$

$$R = diag([0.01\ 0.01\ 0.01\ \frac{\pi}{18}\ \frac{\pi}{18}\ \frac{\pi}{18}]). \quad (6)$$

Values for matrices Q and R were initially chosen arbitrarily and then fined tuned to produce the best results in tests similar to the ones showcased in the Results section. The covariance matrix is initialized as P_0 through a similar procedure resulting in

$$P_0 = diag([\mathbf{0.1\ 0.1\ 0.1}\ 0\ 0\ 0\ \mathbf{0.3\ 0.1\ 0.3}\ 0\ 0\ 0]).$$

The values from both matrices indicate a similar contribution from both prediction and update steps when it comes to estimating the position of the system. However, regarding the orientation estimation, the filter relies much more on the prediction step since the ICP's orientation output is quite susceptible to deviations whereas the position output is very reliable.

An additional matching step will be performed to discard outlier observations, if the prediction of a state variable varies more than $\frac{\pi}{4}$ over the respective observed value, the faulty measurement is flagged and discarded.

The iterative module can be described by the block diagram of Figure 7.

3.4. Initialization

As every iterative process, it must have a suitable initialization state. To provide said initial state, an Initialization Module is required to produce a rough estimate of the brick's pose.

The process is split into two parts, position estimation and orientation estimation. For positioning, the mean value of the segmented point cloud,S_{Camera} (i.e points that belong to the brick's surface) is used, as

$$[X_0, Y_0, Z_0] = mean(S_{camera}). \quad (7)$$

For the orientation, it is assumed that the brick is placed on a horizontal plane, parallel to the floor, which means that it is only required to initialize the rotation angle around the Y-axis. To do so, firstly

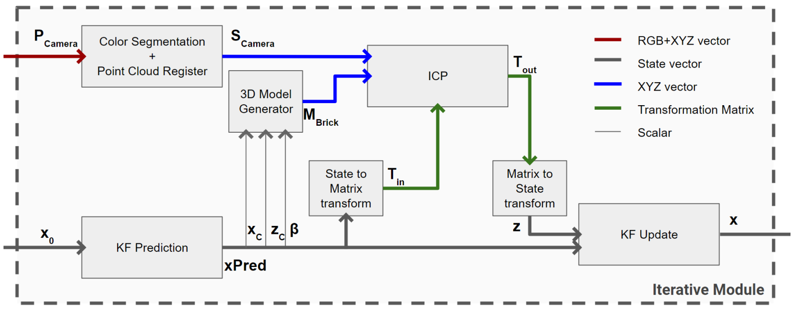


Figure 7: Block diagram of the iterative module.

the points in the segmented point cloud are projected onto the horizontal plane XZ which will generate a 2D point cloud given by

$$XZ_{Camera} = [S_{Camera}]_{1\ 2,:}. \tag{8}$$

Then, SVD is applied to the covariance matrix of the said point cloud which, will output two rotation matrices (U and V^*) and a scaling matrix (Σ).

$$U, \Sigma, V^* = SVD\,(\,cov(XZ_{Camera})\,)\,. \tag{9}$$

Matrix V^* consists in the concatenation of two vectors, $v_1 = [X_1, Z_1]^T$ and $v_2 = [X_2, Z_2]^T$, which represent the two main directions of the decomposed set of points as shown in Figure 8.

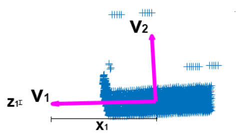


Figure 8: Visual representation of the principal directions from the output of SVD.

The rotation angle can be computed from the components of v_1 (most energy of the two), as

$$\beta = atan2(Z_1, X_1). \tag{10}$$

The rotation angle around the x axis is assumed to be null and the rotation angle around z can be assumed to be close to $-\frac{\pi}{4}$ which is sufficient to account for the Camera's pitch and also to not trigger a faulty measurement on the iterative module.

Finally, the state initialization x_0 is given by

$$x_0 = \left[X_0, Y_0, Z_0, 0, \beta, -\frac{\pi}{4}\right]^T. \tag{11}$$

The initialization module is called before the first iteration of the iterative module, and every time the matching step of the Kalman filter in the iterative module flags that an outlier observation has been made. This is to prevent cases where, discarding the observation, disrupts the tracking of the brick's movement.

