QR code-based self-calibration for a fault-tolerant industrial robot arm

Yizheng Zhang, Wangshu Zhu and Andre Rosendo
Living Machine Lab
ShanghaiTech University
Shanghai, China
Email: {zhangyzh1,zhuwsh,arosendo}@shanghaitech.com

***Abstract*—Malfunctions on industrial robots can cost factories 22,000 dollars per minute. Although the benefits of a fault-tolerant robot arm are clear, redundant sensors would steeply add to the costs of such robots, while machine learning-based methods spend too much time learning the robot's model. In this work, we propose a simple method to infer which joint underwent failure and at which angle this joint is constrained, to then modify the inverse kinematics (IK) algorithm to adaptively achieve the goal. Our method involves combining a QR code with an inexpensive camera, building a virtual link between these two to give the relative position of the end effector. Once one joint/encoder/motor suffers damage we use this virtual link to calibrate this joint by coordinate transformation, to calculate the constrained angle, and to recalculate the trajectory through IK iterations with the Newton-Raphson method. We prove the efficacy of our method with pick-and-place experiments, commonly seen in industrial settings, emulating malfunctions on different joints and at different angles, and our method can successfully finish the task in most cases. We further demonstrate that our method is capable, for almost all of the six degree-of-freedom manipulators, to adapt to joint failures after suffering a position failure. With the steep increase of robots within factories this work presents an elegant approach to keep robots functional until a maintenance is scheduled, reducing downtime.**

***Index Terms*—Fault-tolerance; Self Calibration; Adaptive Robotics; QR Code; Robot Arm;**

I. Introduction

Malfunctions on rotary encoders or electric motors within an industrial manipulator can cause great losses for factories. Long-term work hours increase the tendency of robot arms to suffer various faults. These faults not only shorten manipulators service life, but also make them unable to perform scheduled tasks in the factory, which may even lead to catastrophic consequences. Fault-tolerance is the property that enables a system to continue operating properly in the event of a failure in some of its components, and this property is very important when robot arms are needed to perform tasks in complex and unknown environments, like space applications.

Manipulator faults can be divided into the following three types:

1. Free swinging failure [1], [2], due to the actuator not being capable of actively exert any torque (or force). Another name is torque failure.

2. Position failure, which acts as if the actuator was locked [3], [4], [5], i.e., the joint cannot change its angle or length.

3. Sensor failure [2], where the joint value is perturbed by an unknown, possibly time-varying, value.

To the best of our knowledge, there are mainly two approaches for robots to adapt to malfunctions: model-based and learning-based. Works such as the one presented by [6] tackles the problem from a mathematical perspective, while others, such as [7], [8] and [9] use heuristics to converge to a solution.

From a model-based perspective, Goel et al. [6] focus on a case where one of the joints is locked but the controller continues to control that joint as though it were healthy. For a general class of tasks characterized by point-to-point motion, they examine convergence issues such as whether the manipulator comes to rest and, if so, what is the terminal position and orientation of the end-effector. Their application is limited, as they only consider joint position failure, which means that they already know the constrained angle value.

A great body of research has been combining learning algorithms with robotics, and some focus on applications on robot manipulators: Jutharee and Maneewarn [7] use a Genetic Algorithm optimization method to the 7 degree-of-freedom (DOF) of a semi-humanoid robot when a joint failure occurred. However, similarly to [6], their work assumes that the constrained angle value is known.

More recently, Cully et al.[8] developed a hexapod robot and a manipulator that can adapt its behavior after a malfunction by using a method called Intelligent Trial and Error (IT&E) algorithm. While IT&E is a combination of a simulation-based Genetic Algorithm parametric search with a Bayesian Optimization real-world heuristic, there is no knowledge of what the fault or optimum behavior is. In their data-driven approach their manipulator requires a few iterations to converge to a working solution, and in recent works the same authors propose a variant of the same algorithm for a policy-search within a Reinforcement Learning problem on a 4 DOF manipulator [9]. Although this work shows a remarkable performance, converging to an optimal solution within 35 seconds of iteration time, the computational time spent between iterations is very high. Additionally, the need for trials to converge to a working solution is far from desirable within industrial settings.

