CONTENTS

ROCO504
Catch-bot

Tom Queen
School of Computing,
Electronics and Mathematics
Plymouth University
Plymouth, Devon PL4 8AA
Email: xxxx

Daniel Gregory-Turner
School of Computing,
Electronics and Mathematics
Plymouth University
Plymouth, Devon PL4 8AA
Email: xxxx

Demetrius Zaibo
School of Computing,
Electronics and Mathematics
Plymouth University
Plymouth, Devon PL4 8AA
Email: xxxx

Abstract—**This article discusses the development and construction of a Translational Planar 4-cable Cable-Direct-Driven Robot (CDDR), using soft robotics practices, and its applications in catching and throwing. The CDDR is tested for performance and reliability.**

I. INTRODUCTION

This article discusses the development and construction of a Translational Planar 4-cable Cable-Direct-Driven Robot (CDDR), using soft robotics practices, and its applications in catching and throwing. The CDDR is tested for performance and reliability.

Section II-B of this document discusses the background research, linking to the main design challenges faced when constructing a catching robot. II-D details the various attempted solutions and their resultant outcomes. System capabilities are experimentally analysed in IV, followed by an overall summary in V.

January 4, 2018

II. DESIGN PROCESS

A. Problem

1) Inverse Kinematics: Initially, it was thought that the commands to each motor could be generated by looking directly at the returned X and Y coordinates of the tracked objects. To move the gripper upwards, the top two motors should rotate clockwise and the bottom two anticlockwise. To move the gripper to the left, the two left motors should rotate clockwise and the right two anticlockwise. This led to the following kinematic solution:

$$
\begin{aligned}
M1 &= Y - X \\
M2 &= Y + X \\
M3 &= -Y - X \\
M4 &= -Y + X
\end{aligned}
\tag{1}
$$

For high torque motors, elastic cords and a closed loop between the tracked object and the gripper, this approximation may have been functional. However, it would not have been accurate, slack cords would be common and it would unnecessarily load the motors. When using stepper motors with low-current drivers, this solution caused the steppers to skip if the

gripper was directed more than a few centimetres from the centre of the working area. This led to a re-evaluation of the kinematic solution as can be seen in section II-D4.

2) Force storage: For a robot to throw an object kinetic energy needs to be added to the system. To impart more force to the object than the motors can instantaneously provide a method of storing the force will have to be implemented.

3) Hardware Requirements: In order for a gripper to track a target, a low latency high FPS camera and fast motors will be needed. Section II-B8 describes motor speed requirements and III-G describes the camera which was selected for this task.

B. Research

The problem of designing and implementing a catching robot is not new, and although a thorough analysis of design considerations for catching robots in general is out of the scope of this article a brief summary of related issues will be outlined. The interested reader should read [1] for a more detailed analysis.

1) Body Design: Body design determines the shape of the overall robot, setting many project constraints such as the available workspace, control complexity and physical capabilities of the robot. There are numerous possible designs such as robot arms, as used by [2], [3] and [4], or frame-based robots, as used by [5], and [6]. Due to project time constraints, a simple 2-axis frame based 4-cable CDDR was designed as the main body.

2) Cable-Direct-Driven Robots: Cable-Direct-Driven Robots (or CDDRs) are a type of parallel manipulator wherein the end-effector link is supported in parallel by n cables with n tensioning motors [7]. CDDRs can be made lighter, stiffer, safer and more economical than traditional serial robots [8] since their primary structure consists of lightweight, high load-bearing cables. More complex Cable-driven parallel robots can be designed to move in six Degrees of Freedom (DoF) in three-dimensional space, but these are beyond the scope of this article.

3) Gripper Design: Gripper design refers to the end effector which grasps the thrown object and constrains the design in what objects can be caught and how said objects can be

manipulated. Gripper designs can be distinguished between 2 pairs of classes, passive verses active and soft verses hard. In the first pairing, passive grippers provide no means of control within the gripper itself which can simplify design and control considerations but limits future capabilities. Active grippers increase the design and control complexity by adding actuation, or modifiable gripper characteristics, to perform a wider variety of tasks. Hard grippers are constructed out of rigid, inflexible, materials. These grippers are commonplace within industry where they work with known objects that are strong enough to not break under high stresses. Outside of the industrial setting, however, they are less applicable. This is where soft grippers which can deform and spread the gripper forces over a larger surface area find more usage.

4) Camera Setups: Camera setups typically provide a trade-off between system complexity and control complexity. More cameras, or ones with higher frame rates and resolutions, allow for better object tracking but come at the cost of additional processing and control requirements. There are three common camera setups.