Said process is described in the block diagram of Figure 9.

4. Grasping Strategy

It is now desired to not only locate the brick in the world frame but also to develop a strategy to pick-up said brick efficiently resorting to the previously described method and the ATRV and robotic arm.

Ultimately, efficiency means approaching the brick in the least amount of time possible. This requires executing as many tasks in parallel as possible and avoiding states where both the ATRV and the arm remain stationary

To accomplish this, several referentials are introduced, namely the $World$ frame and the $ATRV$ frame. Other auxiliary frames are considered such as the robotic arm's joints (Figure 10). The transformations between all frames are known.

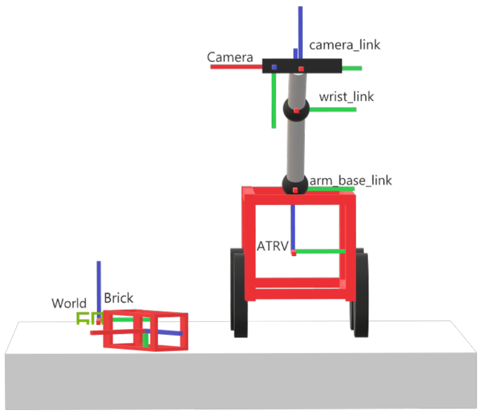


Figure 10: Representation of the problem's several frames.

To approach the brick, the following strategy will be employed, following four sequential stages or states

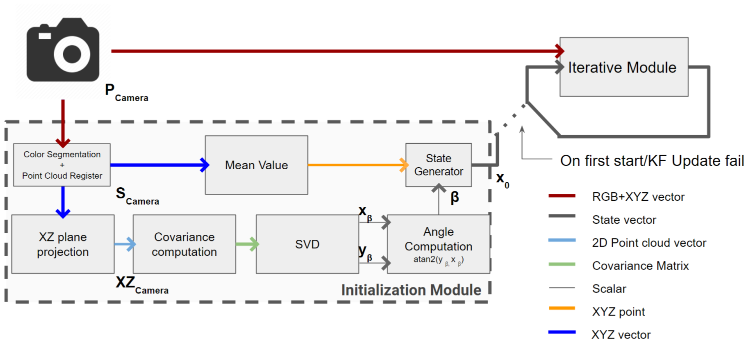


Figure 9: Block diagram of the Initialization Module + Iterative Model.

1. Start moving the UGV towards the previously known location of the brick while looking for it,
2. When found, compute an initial estimate of the brick's pose and update the ATRV's movement goal, improving the estimate of its pose while moving towards it,
3. When close, stop the ATRV movement for a short amount of time for the estimate to converge,
4. When finished, stop the estimation and move the ATRV closer to the brick while the arm stretches to reach the brick.

Each of the referred states are defined by the movement the ATRV is describing and the action that the arm+camera are performing (either observing or acting on the environment). With this, the execution and transition of each state can be modeled by the Petri Net of Figure 11.

The proposed method's nodes in Figure 11 are nodes that represent the behaviour of the proposed detection algorithm.

5. Results & discussion

The following section will evaluate the accuracy and efficiency of the proposed detection method and of the proposed grasping strategy.

Access to the real hardware was not possible and therefore, for the detection method's tests, images were generated on a ROS Kinetic[36]+Gazebo[37] environment, resorting to the OpenNNi Kinect plugin and processed in MATLAB[38] R2016a offline frame by frame, resorting to the Computer Vision Toolbox[39] for the ICP method with the Point to Point metric. Image Processing Toolbox[40] was also used for the Connected Components algorithm.

For the grasping strategy results, simulation environments provided by the IST's MBZIRC 2020 team[41] were used, which also include the path planning and execution for both ATRV and the robotic arm. To use the proposed method on a real-time execution, a python implementation was made using OpenCV[42] for color segmentation and the SVD method, Python-PCL[43] for the point cloud register method (ICP) with a point to point metric and finally Numpy[44] for operations involving matrices.