In our work, we propose a simple and inexpensive method to overcome both position failure and sensor failure. We use

TABLE I

Classic DH parameters

i	alpha(i-1)	a(i-1)	di	theta1
1	$\pi/2$	0	0.2755	q1
2	π	0.4100	0	q2
3	$\pi/2$	0	0.0098	q3
4	$\pi/3$	0	- 0.2501	q4
5	$\pi/3$	0	-0.0856	q5
6	π	0	-0.2028	q6

a QR code to build a virtual link to estimate the translation matrix between the two adjacent links, to calibrate the constrained angle and lastly to plan a new trajectory. We then draw a comparison between our work and previous works from other researchers, and discuss the range of applications where model-based and data-driven approaches can be used. This work (to the best of our knowledge the fastest solution for position failure, sensor failure, or the combination of these two) creates an easy-to-deploy solution for a common problem, in a few seconds and without trials. The impact of this work on industrial robotics will translate into thousands of dollars on savings for factories in a higher level of automation/robotisation.

In Chapter II we present the methods that we use for our experiments, introducing the algorithms and experimental settings. In Chapter III we present our results, while in Chapter IV we discuss these results, their limitations and their relevance in face of similar researches. Finally, in Chapter V we conclude this work.

II. Methods

A. Hand-eye Calibration

In order for the robot arm to use a video camera to estimate the 3D position and orientation of a part or object relative to its own base within the work volume, it is necessary to know the relative position and orientation between the hand and the robot base, between the camera and the hand, and between the object and the camera. These three tasks require the calibration of robot, robot eye-to-hand, and camera. We use the scheme Tsai et al. [10] proposed to calibrate the relative position between robot base and camera.

B. Denavit-Hartenberg parameters

In mechanical engineering, the Denavit-Hartenberg parameters [11](also called DH parameters) are the four parameters associated with a particular convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator. The Kinova Jaco^2 robot arm's DH parameters are shown in Table I.

C. Transformation Matrix

The transformation matrix from coordinate system A to coordinate system B has the following form:

$$BAT = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{1}$$

Where t is the translation vector and R is the rotation matrix. The form of t is shown as follows:

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

There are three forms of rotation matrix R according to the rotary axis:

$$R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3}$$

$$R_Y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{4}$$

$$R_Y(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

D. Calculating the joint angle

Inverse kinematics (IK) makes use of the kinematics equations to determine the joint parameters that provide a desired position for each of the robot's end-effectors. After setting up a desired position of the end-effectors, we can use the IK algorithm to calculate the joint parameters and send these joint parameters to the motor. However, there is a problem that if the motor and the encoder break, the end-effectors can not achieve the desired point. Usually when an industrial manipulator finds itself in this situation, the broken joint will be immovable due to the low back-drivability. Since the encoder also breaks, that means we don't know the angle of the constrained joint. To make the manipulator adapt from the broken joint, we propose a QR code-based method to calibrate the constrained angle.

Firstly, we install a web camera near the robot arm's base, and use the hand-eye calibration to calibrate the relative position between robot and camera.

Apriltag2 [12] is the QR code we decided to adopt and it allows us to have the exact coordinates of this QR code in the camera coordinate system. Utilizing this feature, we put the QR code on the end-effector so that once the camera sees the QR code it obtains its pose information. Standing from the perspective of DH parameters, we can regard this feature as a virtual link of the manipulator.

When one of the joints suffers from position failure [3], [4], [5], the previous kinematic chain breaks and we can use this virtual link to build a new one.

A schematic of how our algorithm works can be found in Fig. 1. Initially, as we have a manipulator with intact joints the transformation matrix $45\mathbf{T}$ is known *a priori*. Assuming, for example, that joint 5 is broken, we will have to build a new chain using the links between base, camera and QR code to build the new transformation matrix $45\mathbf{T}'$, using equation 6.

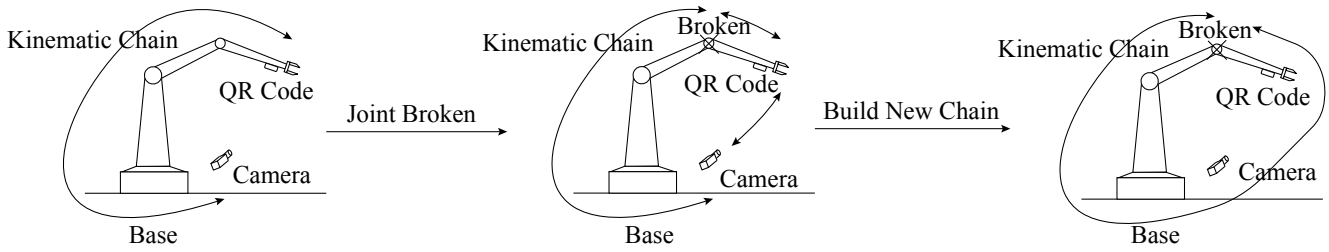

Fig. 1. (Left) The original kinematic chain, which is derived from Denavit-Hartenberg parameters. (Middle) One joint is broken and the kinematic chain is therefore also broken. A QR code on the end-effector is used to estimate the position and orientation of the end effector. (Right) The new kinematic chain.

