Midlands Centre for Data-Driven Metrology (MCDDM)

Summer Internship 2023

Poster Book

MIDLANDS CENTRE
For Data-Driven Metrology

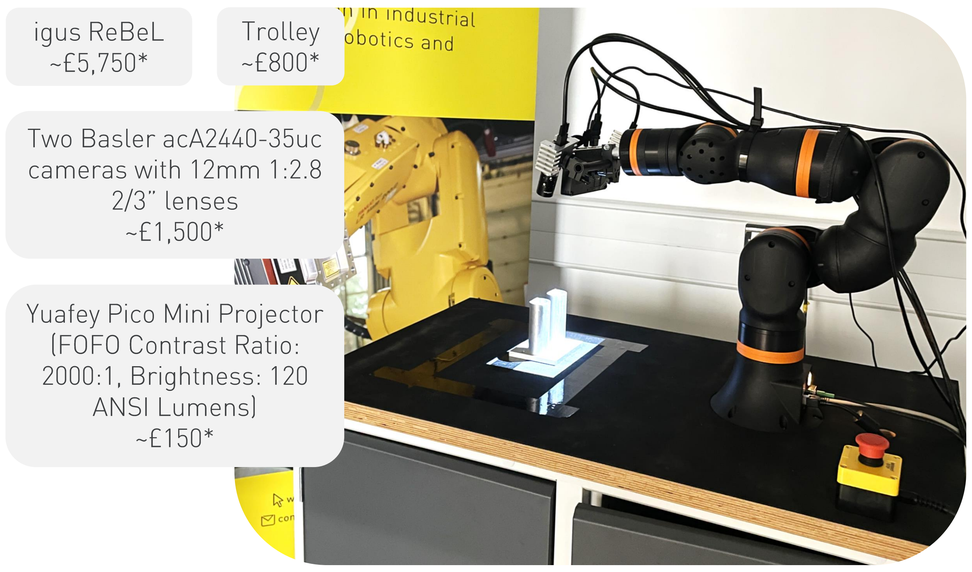
Loughborough University
Intelligent Automation Centre

Low-Cost Cobot Integrated 3D Scanning

MIDLANDS CENTRE For Data-Driven Metrology
Loughborough University
Intelligent Automation Centre

Thomas Mgbor, George Hair, Luke Hutchinson, Cong Sun, Wen Guo, Masoud Sotoodeh-Bahraini, Connor Gill, Peter Kinnell

The high cost of both 3D scanning and collaborative robots (cobots) has been a barrier of entry for manufacturers, particularly to those who are small to medium sized where the precision of typical 3D scanning systems often exceeds their requirements. The increase in the number of affordable cobots has allowed for a low-cost automated scanning system to become more accessible and financially viable. The aim of the project was to integrate a low-cost, lightweight fringe projection system attached to a six-degrees-of-freedom robotic arm and PC to create a portable scanning system.



Six DOF Cobot + Fringe Projection 3D Scanner → Program to Control Cobot and Scanner Simultaneously → Low-Cost Cobot *Integrated* 3D Scanning

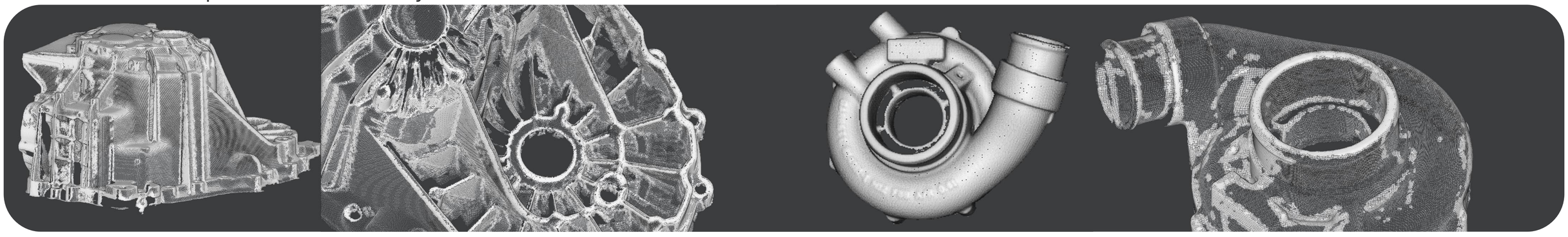
* Estimated prices only, excluding time, labour, machinery costs, PC and software.