5.1. Detection Method

For the following results, efficiency is measured in FPS, the number of estimates per second which can be obtained by the inverse of the time an estimation took to compute. Performance will be measured through the RMS error from the ICP output given by

$$RMS = \sqrt{\sum_{i=1}^{N_P} \frac{(s_i - t_i)^2}{N_P}}, \qquad (12)$$

where $s_i \in [S'_C]$ which corresponds to the points in S_C closest to Mb_B, $t_i \in [T_B^C \times Mb_B]$ corresponding to points in the transformed point cloud and N_P corresponds to the total number of points.

The brick is placed on the ground plane roughly $1m$ away from the camera which is stting at an height of $70cm$. 25 different orientations ranging from -90 to 90 are tested, resulting in the RMS and FPS values of Figure 12.

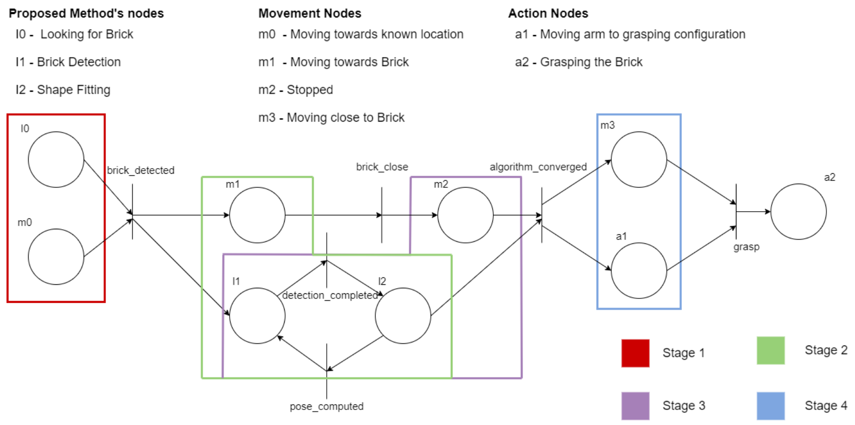


Figure 11: PETRI net modeling the approaching and pre-grasp behaviour.

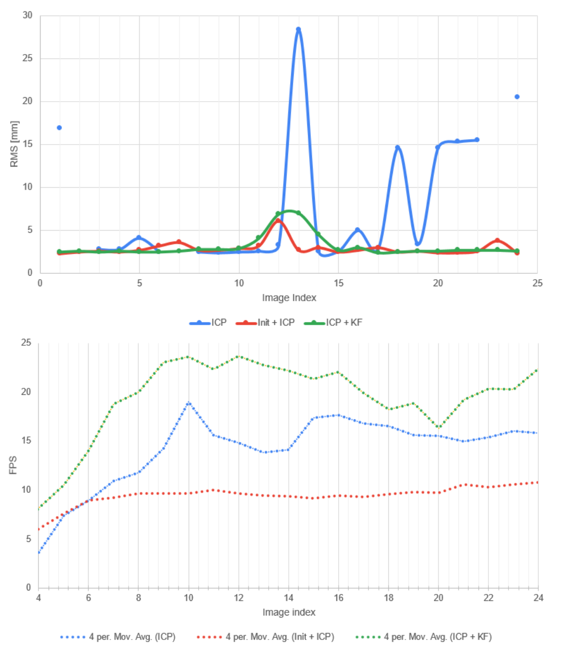


Figure 12: Performance comparison between the single-shot method(ICP), single-shot method with initialization (Init+ICP) and the multi-shot method (KF+ICP).

The proposed detection method is abble to correctly estimate the brick's pose in all attempted orientations, with an average RMS value of $2.8mm$. Additionally there's a clear increase in efficiency by employing the multi-shot iterative method, evidenced by the higher FPS numbers using the Kalman Filter to close the loop.

Additionally tests were performed placing the brick $2m$ away from the camera. In these the camera is approaching the brick at a constant speed and led to the results in Figures 13 and 14.

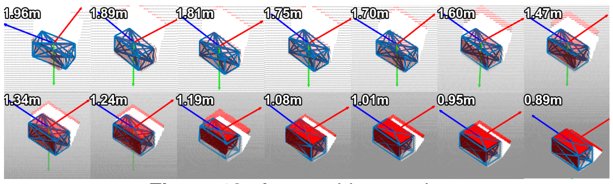


Figure 13: Approaching results.