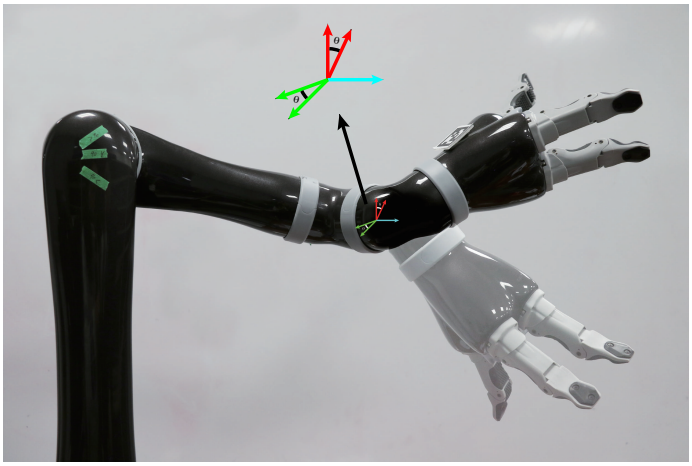

Fig. 2. The relative position between initial coordinate systems and constrained case coordinate systems.

In the equation, the subscript B means base, C means camera, QR means QR code, and E means end-effector.

$$45\mathbf{T}^{'} = 43\mathbf{T}32\mathbf{T}2B\mathbf{T}BC\mathbf{T}CQR\mathbf{T}QRE\mathbf{T}E5\mathbf{T} \quad (6)$$

At the initial position with our intact manipulator we record $45\mathbf{T}$ as a reference matrix. Under the assumption that we know which joint is broken (easily verifiable by a mismatch between control outputs and encoder inputs), we create another transformation matrix $45\mathbf{T}^{'}$, using a constrained joint 5 as an example. The difference between $45\mathbf{T}$ and $45\mathbf{T}^{'}$ lies in the rotation matrix, and the relative position between these two coordinate systems is shown in Fig. 2. Thus, we can obtain the $\cos\theta$ from the two rotation matrices, and consequently obtain θ from $\cos\theta$.

E. The adaptive algorithm

We use a variant of the Newton-Raphson method to solve the inverse kinematic problem of our manipulator. We should find the corresponding joint angles for every single joint, and the Newton-Raphson method enables us to find the root in an iterative process, given a specific x, y, z position. In our problem we need to find the right angles to minimize the difference between the current state position of the end effector of our manipulator and the target position.

Firstly we need to build nonlinear equations:

$$\begin{cases} F(\theta) = 0 \\ F(\theta) = (f_1, f_2, \ldots, f_{12})^T \\ \theta = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)^T \end{cases} \quad (7)$$

Each element f_i of $F(\theta)$ is derived by the following way: given the aim homogeneous transformation matrix $\mathbf{T_{aim}}$ of the hand coordinate system to the base coordinate system, $\mathbf{T_{aim}} = \mathbf{T_6T_5T_4T_3T_2T_1}$, and for each iteration, we can find a current state homogeneous transformation matrix $\mathbf{T}_\theta$, $\mathbf{T}_\theta = \mathbf{T}_{\theta_6}\mathbf{T}_{\theta_5}\mathbf{T}_{\theta_4}\mathbf{T}_{\theta_3}\mathbf{T}_{\theta_2}\mathbf{T}_{\theta_1}$. The two transformation matrices are both 4×4, but we just use the upper 3×4 elements ($\mathbf{T_{11}}$,$\mathbf{T_{12}}$,$\mathbf{T_{13}}$,$\mathbf{T_{14}}$,$\mathbf{T_{21}}$,$\mathbf{T_{22}}$,$\mathbf{T_{23}}$,$\mathbf{T_{24}}$,$\mathbf{T_{31}}$,$\mathbf{T_{32}}$,$\mathbf{T_{33}}$,$\mathbf{T_{34}}$), as the remaining four elements never change. We can subscribe all 12 corresponding entries from the $\mathbf{T}_\theta$ and $\mathbf{T_{aim}}$, one by one, to find $f_i, i = 1, 2, 3 \ldots 11, 12$.