- The simplest setup for control is the eye-in-hand approach, where a camera is mounted inside the gripper, providing a direct feedback loop so long as motion blur doesnt become an issue.
- The most reliable setup for object tracking involves placing multiple cameras around the room in fixed, known locations. From this, the ball position and trajectory can be easily modelled in 3D space, but at the expense of a complex setup and a significantly higher processor demand.
- A middle ground to the previous two is to use head mounted cameras, or rather, cameras mounted to a fixed known location on the robot's body. This typically means that the target cannot be tracked in 3D space as easily, but may present more reliable results that the eye-in-hand solution.

5) Object Tracking: Object tracking consumes the majority of processor resources in such projects. Accuracy, false positives and false negatives determine the reliability of a system. Cameras are typically the sensor(s) of choice for robot catching systems due to the flexibility and often ease, that detection algorithms can provide. Computer vision provides numerous methods of feature detection, a small sample of such techniques include:

- *Colour thresholding*, whereby a colour range of interest is selected and a binary image is produced. Position is estimated using the centre of mass, or via structural analysis.
- *Structural analysis* looks at the edges and corners in a given image in an attempt to detect predefined shapes.
- *Statistical transformations* convert an image, or set of images, into fuzzy regions of interest. Blob detection, Farneback optical flow and convolution are examples of such methods.
- *Depth maps*, generated from multiple images or through

special hardware, provide a form of 3D representation of the environment, allowing other simpler algorithms to detect moving objects and the 3-dimensional direction of motion.

6) Gripper-Object Coordination: Gripper-object coordination is the main control algorithm used in the catching process and defines the efficacy and efficiency of the system. It is usually broken down into numerous input and output controllers. Input controllers relate the sensory data to desired movements of the machine, and output controllers relate this desired motion into actuator commands.

For vision-based sensors, there are two methods of providing this feedback. If the camera is inside the gripper then visual servoing is used, where an offset from the centre of the camera image directly translates to a desired velocity vector. Other systems employ predictive feedback, which predicts a possible location the object will intersect the robot and provides this as a target location for the gripper.

Given an input which directly relates to spatial coordinates or movements, the system can then convert these into actual movements. At its simplest this involves the use of forwards and inverse kinematics, where forward kinematics provides an internal representation of the current system state, and inverse kinematics provide the actuator positions to attain the desired system state. For more complex systems other parameters, including the system centre of mass and force storage, may need to be taken into account.

7) Object Grasping: Object grasping refers to post-catch manipulation of the object. More manipulability often implies more complex gripper designs. For instance [9] use a high speed 3-fingered robotic hand to perform a wide variety of tasks, however, this approach is over-engineered if all that is required of the robot is to catch, a task a standard household bin can perform with the right aim. For the purposes of the project presented in this article the two tasks of interest are the catching and throwing of an object.

8) Motors: Fast motors were needed for the gripper to move quickly enough to catch an object. Our motor speed requirements were based off an object thrown from $2.5m$ away. The object would be thrown at a velocity of $5m/s$ at an angle of $45°$. The time of flight would be $721ms$ and the maximum height of the object would be $0.637m$. Worst case, the gripper would have to travel from corner to corner across the centre of the frame, resulting in a minimum travel speed of $1.635m/s$. If a spool diameter of $10cm$ was used, the required motor speed is $5.2\ RPS$ or $312\ RPM$.

C. Methodology

Given the problem breakdown discovered in the research, it was decided to split development into three sections initially. The first focussing on the frame design and hardware; the second looking into gripper designs; and thirdly, the control system including visual feedback.

For each section, many conceptual designs would be explored and analysed. Afterwards, all ideas would be reviewed and the most achievable and functional designs selected. Some

of the analysis involved creating prototypes as there was insufficient information to make a practical decision otherwise.

After implementing all of the selected solutions, each section was tested for functionality independently. On success, the sections were integrated together and the whole system tested. Design flaws at this stage either meant an integration fault, or that flaws in a subsection weren't readily obvious when tested in isolation and hence took longer to remedy. Finally, the system was tested for full functionality and reliability.

D. Prototypes

Throughout the project a lot of time was spent attempting to get a setup which could physically meet the aims of the project. To this end several design iterations occurred.

1) Motors: Two motor setups were prototyped. The first was to use high torque ($300Ncm$) encoded DC motors (E192.24.125). The maximum speed of these motors was 33rpm. As the required motor speed was 312RPM and the micro motor could only provide 33rpm, a 1:9.45 ratio gearbox was required. Given the high torque output from the motors this should have provided enough speed and torque to move the gripper. However, even after many gearbox design iterations design flaws kept occurring, ranging from gear slippage to material failure. By making the length of each gear inversely proportional to its accumulated speed multiplier, the stress felt by the input gear could be spread out over the rest of the gears. However, as this would have more than doubled the size of our original gearbox, it was determined that an alternative approach may yield better results.