- Non-specialist engineers (undergraduate student interns) were able to set-up, operate and program the solutions.
- Specialist engineers aided in integrating the solutions together.

Software used for both solutions included Polyga's FlexScan3D and CloudCompare:

- FlexScan3D generates the projection, interprets the fringe distortion and generates a point cloud.
- CloudCompare enables point cloud analysis and comparison.

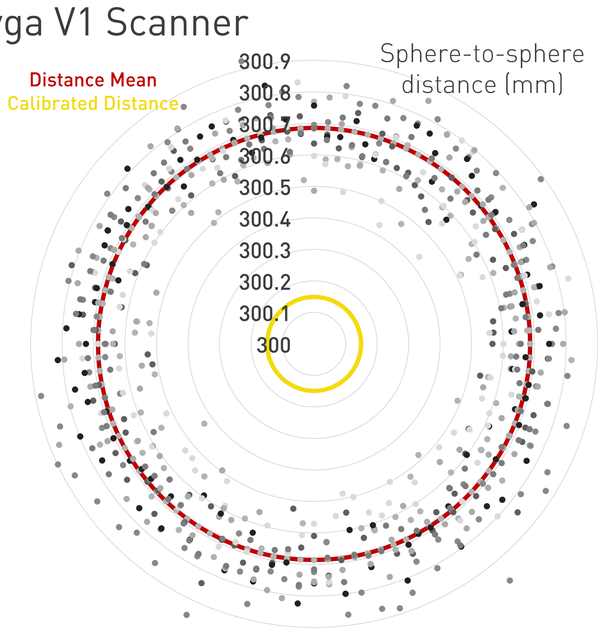
Point Cloud Examples of Scanned Objects:



VDI/VDE 2634 Evaluation:

Experiment 1: Polyga V1 Scanner

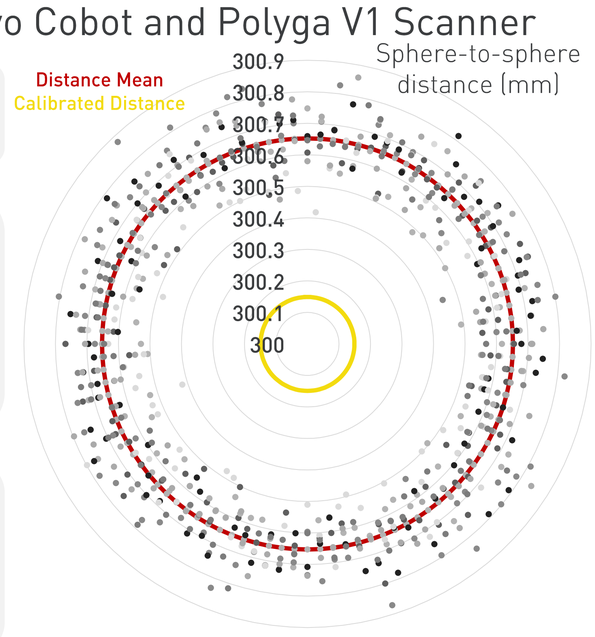
Scanner takes 100 scans consecutively. → Cobot moves scanner to new position and scanner takes 100 scans consecutively. → Process repeated until 7 positions are reached.



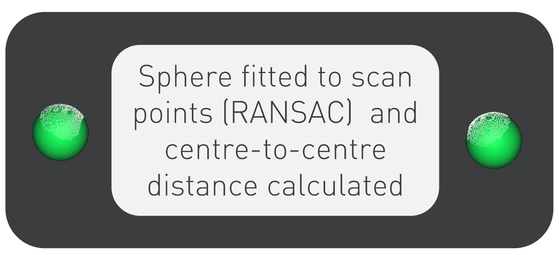
Sphere-to-Sphere Distance Mean	300.686 mm
Sphere Spacing Error	0.537 mm
Sphere-to-Sphere Distance Standard Deviation	0.084 mm

Experiment 2: Inovo Cobot and Polyga V1 Scanner

Scanner takes 1 scan. → Cobot moves to scanner to new position and scanner takes 1 scan. → Cobot moves to 7 total positions before returning to initial. → Process repeated until 100 scans are taken at each position.