Then we can obtain the partial derivatives on $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ from $F(\theta)$ as the Jacobian matrix J, which is 12×6, and we can find the left inverse of J by $(J^TJ)^{-1}J^T$ as the pseudo inverse J+. Lastly, we use the iteration in the Newton-Raphson algorithm to obtain θ^{i+1}. This workflow is demonstrated in Fig. 3.

As we have already calibrated the constrained angle, we update the angles at the end of each iteration and set $\theta_j^i = \theta_{calibrated}$, under the assumption that the jth joint is broken. Our iteration continues until the difference between θ^{i+1} and θ^i is below $1e^{-6}$, and then the trajectory is calculated.

As shown in Fig. 4, we register some noise after our calibration with the constrained angle is done, and we decided to take the average angle as our final result. In the figure in question the average angle is $57.6937°$. As the encoder of the Kinova jaco^2 is very accurate, we regard the value shown by the encoder as the ground truth, which is $57.7629°$, yielding in an error of $0.0691°$. Such value, below $0.1°$, is regarded by us as very precise and in par with error values offered by the manufacturer.

F. Experimental setting

We performed our experiments on a Kinova Jaco^2 manipulator. Firstly, we printed an Apriltag2 QR code with a side length of $3cm$ and placed it on the end-effector. We then 3D printed a camera stand and placed the camera on it, with the relative position between the robot base and this

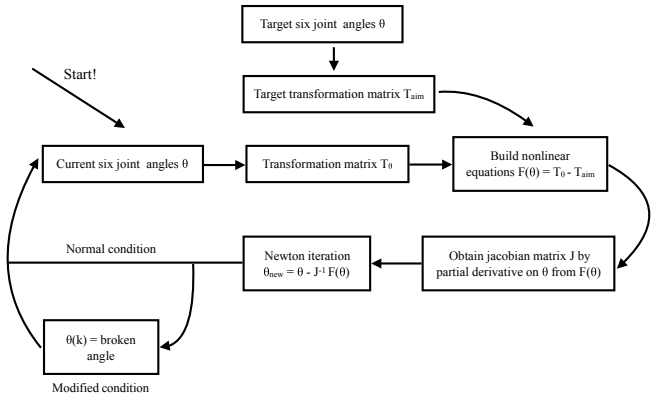

Fig. 3. The pipeline of the variant Newton-Raphson method. It starts from the current position, getting the current joint angles.

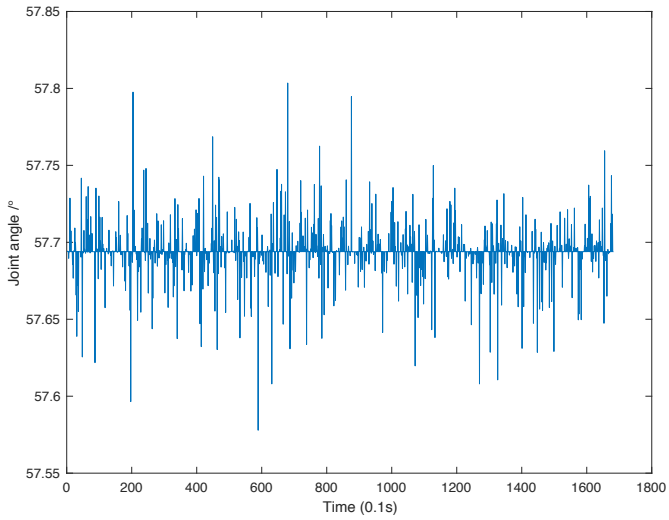

Fig. 4. The constrained angle we calibrated. When the camera saw the QR code, we kept calculating the constrained angle every 0.1 seconds. Due to the noise, the value of joint angle is fluctuating. But the general trend is around at $57.6937°$. So we regard it as the constrained angle.

camera being calibrated as proposed in Tsai et al. [10]. The transformation matrix $QRE\mathbf{T}$ between QR code and end-effector can also be obtained through this calibration. Kinova Jaco^2 manipulator has a home position and we start our experiment from it. The coordinate of our starting point is $(-0.32, -0.16, 0.51)$. In order to verify that our algorithm can work well in the x-axis, y-axis and oblique directions, we chose points A $(0.16, -0.25, 0.02)$, B $(-0.16, -0.25, 0.02)$ and C $(-0.16, -0.5, 0.03)$. In our experiments we mainly adopted paths connecting points A-B, B-C and A-C. A picture of our experimental setup can be found at Fig. 5