The second prototype involved the use of SST58D3820 stepper motors from an old ER1 robot from Evolution Robotics. These motors are specified to hold $7.3Kg.cm$, which drops to $6.5Kg.cm$ at a frequency of 1200 pulses per second (PPS). In full step mode the stepper shaft rotates $1.8°$ per step giving a max speed of 6RPS and thus a maximum cord velocity of $188.5cm/s$.

$$\frac{a \cdot p}{s} = \frac{1.8 \cdot 1200}{360} = 6RPS \qquad (2)$$

$$\frac{\pi \cdot d \cdot a \cdot p}{s} = \frac{\pi \cdot 10 \cdot 1.8 \cdot 1200}{360} = 188.49cm/s \qquad (3)$$

Where a is the step angle, p is pulses per second, d is the spool diameter ($10cm$) and s is the desired speed in RPM. After a small amount of reverse engineering the motors were tested with their original motor driver which, unfortunately, did not appear to provide enough current to meet the motors torque limit; because of this the TB6600 motor driver was selected instead. What was not discovered until quite late in the project, after several control algorithms had been tested, was that the stepper drivers performed significantly worse than their specifications. Without any encoders closed loop control was not possible causing the actual gripper position to drift wildly from the internally recorded gripper position over a short span of time. After further consideration the stepper motors were also abandoned.
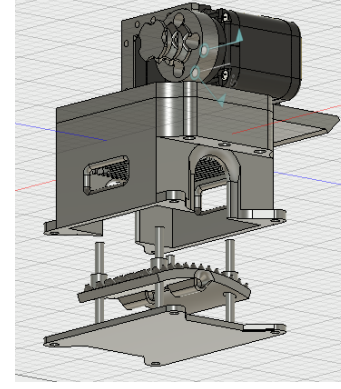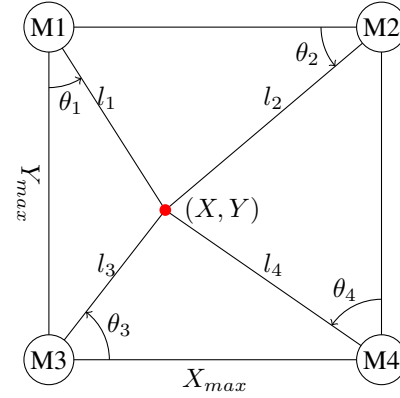

Fig. 1. Exploded view of clamp

Fig. 2. Kinematic diagram with point gripper

2) Camera: The first camera used to prototype various algorithms was a standard USB webcam. It quickly became apparent that objects moving at speed would either present unacceptable amounts of motion blur, or low frame-rates and high latency would not give the control system long enough to react. To resolve these issues a high FPS camera was selected.

3) Clamps: To address the problem discussed in II-A2, clamps were designed to lock each cord in place. Each clamp consisted of a high-friction surface suspended above a fixed plate via four springs. The cord to be clamped runs between the two surfaces. A dynamixel AX-12 servo is connected the suspended surface with a pulley. When actuated, the servo pulls the suspended surface towards the fixed plate, closing the gap and clamping the cord in place. When released, the four springs push the suspended surface away from the fixed plate, quickly releasing the clamped cord. A clamp was produced for each of the five cords, one for each corner of the frame and one for the throwing motor. An exploded CAD model of the developed clamp can be seen in figure II-D3.

4) Kinematic solution 1: To address the issues discussed in II-A1, the following kinematic model was produced. Since no rotational motions and no moment resistance are required at the end-effector, our kinematic model can be simplified to assume that all cables meet at a point [8] in the centre of our end-effector.

Inverse Kinematics:

$$\cos(\theta_3) = \frac{X_{max}^2 + l_3^2 - l_4^2}{2 * X_{max} * l_3} \quad (4)$$

$$X = l_3 \sin(\theta_3) \quad (5)$$

$$Y = l_3 \cos(\theta_3) \quad (6)$$

Or, without trigonometry:

$$X = \left(\frac{X_{max}^2 + l_3^2 - l_4^2}{2 * X_{max} * l_3} \right) = \frac{X_{max}}{2} + \frac{l_3^2 - l_4^2}{2 * X_{max}}$$
$$(7)$$
$$Y = \left(\frac{Y_{max}^2 + l_4^2 - l_1^2}{2 * Y_{max} * l_3} \right) = \frac{Y_{max}}{2} + \frac{l_4^2 - l_1^2}{2 * Y_{max}}$$

Forward Kinematics:

$$l_1 = \sqrt{(X)^2 + (Y_{max} - Y)^2}$$
$$l_2 = \sqrt{(X_{max} - X)^2 + (Y_{max} - Y)^2}$$
$$l_3 = \sqrt{(X)^2 + (Y)^2} \quad (8)$$
$$l_4 = \sqrt{(X_{max} - X)^2 + (Y)^2}$$