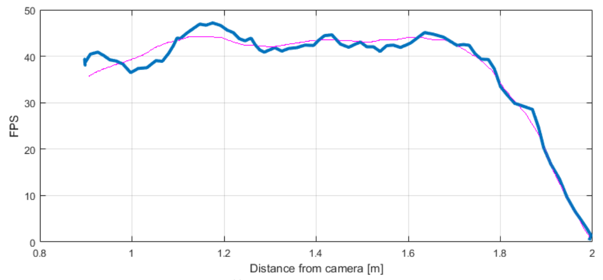


Figure 14: Method's efficiency while approaching the brick. Blue line corresponds to the actual values filtered by averaging 3 measurements and the purple line corresponds to the observed trend.

It is clear from the results that even with the brick far away (fewer points in the point cloud) the detection method is still able to correctly estimate it's pose and also track the brick's movement as it gets closer.

It is also noticeable an improvement of the FPS values in this testing scenario, in the stationary test an average of around 20 detections per second were registered, whereas in the approaching scenario, an average of 40 are recorded. This is because in this scenario, the brick is moving linearly at a slow rate instead of rotating, meaning that there's less movement between two consecutive iterations which leads to the ICP converging in less iterations.

5.2. Grasping strategy

A total of 10 attempts to pick up the brick starting roughly $2m$ away from it (Figure 15) were performed. In these, the brick was placed on the ground plane and an estimate of it's location is known beforehand.

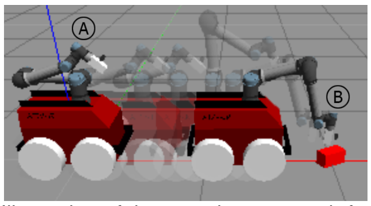


Figure 15: Illustration of the grasping approach from the starting position (A) to the Grasping position (B).

The trials followed the evolution in Figure 16 and the final positions of the arm's end-efector in relation to the brick are in Figure 17.

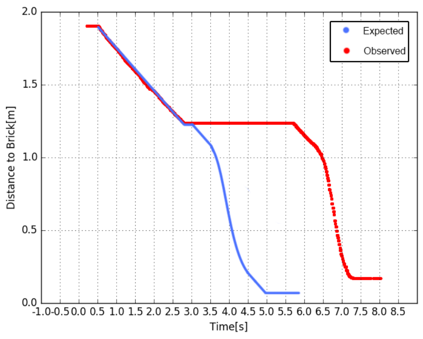


Figure 16: Approaching distance evolution over time in comparison with the expected outcome.

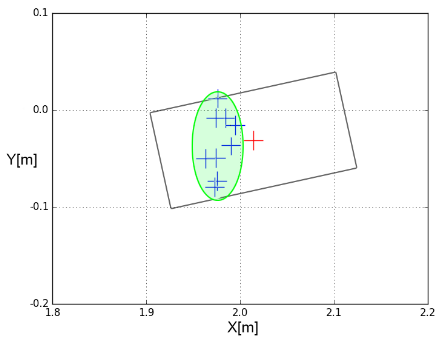


Figure 17: Final end-effector position in 10 Trials (Blue '+'s), compared to brick's center (Red '+').

Evidenced in Figure 16, the approaching took 3 seconds longer than expected. This is due to the fact that the used hardware is not able to handle the simulations and the point cloud processing and led to performance issues. Other than that, the approaching described a fluid and continuous evolution through time as desired.

As for accuracy results, 9 out of the 10 trials saw the end efector on top of the brick surface. However, it is noticeable a large deviation in it's final position, which is due to the pre-configured arm configuration to grasp the brick which is not abble to cope with small deviations in the ATRVs path.

6. Conclusions

Results showed that the proposed estimation method was capable of reliably producing an accurate estimate of the brick's pose while approaching it at a rate of 40 detections per second. More complex objects however would require changes in the detection method, addressed in fuure work. Further results have also shown that Partial occlusion of the brick's faces is acceptable however, if one of them becomes invisible to the camera, the method is not able to compute a valid estimate of the object's pose.

Through the proposed grasping strategy, the system takes 8 seconds to pick a brick placed around 1.7m away from the ATRV, with a failure rate of 10%. Once again, the proposed strategy presents some limitations which are once again addressed in future work.