III. Results

Once the QR code is seen by the camera, our algorithm instantly starts to calculate the constrained angle, if any problem is detected. We performed series of pick-and place experiments to test our algorithm. As shown in Fig. 6, the

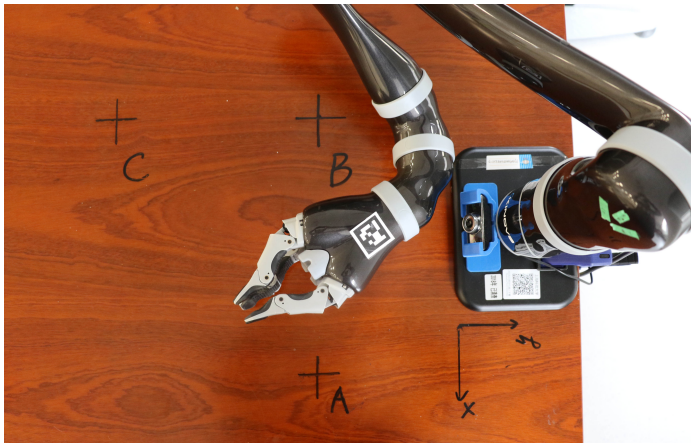

Fig. 5. The environment of our experiment. The QR code is placed on the end-effector, while the camera is at the base. Points A, B and C are used for our experiments.

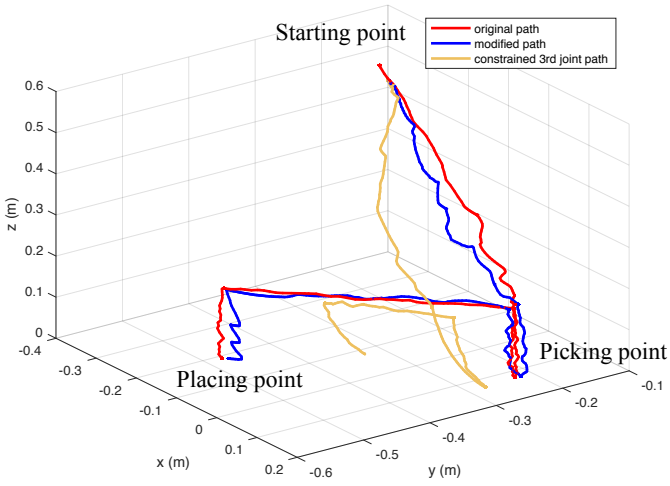

Fig. 6. Trajectories of three cases. The picking point is set to point A, shown in Fig. 5, and the placing point is set to point C. For modified path, we constrain joint 3 at an angle of $50°$. The end-effector first moves directly above the picking point, then closes the gripper and moves up. Then, it moves directly above the placement point, where it moves down to place the object.

red line is the trajectory when the robot arm is working normally and the yellow line is the trajectory when the robot arm's third joint suffers position failure. As one can see, the "defective joint" affects the trajectory of the manipulator and compromises the entire pick-and-place task. Through our adaptive algorithm, shown in blue, the robot arm succeeded with the task. As a pick-and-place activity is a combination of multiple tasks the computational time taken to calculate the new trajectory is close to three minutes, while in simpler operations, such as moving from point A to point B, this time can fall below one minute. Additionally, it is important to note that the constrained joint also affected the start position in a few millimeters.

In order to certify that we can reach the target under multiple conditions, and not by chance in a spurious experiment, we performed seven repeated experiments to move the end-

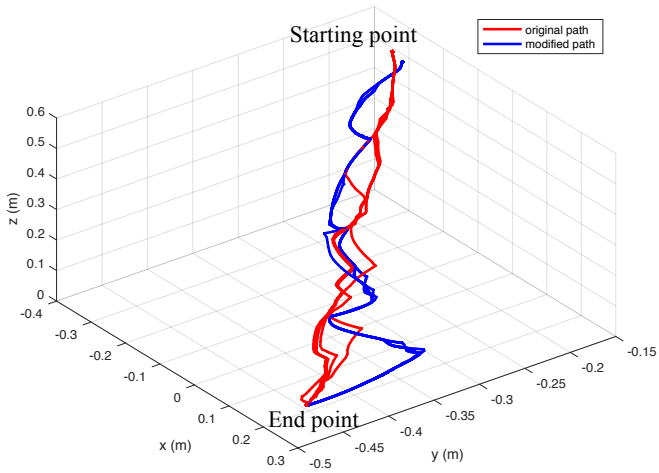

Fig. 7. The trajectories of the end-effector, from start point to end point, under the original algorithm and our Adaptive algorithm. Each algorithm has seven trajectories, and we constrained joint 3 at an angle of 70°. The seven trajectories of our algorithm are almost coincident.