Sphere-to-Sphere Distance Mean	300.652 mm
Sphere Spacing Error	0.503 mm
Sphere-to-Sphere Distance Standard Deviation	0.076 mm



Calibrated Ball Bar Sphere-to-Sphere Distance: 300.149mm

Conclusions:

- Non-specialist engineers are able to set-up a system but may need specialist help to ingrate the system.
- The code written can be adapted for the size of the object being scanned and detail required.
- The Inovo and Polyga V1 Scanner solution is relatively precise, with a standard deviation < 85 µm but not as accurate, with a sphere spacing error of ~500 µm.

Next Steps:

- Develop a software that fully integrates the technologies with ease for the user.
- Experiment with more off-the-shelf low-cost hardware.
- Expand parameter testing.
- Automatic scanner calibration using computer vision.

Thomas Mgbor LinkedIn: George Hair LinkedIn: Luke Hutchinson LinkedIn:

p.kinnell@lboro.ac.uk
www.mcddm.ac.uk
www.intelligent-automation.org.uk

3DMC 3D METROLOGY CONFERENCE
3DMC 2023 Bilbao

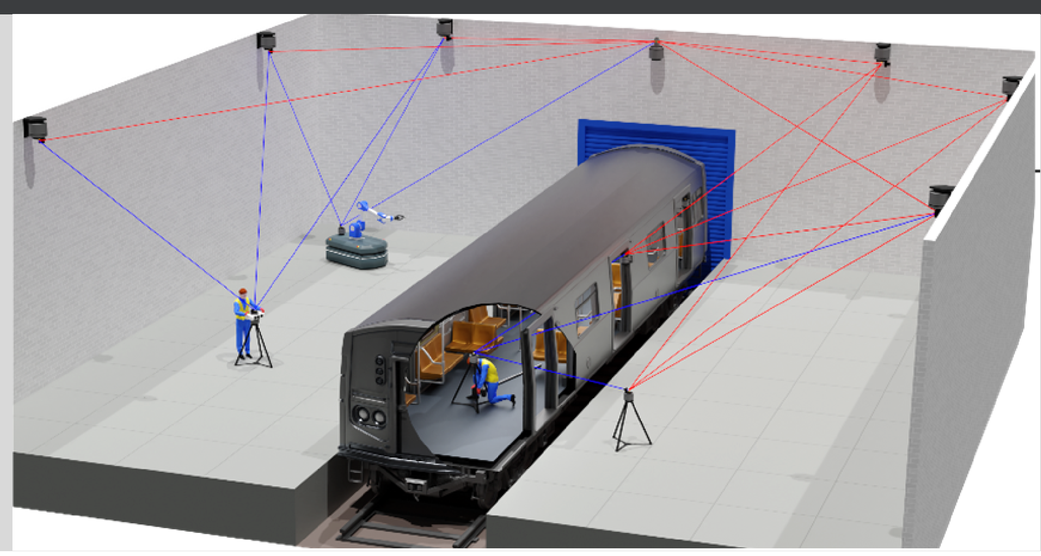
Fast Steering Mirror Network

MIDLANDS CENTRE For Data-Driven Metrology

Loughborough University
Intelligent Automation Centre

Prabrup Chana, Tayyab Farrakh, Sam Muddimer, Peter Kinnell, Jeremy Coupland, Wen Claire Guo

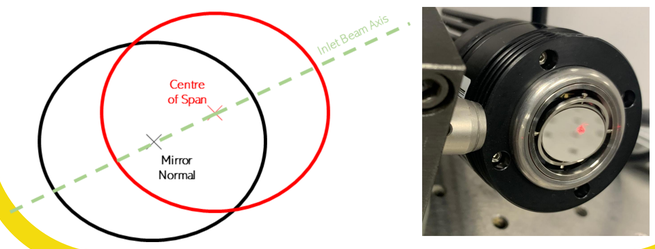
In modern manufacturing, part quality is heavily dependant on the accuracy and precision of measurements taken during the manufacturing process. Existing laser trackers enable non-invasive measurement of large parts to an accuracy of sub-10 microns; however, they are expensive, labour intensive and limited to single-point measurements.
This research presents a new measurement system that utilises a network of steerable mirrors in conjunction with a high accuracy, fixed laser to make rapid measurements in industrial settings. The new system enables an increase in measurement speed, repeatability, path manipulation around obstructions, and measurement validation. Importantly, by removing hardware from the busy factory floor, the system reduces the risks of operational interference and environmental noise, thereby improving data reliability.