5) Control Software: A Teensy 3.2 was configured to use the rosserial library to receive the targets CofM from the computer. The teensy then offset the coordinate system so that pixel 0,0 was in the centre of the image. The teensy kept track of all motor positions, and when a new CofM arrived the current gripper position was calculated from the motor positions using the forward kinematics described by equation 8 in II-D4. The program generates motor commands designed to move the gripper so that the centre of the camera image contains the CofM of the target. This is known as visual servoing. The x and y error between the centre of the camera image and the target is calculated, generating a vector that points towards the target. This vector is translated into the change in length of each cord required to centre the gripper over the tracked target using the inverse kinematics described by equation 7 in II-D4. The desired gripper location is checked to see if it has left the bounds of the working area. If it has, movement in the direction of the axis which has been breached is set to zero. Finally, motor speeds are calculated from the desired changes in lengths as is described in equation 11 in section III-B.

E. Vision Algorithms

An ideal object tracking algorithm would be algorithmically simple, quick to process, and be rugged against the various error sources vision systems encounter, whilst reliably detecting the location of the unknown object with some metric of how far away it is.

Several algorithms were attempted and one failed to achieve this somewhat ambitious target. Some may still be feasible but given the time-frames available they were considered too complex. The four attempted algorithms will be briefly summarised below.

1) Changing Histograms: The first method was intended to provide a metric to identify interesting objects for further analysis. Histograms, using various colour channels, can be used to define how much of an image is dedicated to a given colour and light intensity. As an object approaches a camera it becomes larger, and thus consumed more of the image which should manifest as a positive rate of change over relevant histogram sections.

Two issues prevented this algorithm from being used within the project. The first is that lateral motion of an object coming in or out of view would flag as an incoming object, which could be resolved by explicitly checking for these boundary conditions. The second issue is that as an object travels, the lighting over the object changes and distributes itself inconsistently across the histogram. This is more prevalent indoors, and using a camera with fixed settings may have improved the situation.

2) Optical Flow (Farneback): Optical flow can break up the motion between two frames of an image into lateral and vertical movements. As an object increases in size between images its motion can be used to create an outline around the object. This method was hampered due to interference from horizontal and vertical movements, where methods to reduce this became too complex to implement within the projects timeframe.

3) Kalman filter: To protect against the object being obscured when moving towards the gripper a 2D Kalman filter was implemented. It was found that, whilst an acceptable prediction of object position was calculated, the object was often lost before its maximum height was reached. This resulted in the prediction from the Kalman filter continuing vertically, making the solution inappropriate for our purpose. In future work as it is possible to estimate the velocity of a ball of known size from its circumference in the camera feed it is would be possible to combine this with our 2d Kalman filter to acquire a 3d estimate of trajectory. If an algorithm was produced to detect the ball had reached its maximum height and had begun falling, the predictive features of the Kalman filter could be used to help with latency issues and slow motors.

4) Pyramid Scaling: The third method revolved around how smaller objects would show up less on lower resolution images (or in images with a larger blur applied). This method is equivalent to using a variety of difference of Gaussians (DoGs) to detect edges and attempting to determine the size of an object between the DoGs. The main issue with this method was, again, lateral motion hiding the depth information.

5) YOLO: As a final method the You Only Look Once (YOLO) neural-network based object detection system was implemented to detect a small sample of objects. This method was hampered again by hardware capabilities as, even with YOLO-Tiny, a maximum of 7 frames per second were achieved which was considered too slow for the projects application.
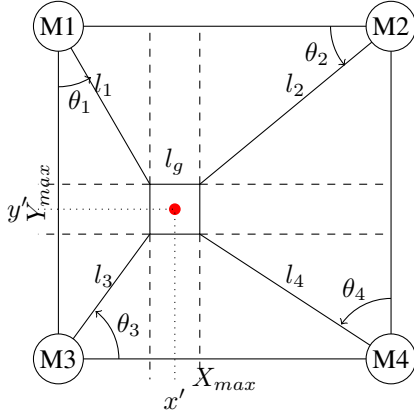
F. Further problems

Fig. 3. Kinematic diagram with square gripper

Fig. 4. The final prototype

1) Kinematics: Initially the kinematic solution considered the gripper as a point. This worked for preliminary testing, but soon proved to be a problem when the limited torque of the stepper motors required equal tension on all cords at all times. This led to the development of the following kinematic model, which considered the gripper as a square.

