# Abstract

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

# Research: (Encompasses the problem statement)

(Problem statement + discovered ideas)

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. 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.

1. 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.

1. 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.

1. 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.

1. 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. For the robot discussed this refers to the inverse kinematics and cable tensioning processes.

F) Object Grasping

Object graspingrefers to post-catch manipulation of the object. In the project presented in this article it refers to the method of throwing the object back towards the thrower, including force storage and release.

# Requirements:

(Do we really need to include these?)

# Prototypes:

(Failed / non-implemented solutions) [3 alternative vision methods]

## 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.

# Solutions:

(Finalised solutions and implementations) [passive gripper + FEA analysis]

# Results:

(Experiments, tests and results)

# Discussion:

(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>