Method

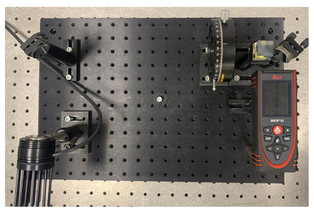
Mirror Control

- Custom python script
- Adjust mirror positions with arrow keys
- Adjustment 'step-size' control
- Saving and recalling of mirror positions for fast calibration and repeat measurements



Laser Distance Measurement Device

- Accuracy ±1000 ppm
- Optical arrangement ensures robust detection of return laser beam

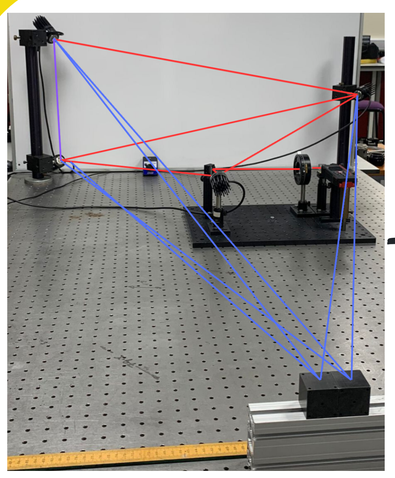


Mirror Position Configuration

- The system requires a cartesian layout of the mirrors in 3D space
- Optimisation eliminates 'error stacking' created when using analytical soltutions, therby improving positions of mirrors

Experiment

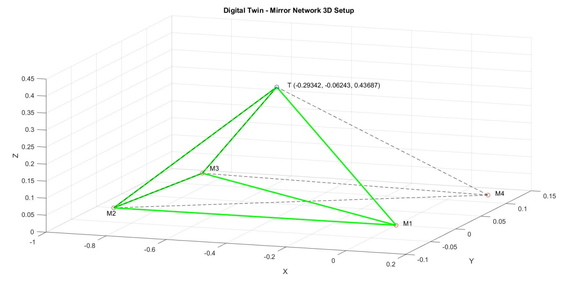
- Network calibrated by measuring distances seen in red and purple.
- Once completed, measurements are taken to a target allowing position to be calculated by trilateration
- Results can be compared against simulation (below)



Network measurement - Retroreflection required
Network measurement - Retroreflection not required
Target measurement

Digital Twin

- Trilateration to configure the 4 mirror network
- Optimisation solves 2 non-linear systems of equations defining Tetrahedron 1 (Green) and 2 (Black) to find target coordinates



Rosen Brock Function Derivation

$$|AB|^2 = (x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2$$
$$|AC|^2 = (x_A - x_C)^2 + (y_A - y_C)^2 + (z_A - z_C)^2$$
$$|AD|^2 = (x_A - x_D)^2 + (y_A - y_D)^2 + (z_A - z_D)^2$$
$$|BC|^2 = (x_B - x_C)^2 + (y_B - y_C)^2 + (z_B - z_C)^2$$
$$|CD|^2 = (x_C - x_D)^2 + (y_C - y_D)^2 + (z_C - z_D)^2$$
$$|BD|^2 = (x_B - x_D)^2 + (y_B - y_D)^2 + (z_B - z_D)^2$$

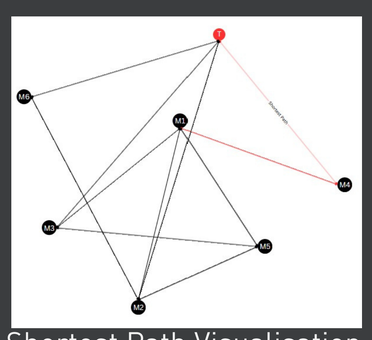
Constrain system to x, y, z axis specified in picture and rearrange to make unknowns the subject

$$b = x_C = \frac{|BC|^2 - |AB|^2 - |AC|^2}{2|AB|}$$
$$c = z_C = \sqrt{|AC|^2 + \frac{|BC|^2 - |AB|^2 - |AC|^2}{2|AB|}}$$
$$d = x_D = \frac{|AB|^2 - |AD|^2 - |BD|^2}{2|AB|}$$
$$e = z_D = \frac{(b-d)^2 + |AD|^2 - d^2 + c^2 - |CD|^2}{2c}$$
$$y_C = |AD|^2 - d^2 - e^2$$