Inverse Kinematics:

$$X = \frac{X_{max}}{2} + \frac{l_3{}^2 - l_4{}^2}{2\left(X_{max} - l_g\right)}$$

$$Y = \frac{Y_{max}}{2} + \frac{l_3{}^2 - l_4{}^2}{2\left(Y_{max} - l_g\right)}$$

(9)

Forward Kinematics:

$$l_1 = \sqrt{\left(X - \frac{l_g}{2}\right)^2 + \left(Y_{max} - Y + \frac{l_g}{2}\right)^2}$$
$$l_2 = \sqrt{\left(X_{max} - X + \frac{l_g}{2}\right)^2 + \left(Y_{max} - Y + \frac{l_g}{2}\right)^2}$$
$$l_3 = \sqrt{\left(X - \frac{l_g}{2}\right)^2 + \left(Y - \frac{l_g}{2}\right)^2}$$
$$l_4 = \sqrt{\left(X_{max} - X + \frac{l_g}{2}\right)^2 + \left(Y - \frac{l_g}{2}\right)^2}$$

(10)

2) Skipping steps: The prototype was assembled with SST58D3820 stepper motors and TB6600 drivers. The T-B6600 is designed to deliver up to 3.5A RMS. The drivers current can be limited down to 0.8A RMS in eight steps. Performance in the SST58D3820 datasheet is rated for when the motors are being supplied with 2.0A per phase. From this, we can see that the stepper motor requires 250mA per phase more than the driver can provide. When the robot was switched on the motors could barely hold the weight of the gripper when stationary. Experiment IV-A was performed with the results showing that the stepper driver would not be suitable for this
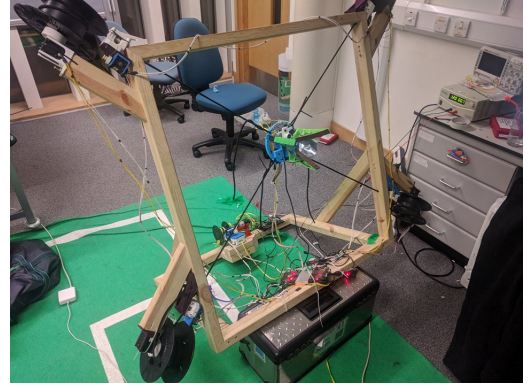
robot. This issue could have been partly mitigated if the motors were encoded.

3) Motor velocities: For the tension to remain equal on all cords during motion of the gripper, the motors must turn at different rates. For example: if the gripper starts in the centre of the frame and moves upwards, the top two lengths will shorten and the bottom two cords will get longer. To maintain a uniform velocity of the gripper, the top two motors must slow down and the bottom two must speed up. For gripper trajectories constrained to the centre of one axis this isn't much of a problem, but for trajectories traversing multiple axis the motors pull against each other resulting in steps being skipped and tension lost.

III. IMPLEMENTATION

A. Software

Frames enter the object tracking node at a rate of 60FPS. The object tracker performs a series of filters in different colour spaces on the image before calculating its centre of mass. This coordinate is published to the kinematic controller 17.5ms after the frame enters the object tracker. Upon entering the kinematic controller, the coordinate frame is offset so that the centre of the camera image is now at pixel 0,0. The required change in length of each cord is then calculated and set (via the method described in II-F1). From the desired changes in length, the required speed of each motor is calculated and set. The kinematic controller node then calculates the current gripper position in order to calculate the changes in length for the next loop. Figure 5 shows the high-level software flow diagram of the robot.

B. Motor velocities

To address the issue raised in II-F3 a simple speed scaling system was devised. This algorithm was called once per main loop after the required changes in lengths are calculated. The algorithm takes in the requested changes in length of each cord and divides each change in length by the largest. The
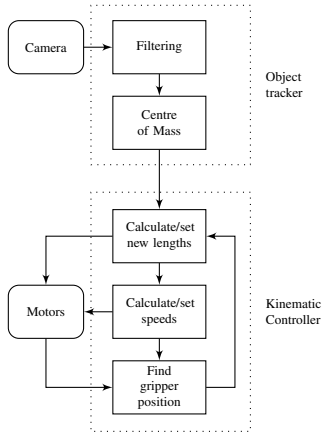
Fig. 5. High-level software flow diagram.

relative speeds of each motor are then scaled by the global speed scalar.

$$\vec{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} \qquad x = \max\left(\vec{l}\right) \tag{11}$$

$$x = \max \vec{l}$$

$$\vec{s} = \frac{g}{x} \cdot \vec{l}$$

Where g is the global speed scalar and s contains the speed of each motor.