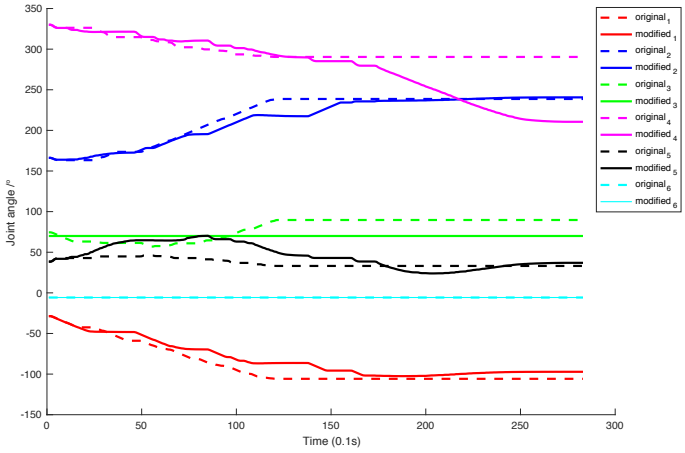

Fig. 8. The changes in six joints, from start point to end point, under the original algorithm and our Adaptive algorithm. We constrained joint 3 at an angle of 70°. The rotation of joints are different under the two algorithms, but eventually they reach the same target point.

effector from starting point $(-0.32, -0.16, 0.51)$ to end point $(0.20, -0.45, 0.02)$. Our results, in Fig. 7, show that although our algorithm predicts different corrective measures along the trajectory, the differences between these trajectories are very small, and our method always converges to the intended final goal in a high repeatability.

To better understand the influence of the differences seen in the previous experiment, we evaluated individual joints in the same point-to-point experiment. The Fig. 8 shows the rotation of six joints under the original algorithm and our Adaptive algorithm. It can be seen that due to the constraint in joint 3 (green) the rotations of the other joints had to adapt considerably.

As changes in joint angles of a manipulator might also affect negatively the final orientation of the end-effector, which could

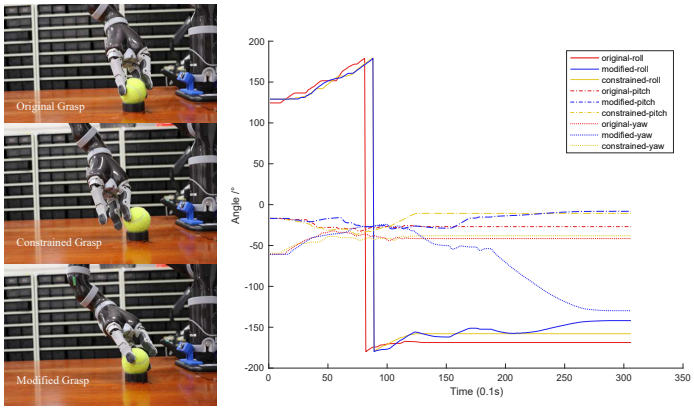

Fig. 9. The change in roll, pitch, yaw under two algorithms. Comparing the original, modified and constrained paths we can see the changes in orientation.

prevent the manipulator from ultimately performing the action that it was intended to do, we also compared the roll, pitch and yaw of the end-effector under these two algorithms. As can be seen from Fig. 9, when the joint is constrained the end-effector will deviate from the target and the grasp will fail. At the same time, although our adaptive algorithm successfully captured the target object, the orientation of the end-effector changed.

Finally, we decided to verify the extent to which the manipulator can still maintain function after a range of malfunctions on different joints. We chose different angle values to constrain our joints at, and focused on joints 3, 4 and 5. We chose these joints based on the work from [13] . According to this work, DOFs from a manipulator can be divided into two groups: Major DOFs (MDOFs) which are critical in performing a task, and Secondary DOFs, which are less important and replaceable to a certain degree. If a SDOF suffers position failure, the manipulator has a higher probability to modify itself and reach the goal.