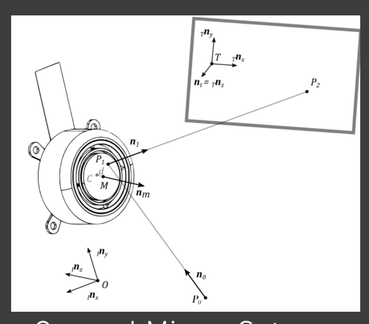
Ongoing Work

Path Finding

- Calculate suitable paths for directing laser to any given target
- A python script uses permutations to identify every possible path when including all mirrors 'n'
- Optimise for shortest path to reduce error in measurement



Shortest Path Visualisation



General Mirror Set up

Results

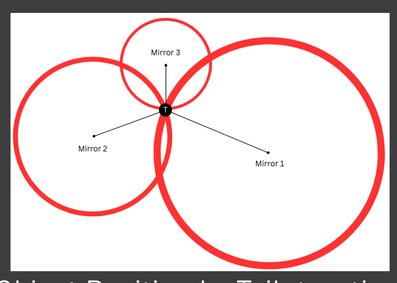
Distance between 2 Target Points

	Digital Twin	Experimental
2D - 2 Mirrors	53.4 mm	40mm
3D - 4 Mirrors	69.74mm	70mm

1. Promising results seen in 3D system
2. Use higher accuracy laser system to improve results
3. Incorporate mirrors in network to increase number of measurement paths

Quantify Errors & Uncertainties

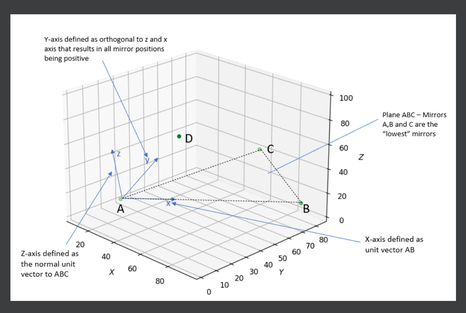
- Laser precision
- Refractive index changes in laser path
- Angle of incidence of the incoming beam
- Rotated target plane
- Off-centered incoming beam
- Distant center of rotation



Object Position by Trilateration

Automated Targeting

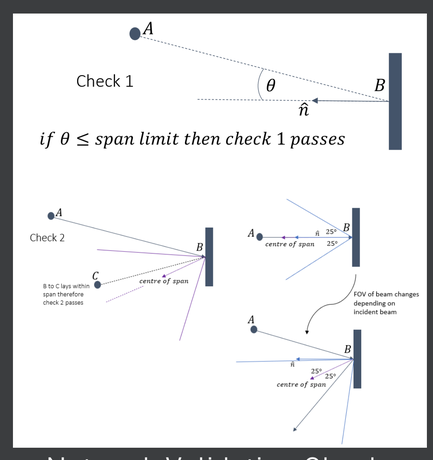
- Optimise PID control to enable multiple measurements to be taken rapidly.
- Find targets with stereovision system to enale fast measurement and tracking.



Network Simulation

System Configuration Finder

- Self determine suitable networks for any given space
- Optimize measurement capability
- Uses network checks to find feasible network