C. Servos and multi-threading

Due to the points mentioned in II-F2 the decision was made to change the motors to Dynamixel MX-64 servos. The MX-64 can supply $60Kg.cm$ at 62 RPM, giving a top gripper speed of $0.32m/s$ which is considerably slower than our requirement. The MX-64 servos communicate over either an RS-485 or a TTL bus and can be daisy-chained if required. Due to the nature of the communication protocols involved in the dynamixel bus (clarify), the rate at which commands could be sent to each servo were reduced with each successive servo added to the daisy chain. Initially the four servos responsible for translating the end-effector were daisy-chained to a single dynamixel bus resulting in a command rate of 7Hz. This was increased to 15Hz by increasing the baud rate of the servos from 57.6kb/s to 1Mb/s. The reduced control rate led to the decision to use separate busses for each time-critical motor resulting in a command rate per motor of 60Hz. Another bus was used for all five clamps, and a final sixth bus was used to actuate the gripper as the uncertainty of the command latency to this servo was required to be as low as possible in order to successfully pre-empt and act upon the target landing in the gripper.

When a change in servo speed is requested, the dynamixel driver blocks the process thread that requested the change. The
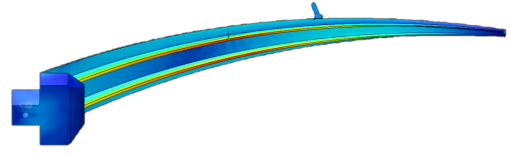

Fig. 6. Finite Element Analysis of one leg of the passive gripper

driver waits for a response from the motor before it allows the thread to continue. This reduced the rate at which motor instructions left the kinematic controller program to 10Hz. Creating four additional process threads for each of the speed-set service calls brought the loop rate back to 60Hz.

D. Processor

As servos were to be used instead of stepper motors, a microcontroller was no longer required for the control of the end-effector. Software previously running on the Teensy (forward/inverse kinematics, boundary checking etc...) was ported to a C++ program on a laptop running ROS on Ubuntu.

E. Throw Sequence

Once a ball is caught, the internal state machine switches to throwing mode. The first step here is to either aim where to throw the ball or to centre the gripper to shoot forwards. This is then followed by tensioning the four outer motors, calibrating how far the ball should be thrown, and clamping their respective cables to store the energy.

The firing sequence consists of pulling the gripper, using the back motor, a set distance; clamping to store the added energy; unwinding the back motor so as to be able to be able to throw continuously without the use of manual resets, and unclamping the back motor to release the gripper forwards. A short time after releaseing, the gripper opens up to release the ball and, eventually, the four side motors are similarly unclamped, ready to catch another ball.

F. Grippers

Due to the complexities of gripper design, two distinct grippers were constructed.

1) Soft Passive Gripper: A soft passive gripper design was created due to the simplicity and speed of manufacture. Figure [figure 2] shows an image of the finished gripper. Being 3D printed from PLA meant a prototype could be rapidly made and tested for functionality. With legs detachable from the gripper frame, several different designs and configurations could be tested. The legs went through multiple designs iterations, improved by finite element analysis (FEA) models. Stresses and deflections of the system were modelled using a static force of 1N applied uniformly over the catching face.

Several designs were tested, varying the number and thickness of the struts. ABS, rubber (as an analogue to NinjaFlex) and PLA (after importing the material properties [10]) materials were tested. Figure III-F1figure 3, 3mm] shows the FEA results for the finalised design.

Due to the FEA iterative design process the legs worked as expected on the first print, with only minor design modifications to attach to the frame necessary. The object to be caught was initially a stress-ball which was used to generate requirements for the passive gripper. This was eventually replaced with a tennis ball, which meant the legs were sometimes too weak to capture it.

2) Soft Active Gripper: The soft active gripper design has the potential to guarantee that objects (if caught) wont slip out, and can more readily accommodate a larger variety of objects. Figure [figure 4] shows an image of the finished gripper.

The design, which is predominantly a replica of FESTOs Fin Gripper [11], was selected due to its tried and tested nature and overall simplicity. A CAD model of the design was edited to add a means of actuating the gripper, three methods were considered; first was a slider-crank mechanism, however, this was decided against due to the size of the mechanism that would be needed to actuate the gripper. Second, the gripper was redesigned to add a rack-pinion gear to each finger to enable actuation. This was decided against as this would have required three extra dynamixel motors with large gears to actuate each finger at an acceptable speed. The final design altered the shaft which actuates the gripper to be used as a rack gear which could then be actuated by a single dynamixel motor.

G. Camera

The finalised camera setup incorporated a single PS3-Eye camera in an eye-in-hand configuration. The PS3-Eye was selected due to several useful characteristics:

- 187fps at 320x240 resolution
- Software-modifiable camera properties
- Lightweight after removing the casing
- £0.50 at time of purchase

Owing to the high frame rate, motion blur became less of a concern when using the eye-in-hand configuration, which allowed for a simplified control algorithm. The camera can be seen attached to the active gripper in Figure [figure 4].

