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*—The abstract goes here.**

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.

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

January 4, 2018

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

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 re-evaluation of the kinematic solution as can be seen in section II-D5.

2) Force storage:

3) Camera speed: In order to track the target quickly, a low latency high FPS camera was needed.

4) Motors: Fast motors were needed to allow the gripper to keep track of the target. The working area of the frame measured XXXX 90cm by 75cm. If the gripper was initialised to the centre of the frame and a ball was thrown from three meters away, then the robot would have 0.7 seconds to move from center to the corner of the working area. PROOF

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 for this article, a brief summary of related issues will be outlined. The interested reader should reference [article reference 1] for a more detailed analysis.

1) CDDR: CDDRs are a type of parallel manipulator wherein the end-effector link is supported in parallel by n cables with n tensioning motors [1]. CDDRs can be made lighter, stiffer, safer and more economical than traditional serial robots [2] 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.

2) 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 [article reference 2], [article reference 3] and [article reference 4], or frame-based robots, as used by [article reference 5], [article reference 6] and [article reference 7]. Due to project time constraints, a simple 2-axis frame based 4-cable CDDR was designed as the main body.

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. For a brief overview of the many varieties of gripper design, the interested reader is referred to [article references 9 20].

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, mounted to a fixed known location on the robots 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 positive and negative rates determine the reliability of a system. For robot catching systems cameras are typically the sensor(s) of choice 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, and 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 of 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 and provides this position as feedback. 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 or 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 [article reference 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 is catching, 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.

C. Requirements

D. Solutions

To achieve the required speed and torque several motor solutions were evaluated.

1) Micro motors: The first was to use high torque ($300Ncm$) encoded DC motors (E192.24.125). The maximum speed of these motors was 33rpm. In order to achieve the required gripper speed of 1.5m/s a gearbox was needed.

$$\frac{s}{\pi \cdot d \cdot r} = \frac{150}{\pi \cdot 10 \cdot 0.55} = 1 : 8.68 \tag{2}$$

Where d is the spool diameter ($10cm$), s is required cord speed ($150cm/s$) and r is motor speed ($0.55\ RPM$). This ratio would result in a maximum cord velocity of $149.9cm/s$.

2) Stepper motors: Another solution was to use stepper motors. Four SST58D3820 stepper motors were acquired. 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 $188.5cm/s$.

$$\frac{1.8 \cdot 1200}{360} = 6RPS \tag{3}$$

$$6 \cdot \pi \cdot 10 = 188.49cm/s \tag{4}$$

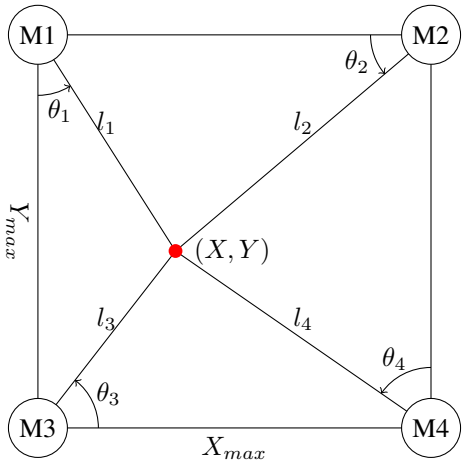


Fig. 1. Kinematic diagram

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.

4) Camera: The Sony Playstation3 eye camera can be purchased second-hand for 50p. This camera can output 187 frames per second (FPS) at a resolution of 320x240, or 60 FPS at a resolution of 640x480. In addition, the camera allows for control of exposure, gain, white balance, saturation and hue shift.

5) 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 [2] in the centre of our end-effector.

Inverse Kinematics:

$$\cos(\theta_3) = \frac{{X_{max}}^2 + {l_3}^2 - {l_4}^2}{2 * X_{max} * l_3} \tag{5}$$

$$X = l_3 \sin(\theta_3) \tag{6}$$

$$Y = l_3 \cos(\theta_3) \tag{7}$$

Or, without trigonometry:

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

$$Y = \left(\frac{{Y_{max}}^2 + {l_4}^2 - {l_1}^2}{2 * Y_{max} * l_3}\right) = \frac{Y_{max}}{2} + \frac{{l_4}^2 - {l_1}^2}{2 * Y_{max}} \tag{8}$$

Forward Kinematics:

$$\begin{aligned} l_1 &= \sqrt{(X)^2 + (Y_{max} - Y)^2} \\ l_2 &= \sqrt{(X_{max} - X)^2 + (Y_{max} - Y)^2} \\ l_3 &= \sqrt{(X)^2 + (Y)^2} \\ l_4 &= \sqrt{(X_{max} - X)^2 + (Y)^2} \end{aligned} \tag{9}$$