ARC teams reached failure rates of 25%. Yet, objects and conditions were much more demanding. With this in mind, it is hard to compare the two approaches since the results came from two very different scenarios. Nevertheless, the fastest team managed to pick up the first object in 30 seconds, whilst the proposed strategy managed to approach and pick up the brick in under 8 seconds. Even with the referred performance issues that required the system to stop 12 times more time (3s) than necessary (0.25s). Therefore, it is possible to conclude that employing a multi-shot method led to a much more time-efficient grasping strategy.

6.1. Future Work

Future work s should focus on three main topics, the adaptation of the proposed method of pose estimation to multiple and more complex objects, general improvements to the grasping strategy and finally further testing to determine the reliability of the proposed grasping strategy in different scenarios, especially outdoor ones.

Regarding the firstly mentioned topic, this could be achieved by employing an adaptation of the Deep Learning object detection methods referred in Section 2. These compute the mask that contains the object[45][46], similarly to this study. By employing the Deep Learning method, multiple objects can be detected which would require an object tracking method [47] to be used. Regarding shape matching, the method of heuristically generating the expected point cloud through the previous state can be maintained, with the object's 3D

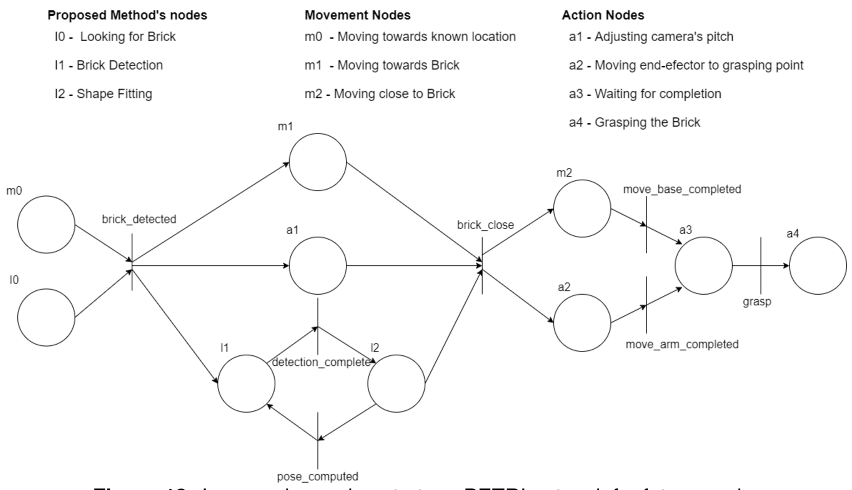


Figure 18: Improved grasping strategy PETRI network for future work.

model consisting of a combination of point clouds from different points of view.

As for the grasping strategy, the final arm configuration should be configured autonomously in run time, instead of pre-computed and fixed. Other than this, the camera should move as the ATRV approaches the brick, to keep it in the center of its field of view. This can also allow bigger bricks to be used since it can maintain all three faces in sight. The need for the ATRV to stop for a final estimation can be dropped, if the hardware is capable of running the proposed method at a moderate frame rate (20 FPS). The referred changes are summarized in the PETRI diagram of Figure 18.

References

[1] Valdrin Krasniqi et al. “Control algorithm of a pick and place three dimensional robots”. In: *IFAC Proceedings Volumes (IFAC-PapersOnline)*. Vol. 15. PART 1. IFAC Secretariat, 2013, pp. 440–443. ISBN: 9783902823427. DOI: 10.3182/20130825-4-US-2038.00029.

[2] Vahid Aminzadeh et al. “A new algorithm for pick-and-place operation”. In: *Industrial Robot* 37.6 (2010), pp. 527–531. ISSN: 0143991X. DOI: 10.1108/01439911011081678.

[3] Rahul Kumar et al. “Object Detection and Recognition for a Pick and Place Robot Tongue Drive System View project DIY PPE Design for Developing Countries Fighting COVID-19 View project Object Detection and Recognition for a Pick and Place Robot”. In: (). DOI: 10.13140/2.1.4379.2165. URL: https://www.researchgate.net/publication/271769133.