Network Validation Checks

Author LinkedIn: Prabrup Chana

Author LinkedIn: Sam Muddimer

Author LinkedIn: Tayyab Farrakh

www.mcddm.ac.uk
www.intelligent-automation.org.uk

Camera Control GUI with Camera Calibration and Body Tracking

Akilan Nambi *Masoud Sotoodeh-Bahraini* *Peter Kinnell* *Claire Guo* *Cong Sun*

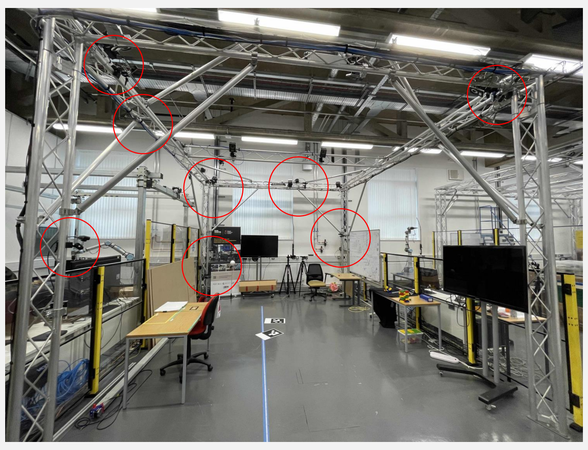
MIDLANDS CENTRE For Data-Driven Metrology

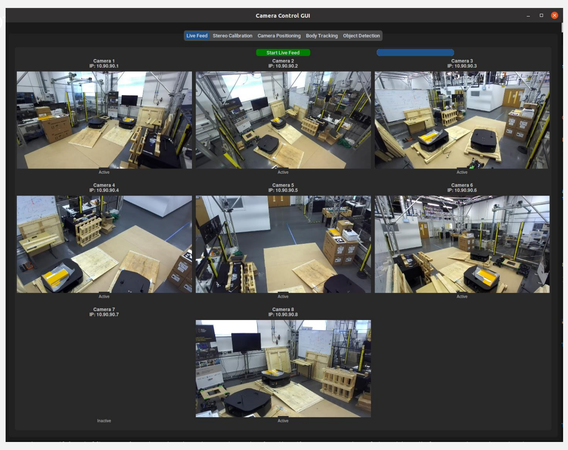
Loughborough University
Intelligent Automation Centre

The Intelligent Automation Centre has 8 ZED 2 cameras from StereoLabs placed around the lab and orientated to look at the centre. Staff using these cameras are forced to go through the tedious task of individually accessing each one to find one that suits their purpose, then running their chosen application. Creating a 'Camera Control GUI' allows staff to easily see the camera feeds as well as running applications such Body Tracking and Camera Calibration.

Setup

ZED 2 cameras are placed around the lab cell, each of which are connected to a ZED Box. By connecting all the ZED boxes via ethernet to a router, we have a local network from which we have the ability to access every camera via their IP addresses. The ZED Boxes also allow us to run applications such as Body Tracking on the cameras due to the GPU within each ZED Box.