TABLE II
THE ROBUSTNESS OF OUR ALGORITHM WITH DIFFERENT CONSTRAINED ANGLE IN DIFFERENT JOINTS

Constrained Angle	Joint 3	Joint 4	Joint 5
30°	FAILURE	SUCCESS	SUCCESS
50°	SUCCESS	SUCCESS	SUCCESS
70°	SUCCESS	SUCCESS	SUCCESS
90°	FAILURE	SUCCESS	SUCCESS
110°	FAILURE	SUCCESS	FAILURE
130°	FAILURE	SUCCESS	FAILURE
150°	FAILURE	SUCCESS	FAILURE

As the Table II shows, joint 3 is a MDOF, so the operation range of this joint in the eventuality of a malfunction is limited (between 50° and 70°). Since joint 4 is a secondary DOF (SDOFs)[13] , the tolerable range is very large. However, even though joint 5 is a SDOF, due to the long length of the link the tolerable range is of 60°. We assumed that joint 1 and 2 are MDOFs, and failure in those joints would

immediately constrain the positioning of the end-effector. Joint 6, on the other hand, although redundant when positioning is considered, is very important for the orientation of the end-effector. A video with our experiments can be found here: https://youtu.be/WaoXPJuKEvs .

IV. Discussion

A. Influence on orientation

With regards to the orientation after the malfunction, the results from Fig. 9 and Table II lead us to believe that our algorithm can be used as a useful back-up plan to be implemented while a definitive solution (or the scheduled maintenance) takes place. As objects are approached from above in pick-and-place tasks, common in industrial assembly lines, the presence of a 180° dome of grasping affordances might not be a reality, and this can be a limiting factor for the success of our algorithm. However, previous authors, from either model-based [7] or learning-based [6] [8] 4 approaches were also hindered by similar problems, and we believe that the ideal solution would go beyond adaptation, by allowing robots to fix their motors/sensors.

Malfunctions are not all the same: When a motor malfunction takes place instead of a sensor malfunction, there is no need for a self-calibration. The works [6] and [7] assume knowledge of all joint angles, and in this case the problem can be bluntly written as a reduction to $N-1$ DOF, with a frozen link. Sensors can break, cables can be disconnected, or just suffer from unknown noise sources, and when the closed-loop motor control is compromised our self-calibration combined with the adaptation algorithm is vital for the task continuity. The precision of our algorithm in predicting the angle of the constrained joint, shown in Fig. 4, demonstrates that we can reliably overcome the absence of a sensor on a joint by adding vision to our system.

B. Try not. Do, or do not

Solutions can range from model-based to learning-based approaches, but from the assumption that robot arms are manufactured by engineering companies we can easily conclude that a prior knowledge of that chosen morphology already exists. Ignorance over such prior, starting from a *tabula rasa*, will slow down convergence as previously demonstrated with a five dimensional real-world-based problem taking 72 hours to converge [14].

Simulation-based learning, on the other hand, could drastically reduce this time, but these are often used as benchmarks for robots learning new tasks, not old ones. As a computer simulation requires a model to simulate, the presence of such model can speed up iterations, but if the manipulator model is accurate (intact joints and links) the computational time to reach a Cartesian coordinate through iterations would be superior to the same task with Inverse Kinematics.

Simulations combined with real-world, as shown by [8] and [9], can be very powerful methods to account for malfunctions. These works show that a simulation can take place in an inaccurate model to create a coarse prior knowledge, while data-efficient real-world iterations can minimise the iteration time and guarantee transferability to the real-world. Such methods can account for motor and sensor malfunctions, and even go beyond, also assuming broken links. However, one major drawback from these methods lie in the computational time: while our method takes a maximum of three minutes for a multi-point pick-and-place and less than one minute for single point-to-point tasks, learning algorithms are notoriously intensive on their approach to data. As an example, the 15 seconds of iteration time from [9] in a cart-pole balancing problem required 25 minutes to compute 8 learning episodes.

In addition to the time taken to converge to a solution, prior-based physical learning (or also called Grey-box methods, as they have Black-box and White-box elements [15]) requires trials to find a solution (the higher the required precision the more trials). From an industrial perspective, having to produce defective products to reach a commercialisable final product is unacceptable. Beyond being easily implemented with one inexpensive camera, guaranteeing a high precision of the end-effector positioning, and taking a few minutes to calculate a new trajectory, in our proposed method "there is no try".