[4] *MBZIRC 2020*. http://mbzirc.com/challenge/2020. URL: http://mbzirc.com/challenge/2020 (visited on 01/06/2020).

[5] Amazon. *Amazon Picking Challenge - The Future of Robotics*. http://amazonpickingchallenge.org/. URL: http://amazonpickingchallenge.org/ (visited on 01/03/2020).

[6] Carlos Hernandez et al. *Team Delft's Robot Winner of the Amazon Picking Challenge 2016*. Tech. rep. arXiv: 1610.05514v1.

[7] Swagat Kumar et al. *Design and Development of an automated Robotic Pick & Stow System for an e-Commerce Warehouse*. Tech. rep. 2017. arXiv: 1703.02340v1. URL: http://wiki.ros.org/ROS/Tutorials.

[8] Nikolaus Correll et al. “Analysis and Observations from the First Amazon Picking Challenge”. In: *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING* (), p. 10. DOI: 10.1109/TASE.2016.2600527. arXiv: 1601.05484v3.

[9] D Morrison et al. *Cartman: The low-cost Cartesian Manipulator that won the Amazon Robotics Challenge*. Tech. rep. arXiv: 1709.06283v2.

[10] Paul Viola and Michael Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. Tech. rep. 2001.

[11] David G Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. Tech. rep. 2004.

[12] Navneet Dalal and Bill Triggs. *Histograms of Oriented Gradients for Human Detection*. Tech. rep. URL: http://lear.inrialpes.fr.

[13] Ross Girshick et al. *Region-based Convolutional Networks for Accurate Object Detection and Segmentation*. Tech. rep. URL: http://www.cs.berkeley.edu/rbg/rcnn..

[14] Ross Girshick. *Fast R-CNN*. Tech. rep. arXiv: 1504.08083v2. URL: https://github.com/rbgirshick/.

[15] Wei Liu et al. *SSD: Single Shot MultiBox Detector*. Tech. rep. arXiv: 1512.02325v5. URL: https://github.com/weiliu89/caffe/tree/ssd.

[16] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. Tech. rep. URL: https://pjreddie.com/yolo/..

[17] Pedro Caleiro, António Neves, and Armando Pinho. "Color-spaces and color segmentation for real-time object recognition in robotic applications". In: *Revista Do DETUA* 4 (Jan. 2007).

[18] Morten Rufus Blas et al. "Fast color/texture segmentation for outdoor robots". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2008), pp. 4078–4085. DOI: 10.1109/IROS.2008.4651086.

[19] Zbigniew Wasik and Alessandro Saffiotti. "Robust color segmentation for the RoboCup domain". In: *Proceedings - International Conference on Pattern Recognition*. Vol. 16. 2. 2002, pp. 651–654. DOI: 10.1109/icpr.2002.1048386.

[20] Kok-Lim Low. *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. Tech. rep.

[21] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. ISSN: 01628828. DOI: 10.1109/34.121791.

[22] Niloy J Mitra and An Nguyen. *Estimating Surface Normals in Noisy Point Cloud Data*. Tech. rep. 2003.

[23] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. *Efficient RANSAC for Point-Cloud Shape Detection*. Tech. rep. 1981, pp. 1–12.

[24] Marcus Gualtieri et al. "High precision grasp pose detection in dense clutter". In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2016-Novem. Institute of Electrical and Electronics Engineers Inc., 2016, pp. 598–605. ISBN: 9781509037629. DOI: 10.1109/IROS.2016.7759114. arXiv: 1603.01564.

[25] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. *Robotic Grasping of Novel Objects using Vision*. Tech. rep.

[26] Ian Lenz, Honglak Lee, and Ashutosh Saxena. *Deep Learning for Detecting Robotic Grasps*. Tech. rep.

[27] Joseph Redmon and Anelia Angelova. *Real-Time Grasp Detection Using Convolutional Neural Networks*. Tech. rep.

[28] Wikipedia. *Iterative Closest Point*. 2020. URL: https://en.wikipedia.org/wiki/Iterative{_}closest{_}point.

[29] Paul J. Besl and Neil D. McKay. *A Method for registration of 3D Shapes*. URL: http://www-evasion.inrialpes.fr/people/Franck.Hetroy/Teaching/ProjetsImage/2007/Bib/besl{_}mckay-pami1992.pdf (visited on 07/29/2020).