H. Object Tracking

The final object tracking algorithm could reliably track a red tennis ball up to 4 meters away. It was found that colour thresholding was the only solution to object tracking which could be developed within the time frame of the project, whilst also providing reliable, high-speed object recognition. An object is selected by clicking on the centre of the object in the camera feed. An average value for hue, saturate and variance (HSV) is then found from pixels surrounding the point selected. Using the HSV values, a minimum and maximum value are created by offsetting the average values by a predefined amount. These HSV values are then used to create a binary thresholded image which is dilated to remove image noise which appear as black speckles. Contour detection is performed on the binary image, the output contours are then run through OpenCVs minEnclosingRect function which will create the minimum rectangle which encloses the entire contour. To remove some unwanted contours, the difference of height and width of each created rectangle are compared and the contour is rejected if the difference is too large. With the remaining contours, a difference between the previously accepted objects CofM and the CofM of each contour is taken. The difference is used to find the distance between the previously accepted objects CofM and the contours CofM, the contour with the smallest distance is then accepted to be the object and published to the kinematic controller.

IV. EXPERIMENTS

A. Stepper Driver Performance

1) Method: To assess the performance of the TB-6600, an experiment was performed where commands were sent to the robot instructing the gripper to move to the top corner and then return to the centre with cord lengths measured before and after.

2) Results: It was found that requesting any movement of the gripper resulted in the stepper motors skipping steps, often resulting in the gripper dropping tens of centimetres over short movements.

3) Discussion: Further investigation showed that the stepper drivers were drawing minimal current, and would overheat and shutdown if their current limit was set to more than half of what they were rated for. Even with a current limit set at 2.0A RMS, each stepper never drew more than 1.0A. This experiment showed that these drivers would not be suitable to drive the stepper motors in our project as they could not drive the stepper motors anywhere close to the specifications in the motors datasheet.

B. Object tracking program rate

1) Method: Timers were added to the object tracking program. The system time was taken when a camera frame entered the program and again when a CofM left the program. The difference in time was saved to a file every time the program looped. The program was run for ten minutes, after which a mean was calculated from the saved durations.

2) Results: The recorded loop rate of the object tracking program was $17.5ms$, or 57Hz.

3) Discussion: The camera frame rate during this test was set to 60Hz. As the recorded loop rate is close to 60Hz, it is believed that the object tracking module could complete a loop faster than the camera could generate frames. A useful future experiment would be to also calculate the duration between the object tracking module finishing and a new camera frame arriving.

C. Kinematic controller rate

1) Method: Timers were added to the kinematic controller program. The system time was taken when a coordinate entered the program and again when motor positions and speeds were published. The difference in time was saved to a file every time the program looped. The program was run for five minutes after which a mean was calculated from the saved durations.

2) Results: The kinematic controller takes $17ms$ to run, giving a loop rate of 60Hz.

3) Discussion: This experiment was performed in order to see whether latencies in the kinematic controller were preventing the robot from catching the ball. The program only runs when a CofM from the object tracking program arrives. As the camera frame rate was set to 60Hz during this experiment, it was concluded that the kinematic controller was probably completed before the next CofM arrived. A useful future experiment would be to also calculate the duration between the program finishing and a new CofM arriving.

D. Camera latency

1) Method: The camera was connected to a computer and the video feed was viewed on the computer monitor. The camera was pointed at a digital stopwatch on a smartphone so that the stopwatch could be seen on the computer monitor. A second camera was orientated so that both the stopwatch and the computer monitor showing the video of the stopwatch could be seen and a photograph was taken.

2) Results: The photograph on the second camera showed a time of 4.187 seconds on the computer monitor and 4.112 seconds on the stopwatch. The difference in these times is $75ms$ which is the latency of the USB camera used to track the ball in this project.

3) Discussion: A thrown ball travelling at $5m/s$ at $45°$ will be falling at approximately $3.5m/s$ when it falls past the height at which it was thrown. A latency of $75ms$ results in the ball falling $26cm$ further than the camera reports. Without predictive algorithms, this latency will prevent the system from working correctly.

E. Catching Performance

1) Method: A ball was thrown twenty times from three meters away from the final version of the robot with the thrower aiming at the gripper, and 20 times with the thrower aiming within the frame but not at the gripper. The robot was angled so that the gripper was raised by $25°$ towards the ceiling.

2) Results: The robot caught the ball seven out of twenty times when the ball was aimed at the gripper. When the ball was not aimed at the gripper, the robot did not catch the ball once.

3) Discussion: The robot does not perform as originally designed. The main issue is the speed of the servos and the camera latency.

V. OVERALL DISCUSSION

A. Overall Discussion

From the experiments performed it is clear that the object tracking and gripper-object coordination challenges were successfully solved throughout this project. It is also clear that, even with a functioning system, there is still much work left to be done. This section shall briefly review each of the 6 challenges, outlining successes and areas of further development.