C. Limitations

The position error that we present in this work can still be considered too big for specific industrial applications, and it is an accumulation of the hand-eye calibration error, camera calibration error and QR code error. Still, since we only use one inexpensive camera with one ink-jet printer-printed QR code there is still room for improvement in the final precision, by combining higher resolution cameras with multiple QR codes and go below the millimetric scale.

From Fig. 7 we could see that our proposed trajectory suffers minor deviations while following their path, and we believe that such differences arise from the truncation error on sensor sampling and motor commands. It is important to observe that the original path planning from Kinova Jaco^2 manipulator also showed a similar behavior, albeit at a smaller scale. Although the uncertainty associated with the new path might represent a problem for industrial settings with highly-constrained operation spaces, this possible problem can be easily prevented with the addition of via points to the original trajectory. Additionally, our Adaptive algorithm is only meant as a stop-gap procedure until a formal repair can be implemented to the manipulator during the scheduled maintenance.

The time to calculate a new trajectory, which can reach three minutes of computation, can still be further improved. As we are currently using Python, a high-level scripting language, the execution is slow. We are currently rewriting our algorithm in C language to maximise performance. Additionally, we also plan on modifying the Jacobian matrix, shown in Fig. 3 in a iterative process [16], or multi-thread (CPU cores or FPGA) the calculation of the IK to increase the speed.

One of the current shortcomings for our calibration method is when the QR code is not within the visible range of the camera (due to the angle at which the malfunction happened), and the virtual link cannot be established. In such cases the

manipulator needs to rotate the remaining normal joints of the robot arm to let the QR code enter the camera's field of view. Alternatively, we can increase the number of QR codes to reduce the probability of such eventuality happening.

V. Conclusion

We proposed a method capable of overcoming position failure and sensor failure on an industrial manipulator, with a high accuracy, quick computational time and inexpensive. Since our method only uses a web camera and a printed QR code per manipulator, the cost of upscaling this method to allow every manipulator within a factory to adapt to malfunctions is negligible. Also, as our method utilizes known models of commercially available robots it can be easily deployed to a plethora of robot arms and snake robots.

In the future, we believe that our method will be very useful for fully autonomous factories and on search-and-rescue snake robots. The probability of failures increase in systems with a large number of robots and of joints, and our method permits these systems to keep going without compromising on precision. In the future we will extend this work to allow the simultaneous calibration of two or more constrained joints, and also test similar techniques to bring precision to soft robotics.

References

[1] J. D. English and A. A. Maciejewski, "Fault tolerance for kinematically redundant manipulators: Anticipating free-swinging joint failures," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 566–575, 1998.

[2] ——, "Measuring and reducing the euclidean-space effects of robotic joint failures," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 1, pp. 20–28, 2000.

[3] A. A. Maciejewski, "Fault tolerant properties of kinematically redundant manipulators," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 638–642.

[4] C. J. Paredis, W. F. Au, and P. K. Khosla, "Kinematic design of fault tolerant manipulators," *Computers & electrical engineering*, vol. 20, no. 3, pp. 211–220, 1994.

[5] R. G. Roberts and A. A. Maciejewski, "A local measure of fault tolerance for kinematically redundant manipulators," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 543–552, 1996.

[6] M. Goel, A. A. Maciejewski, and V. Balakrishnan, "Analyzing unidentified locked-joint failures in kinematically redundant manipulators," *Journal of Robotic systems*, vol. 22, no. 1, pp. 15–29, 2005.

[7] W. Jutharee and T. Maneewarn, "Gesture reconfiguration from joint failure using genetic algorithm," in *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, Oct 2016, pp. 1137–1142.

[8] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015.

[9] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 51–58.

[10] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[11] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. ASME E, Journal of Applied Mechanics*, vol. 22, pp. 215–221, June 1955.

[12] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[13] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[14] A. Rosendo, M. Von Atzigen, and F. Iida, "The trade-off between morphology and control in the co-optimized design of robots," *PloS one*, vol. 12, no. 10, p. e0186107, 2017.

[15] A. Kroll, "Grey-box models: Concepts and application," *New Frontiers in Computational Intelligence and its Applications*, vol. 57, pp. 42–51, 2000.

[16] Y. Chen, J. E. McInroy, and Y. Yi, "Optimal, fault-tolerant mappings to achieve secondary goals without compromising primary performance," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 4, pp. 680–691, 2003.