[30] Jaromir Konecny, Michal Prauzek, and Jakub Hlavica. "ICP Algorithm in Mobile Robot Navigation: Analysis of Computational Demands in Embedded Solutions". In: *IFAC-PapersOnLine* 49.25 (2016), pp. 396–400. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.12.079.

[31] Barbara Siemiątkowska and Arkadiusz Zychewicz. "The application of ICP and SIFT algorithms for mobile robot localization". In: *CISM International Centre for Mechanical Sciences, Courses and Lectures*. Vol. 524. Springer International Publishing, 2010, pp. 391–398. DOI: 10.1007/978-3-7091-0277-0_46.

[32] Jorge L. Martínez et al. "Mobile robot motion estimation by 2D scan matching with genetic and iterative closest point algorithms". In: *Journal of Field Robotics* 23.1 (2006), pp. 21–34. ISSN: 1556-4959. DOI: 10.1002/rob.20104. URL: http://doi.wiley.com/10.1002/rob.20104.

[33] Héber Sobreira et al. "Map-Matching Algorithms for Robot Self-Localization: A Comparison Between Perfect Match, Iterative Closest Point and Normal Distributions Transform". In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 93.3-4 (2019), pp. 533–546. ISSN: 15730409. DOI: 10.1007/s10846-017-0765-5. URL: https://link.springer.com/article/10.1007/s10846-017-0765-5.

[34] Jack Woods et al. “The application of iterative closest point (ICP) registration to improve 3D terrain mapping estimates using the flash 3D ladar system”. In: *Laser Radar Technology and Applications XV*. Ed. by Monte D. Turner and Gary W. Kamerman. Vol. 7684. SPIE, 2010, 76840N. DOI: 10 . 1117 / 12 . 849724. URL: http : / / proceedings . spiedigitallibrary . org / proceeding . aspx ? doi = 10 . 1117 / 12 . 849724.

[35] Tianyuan Gu and Ning Zhang. “Application of iterative closest point algorithm in automatic flight of speedy UAV”. In: *2014 IEEE Chinese Guidance, Navigation and Control Conference, CGNCC 2014*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1456–1459. ISBN: 9781479946990. DOI: 10.1109/CGNCC.2014.7007407.

[36] *kinetic - ROS Wiki*. URL: http://wiki.ros.org/kinetic (visited on 09/08/2020).

[37] *Gazebo: ROS overview*. URL: http : / / gazebosim . org / tutorials ? tut = ros{_}overview (visited on 09/08/2020).

[38] *MATLAB - MathWorks - MATLAB & Simulink*. URL: https : / / www . mathworks . com / products / matlab . html (visited on 09/08/2020).

[39] *Computer Vision Toolbox - MATLAB & Simulink*. URL: https : / / www . mathworks . com/products/computer-vision.html (visited on 09/08/2020).

[40] *Image Processing Toolbox - MATLAB*. URL: https : / / www . mathworks . com / products / image.html (visited on 09/08/2020).

[41] M. Basiri et al. *An autonomous mobile manipulator to build outdoor structures consisting of heterogeneous brick patterns*. Tech. rep. 2020.

[42] *OpenCV*. URL: https://docs.opencv.org/2.4/index.html (visited on 08/29/2020).

[43] *Python-PCL*. URL: https : / / github . com / strawlab / python - pcl (visited on 08/29/2020).

[44] *NumPy*. URL: https://numpy.org/ (visited on 08/29/2020).

[45] Jianing-sun. *Mask-YOLO: A Multi-task Learning Architecture for Object Detection and Instance Segmentation*. URL: https : / / github . com / jianing - sun / Mask - YOLO (visited on 08/31/2020).

[46] Kaiming He et al. *Mask R-CNN*. Tech. rep. arXiv: 1703 . 06870v3. URL: https : / / github.com/.

[47] A Acm Reference Format: Yilmaz, O Javed, and M Shah. “Object tracking: A survey”. In: *ACM Comput. Surv* 38 (2006), p. 45. DOI: 10 . 1145 / 1177352 . 1177355. URL: http : / / doi.acm.org/10.1145/1177352.1177355.