# Abstract

* First paragraph of the introduction. Describe the purpose of the paper and what will be included in the findings, without giving spoilers as to what the findings actually are

# Introduction:

* Inc. requirements as project goals?

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. **Figure [figure 1]** shows an image of the constructed CDDR described in this article.

Aims of the project include:

* **CATCHMENT AREA REQUIREMENT**
* **MAX OBJECT WEIGHT REQUIREMENT**
* **MIN ACCEPTABLE SPEED REQUIREMENT**

# 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.

## **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.

## **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]**.

## **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 doesn’t 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.

## **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.

## **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.

## **Object Grasping**

Object graspingrefers 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.

# Prototypes:

## Hardware

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.

### 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 **[stepper motors]** from an old **[insert robot name here]**. After a small amount of reverse engineering the motors were tested with their original motor driver which, unfortunately, couldn’t provide enough current to meet the motors torque limit. After switching the motor driver to the **[insert motor driver name here]** the system performed somewhat expectedly. What wasn’t 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.

### Visual Feedback Loop Hardware

The first camera used to prototype various algorithms was a standard USB webcam. Although useful for some of these purposes, it quickly became apparent that objects moving at speed would either present unacceptable amounts of motion blur, or would simply reach the gripper too quickly for adequate control. To resolve this issue, a dedicated camera was selected and used. However, with the faster frame rates the selected processor, originally a Raspberry Pi 3, proved to be too slow and had to be replaced.

## 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.

### 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.

### 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.

### Pyramidal 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.

### YOLO

As a final method, the You Only Look Once (YOLO) algorithm 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 ***XXX*** frames per second were achieved, which was considered too slow for the projects application.

## Control

* Tom’s approximation / quad-copter approximation
* Stepper functionality test (Greenhill Zone)?

# Solutions:

## Body Design

* Frame design
* Dynamixel MX-64’s
  + Performance / limitations
  + Power supply / consumption
* Laptop as sole processor

## Gripper Design

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

### 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.

### Soft Active Gripper

The soft active gripper design guarantees that objects (if caught) won’t 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 FESTO’s Fin Gripper **[article reference 21]**, was selected due to its tried and tested nature, and overall simplicity.

## Camera Setup

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]**.

## 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

## Gripper-Object Coordination

* Visual Servoing
* Kinematic model
  + Square simplification
  + Smoothing, dead-band and boundary conditions
* Force storage + independence from kinematics
* Control process flow diagram

## Object Grasping

* Throwing mechanism
  + Clamp design
* Control process flow diagram

# Results & Discussion:

(Experiments, tests and results) + (Hind-sight + future work)

# Summary:

(Closing statements)

# References:

1. <https://www.semanticscholar.org/paper/Visual-tracking-and-grasping-of-a-dynamic-object-f-Sorg/242b24b232ea314ecdfcc8dbf0afbc0703711b2f>
2. <http://ieeexplore.ieee.org/document/6810147/>
3. <http://www.rst.e-technik.tu-dortmund.de/cms/de/Forschung/Schwerpunkte/Robotik/TUDOR_neu/index.html>
4. <http://ieeexplore.ieee.org/document/5980073/>
5. <http://ieeexplore.ieee.org/document/6385963/>
6. <https://www.omron.com/innovation/forpheus_technology.html>
7. <https://youtu.be/VypBbrvugW0>
8. <http://onlinelibrary.wiley.com/doi/10.1111/j.1541-4337.2010.00126.x/full>
9. <http://www.k2.t.u-tokyo.ac.jp/fusion/HighspeedHand/index-e.html>
10. <https://youtu.be/jWD-B796sao>
11. <https://youtu.be/90cXfaFM4O8>
12. <https://youtu.be/ZPUvA98uSj8>
13. <https://youtu.be/uPx8xwRpfFk>
14. <https://youtu.be/3OjhoVuAQkQ>
15. <https://youtu.be/0DgORq2bbTI>
16. <https://youtu.be/deHQfqFjkeM>
17. <https://youtu.be/9xSvRSg7VZA>
18. <https://youtu.be/nsnyl8llfH4>
19. <https://youtu.be/Bw0Ps8-KDlQ>
20. <https://youtu.be/DXkahXTPscs>
21. <https://www.festo.com/cms/en_corp/9779.htm>