E. Prototype

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 involved using Micromotors E192-2S attached to a custom $1 : 20$ speed-ratio gearbox. 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. As such, 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 . After a small amount of reverse engineering the motors were tested with their original motor driver which, unfortunately, couldnt provide enough current to meet the motors torque limit. After switching the motor driver to the TB6600 the system performed somewhat expectedlyXXXXXXXXXXXXXXXXXXXXXXXXXX. What wasnt discovered until quite late in the project, after several control algorithms had been tested, was how unreliably the system performed. Without any encoders, closed loop control was not possible, causing the gripper position to drift wildly over a short span of time. After further consideration, the stepper motors were also abandoned.

2) Visual Feedback Loop Hardware: 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) 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 9 in II-D5. 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 center the gripper

over the tracked target using the inverse kinematics described by equation 8 in II-D5. 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 12 in section II-H1.

F. 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 failed, to achieve this somewhat ambitious target. Some may still be feasible, but given the timeframes 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 being 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) Pyramid Scaling: The third method revolved around how smaller objects would show up less on smaller 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.

4) 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.

G. Further problems

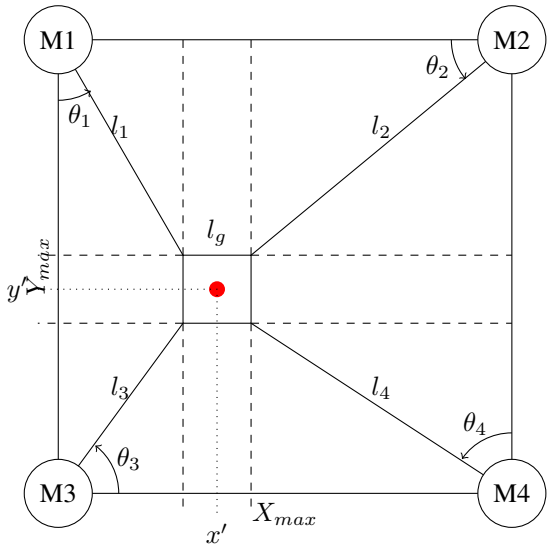
1) Gearbox: A gearbox ratio of 1:8.68 was chosen as can be seen in II-D1. If a 500g gripper were to be raised along either vertical wall of the workspace, the output of the gearbox raising it would experience $5Kg.cm$ of force. Transferring this force through the gearbox, a 1cm input gear would experience $43.4Kg.cm$ of force. 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 (SIZESXXXXXXXXXXX), the decision was made to use stepper motors instead.

2) Stepper Drivers: Four TB6600 were purchased to drive the SST58D3820 stepper motors. The TB6600 is designed to deliver up to 3.5A RMS. The drivers current can be limited down to 0.8A RMS in eight steps. The 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. An issue arose

drivers were connected to the stepper motors and the current to the driver was measured, the stepper driver

3) Skipping steps: The prototype was assembled with SST58D3820 stepper motors and TB6600 drivers. When the robot was switched on the motors could barely hold the weight of the gripper when stationary. 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. 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. 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 issue could have been partly mitigated if the motors were encoded.

4) 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:

$$X = \frac{X_{max}}{2} + \frac{{l_3}^2 - {l_4}^2}{2\left(X_{max} - l_g\right)}$$
$$Y = \frac{Y_{max}}{2} + \frac{{l_3}^2 - {l_4}^2}{2\left(Y_{max} - l_g\right)} \tag{10}$$

Forward Kinematics:

$$\begin{aligned} l_1 &= \sqrt{\left(X - \frac{l_g}{2}\right)^2 + \left(Y_{max} - Y + \frac{l_g}{2}\right)^2} \\ l_2 &= \sqrt{\left(X_{max} - X + \frac{l_g}{2}\right)^2 + \left(Y_{max} - Y + \frac{l_g}{2}\right)^2} \\ l_3 &= \sqrt{\left(X - \frac{l_g}{2}\right)^2 + \left(Y - \frac{l_g}{2}\right)^2} \\ l_4 &= \sqrt{\left(X_{max} - X + \frac{l_g}{2}\right)^2 + \left(Y - \frac{l_g}{2}\right)^2} \end{aligned} \tag{11}$$

5) 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 center of one axis this isn't much of a problem, but for trajectories traversing multiple axis xxxxxxxxxx

6) Motors:

7) Arduino:

H. Redesign

1) Motor velocities: To address the issue raised in II-G5 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 relative speeds of each motor are then scaled by the global speed scalar.

$$\vec{l} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \end{bmatrix} \qquad x = \max\left(\vec{l}\right)$$
$$x = \max \vec{l} \tag{12}$$
$$\vec{s} = \frac{g}{x} \cdot \vec{l}$$

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

2) Servos: Due to the points mentioned in II-G3 the decision was made to change the motors to Dynamixel MX-64 servos. The MX-64 can supply $0.6Kg.cm$ 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.