B. Body Design

Although the system does function, experiment E clearly shows that it is non-functional. There is no single root cause, however, experiment A indicates that motor selection and power is, and has been, a continuous issue throughout this project. As such, a clear design improvement would be an improved motor and motor driver setup. Also noted in the methodology of experiment E is how the robot is tilted during the experiment. This was because it was observed to have an improved reliability at this angle. Hence, another improvement and test set would involve a built-in mechanism to alter the robots inclination and determine usability at various levels.

C. Gripper Design

Although an experiment was not explicitly created to test the gripper functionality, it was observed that both passive and active grippers functioned reliably as intended. The passive gripper had an issue with the ball passing straight through it when thrown at a high enough velocity, so a recommended improvement would be incorporating netting toward the back to prevent this. Further developments in gripper design would involve creating and testing hard versus soft gripper designs.

D. Camera Setup

The camera setup worked with reasonable success. Given the high fps and resolution of the camera, objects were reliably trackable even in the eye-in-hand configuration. However, Experiment D provides insight into the main failure of this setup, camera latency. This is a difficult issue to resolve with standard USB 2.0 based cameras. A recommended improvement would be sourcing a camera using the USB 3.0 protocol or investing in other communication systems to resolve this issue.

E. Object Tracking

Experiment B indicates that the tracking algorithm requires little processing to go from a raw input to a tracked object. However, this setup is limited to balls of known colour, preferably red. This is a huge limitation on the systems capabilities. As such, further development would involve trialling various different tracking algorithms to allow a wider range of objects to be detected. Testing under more lighting conditions and environments, to ruggedise against related errors, would also help improve the system.

F. Gripper-Object Coordination

Via the use of encoded motors in the MX-64's, and a reliable tracking algorithm, the system control was successful. Experiment A demonstrates the necessity for the motor encoders, however, other alternatives could be devised to alleviate this requirement and enable a wider range of motors to be selected for improved body design. One such method would be to use distance sensors on the gripper to determine its x-y coordinate.

G. Object Grasping

The scope of this project did not allow for complex gripper designs, so advanced object manipulation has not been accomplished. Having said this, although the throwing mechanism is untested, it does function. Improvements to this setup would involve accurate mathematical models to determine the distance and trajectory of a thrown object, as well as determining a control algorithm to throw in the desired direction.

VI. Conclusion

This document has outlined the development process for a CDDR catching and throwing robot. The resulting system used under-specified components due to the originally sourced motors underperforming and budget constraints. A reliable control system with positional and object location feedback was presented. Two successful gripper designs were created for the purpose and were functionally analysed.

Further development would involve sourcing motors capable of achieving the desired tasks, and modifying the base so as to more readily experiment with angled catchment areas. Detailed models of the stored energy in the throwback mechanism, along with expected trajectory planning, would also be of use.

Acknowledgment

References

[1] M. Sorg, "Visual tracking and grasping of a dynamic object: from the human example to an autonomous robotic system," 2003.

[2] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, Oct 2014.

[3] J. Malzahn, "Modeling and control of multi-elastic-link robots under gravity," Ph.D. dissertation, 2014.

[4] B. Buml, F. Schmidt, T. Wimbck, O. Birbach, A. Dietrich, M. Fuchs, W. Friedl, U. Frese, C. Borst, M. Grebenstein, O. Eiberger, and G. Hirzinger, "Catching flying balls and preparing coffee: Humanoid rollin'justin performs dynamic and sensitive tasks," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3443–3444.

[5] R. Ritz, M. W. Mller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4972–4978.

[6] Omron.com, "Overview of forpheus technologies," 2017. [Online]. Available: https://www.omron.com/innovation/forpheus_technology.html

[7] S.-R. Oh and S. K. Agrawal, "Cable suspended planar robots with redundant cables: Controllers with positive tensions," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 457–465, 2005.

[8] R. L. Williams II and P. Gallina, "Translational planar cable-direct-driven robots," *Journal of Intelligent and Robotic Systems*, vol. 37, no. 1, pp. 69–96, May 2003. [Online]. Available: https://doi.org/10.1023/A:1023975507009

[9] A. Namiki, Y. Imai, M. Ishikawa, and M. Kaneko, "Development of a high-speed multifingered hand system and its application to catching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3, Oct 2003, pp. 2666–2671 vol.3.

[10] M. Jamshidian, E. A. Tehrany, M. Imran, M. Jacquot, and S. Desobry, "Poly-lactic acid: Production, applications, nanocomposites, and release studies," *Comprehensive Reviews in Food Science and Food Safety*, vol. 9, no. 5, pp. 552–571, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1541-4337.2010.00126.x

[11] Festo, "Bionictripod with fingripper," 2009. [Online]. Available: https://www.festo.com/cms/en_corp/9779.htm