3) 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.

I. Final solution

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 center 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-G4). 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 3 shows the high-level software flow diagram of the robot.

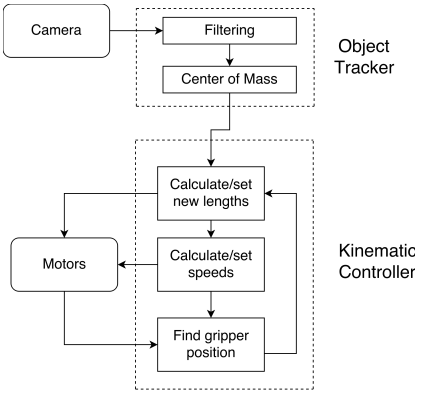


Fig. 2. High-level software flow diagram.

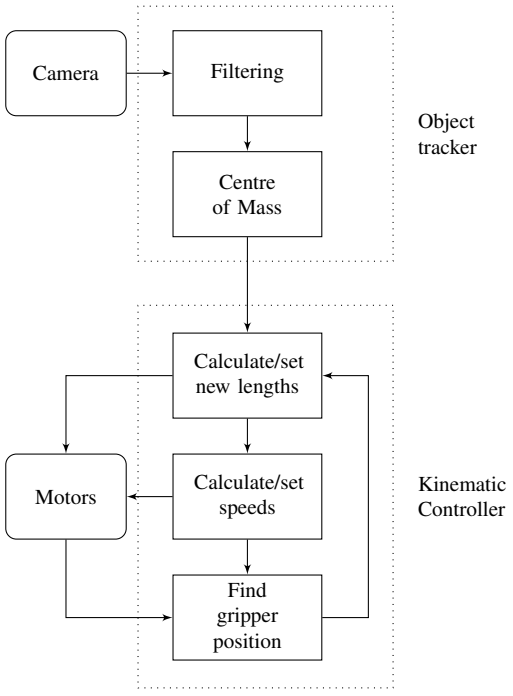


Fig. 3. High-level software flow diagram.

B. Frame

C. Kinematics

D. 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 separable from the frame several different designs and configurations could be tested. The leg designs went through multiple designs, iteratively improved by analysing 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 [article reference 8]) materials were tested. Figure [figure 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 stress-ball used initially 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 guarantees 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 [article reference 21], was selected due to its tried and tested nature, and overall simplicity.

E. Clamps

F. 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].

G. Object Tracking

The final object tracking algorithm could reliably track a red tennis ball up to 4 meters away. To achieve this:

- QV4L2 was used to customise the camera parameters, specifically:
  - Maximise the saturation
  - Shift the hue by 45 degrees
  - Turn off auto-gain, auto-white-balance and auto-exposure
  - Manually adjusting the brightness for the given environment
- OpenCV in C++ was used to:

- Auto-calibrate threshold parameters based on an initial selection
- Threshold raw images in the HSV colour space
- Erode the resulting binary image to remove false positives
- Find the centre of mass of the eroded image to approximate the balls location

Figure [figure 5] shows an example setup of the image processing algorithm, including the raw and thresholded images. Although this algorithm provided reliable results it has several significant limitations.

- Heavily optimised to find a single colour
- Environment must be free of the selected colour
- There is no camera calibration, introducing non-linear errors into the balls location

IV. Experiments

A. Method

B. Results

C. Discussion

V. Overall Discussion

[3]

VI. Conclusion

[4] [2] [5]

Acknowledgment

The authors would like to thank...

References

[1] 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.

[2] 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

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

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

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

References

[1] Homer J. Simpson. *Mmmmm...donuts.* Evergreen Terrace Printing Co., Springfield, SomewhereUSA, 1998

[2] ROSserial Wiki *Mmmmm...donuts.* Evergreen Terrace Printing Co., Springfield, SomewhereUSA, 1998
@onlinerosserial, author = Misc, title = Rosserial wiki, date = 30/12/17, url = http://wiki.ros.org/rosserial,

[3] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[4] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[5] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[6] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[7] K. Elissa, "Title of paper if known," unpublished.

[8] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[10] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.