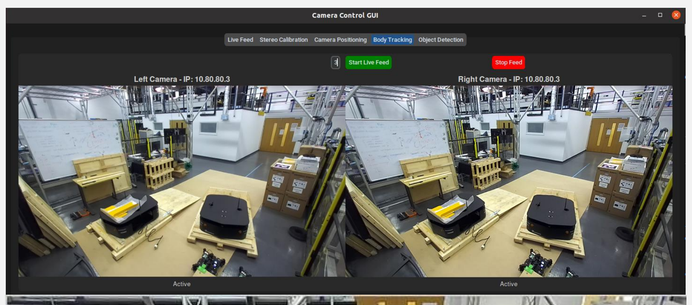


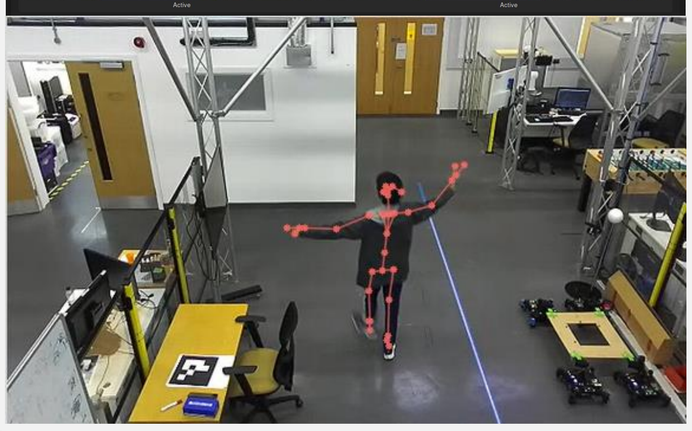
ZED Box

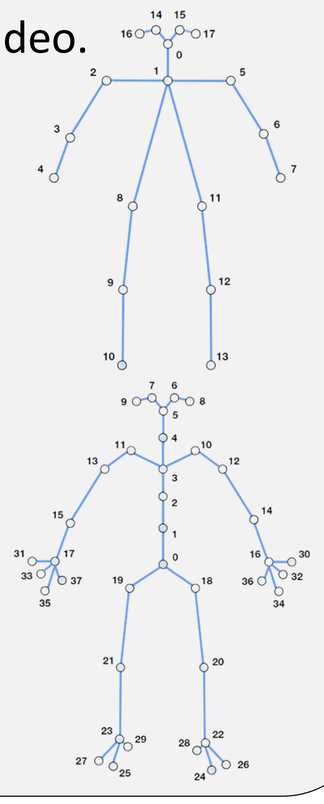
ZED 2 Stereo Camera

Body Tracking

The user can select a specific camera and bring up two live feeds. One showing the normal view and the other with body tracking running. Body tracking overlays the normal view with a skeleton image representing the joints and basic face structure of a person. The detail of the skeleton can be altered from 18 points to 38 points which affects the speed and processing power of the video.





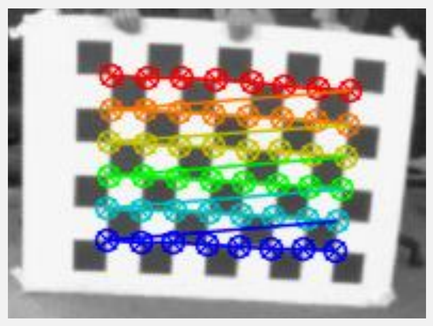


Camera Calibration

Camera Calibration is a large part of this project and involves calibrating both the intrinsic and extrinsic camera parameters, as well as the positions of the cameras relative to each other.

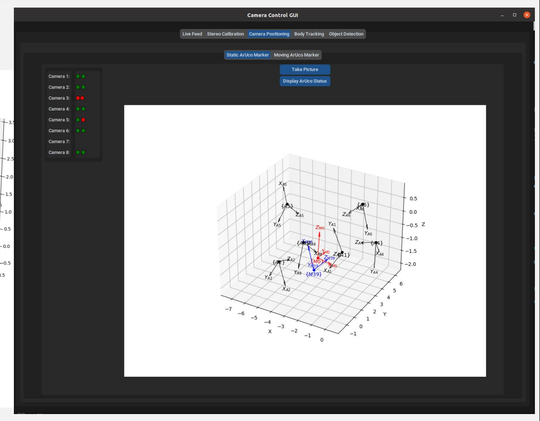
Stereo Calibration

Using OpenCV and a Checkerboard, I can calibrate each ZED 2 camera to calculate their intrinsic and extrinsic parameters.

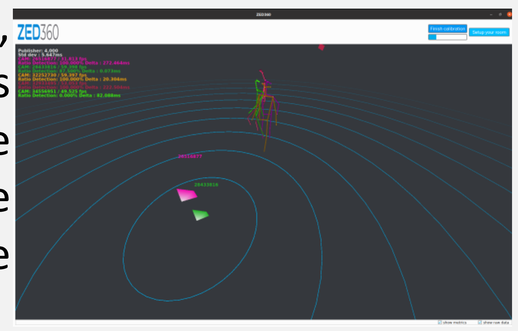


Camera Positioning Calibration

Using different calibration tools such as ArUco Markers, Checkerboards and the ZED 360 skeleton calibration, I can calculate the camera positions relative to each other within the environment.



Creating a simulated environment, I can run optimisation algorithms to minimise the errors and create accurate results for these positions. These results can be compared to the ZED 360 optimisation which uses body tracking on one person walking around the room to calibrate the cameras positions.



Conclusion

A GUI has been created that allows the lab's staff to view the cameras' live feeds and run several applications through the GUI. On start up, the GUI runs a bash file which automatically turns on the streaming script on all the cameras. Optimisation of the camera positioning is a key focus and staff are able to move and have both a static ArUco marker as well as make a dataset with a moving marker to calculate and record camera positioning. The GUI has been redesigned to be user friendly and aesthetic, while also functioning as a powerful tool for users

Akilan Nambi
LinkedIn:

p.kinnell@lboro.ac.uk
www.mcddm.ac.uk
www.intelligent-automation.org.uk

Body measurement through photogrammetry

The process of using a phone's camera to generate measurements for clothing

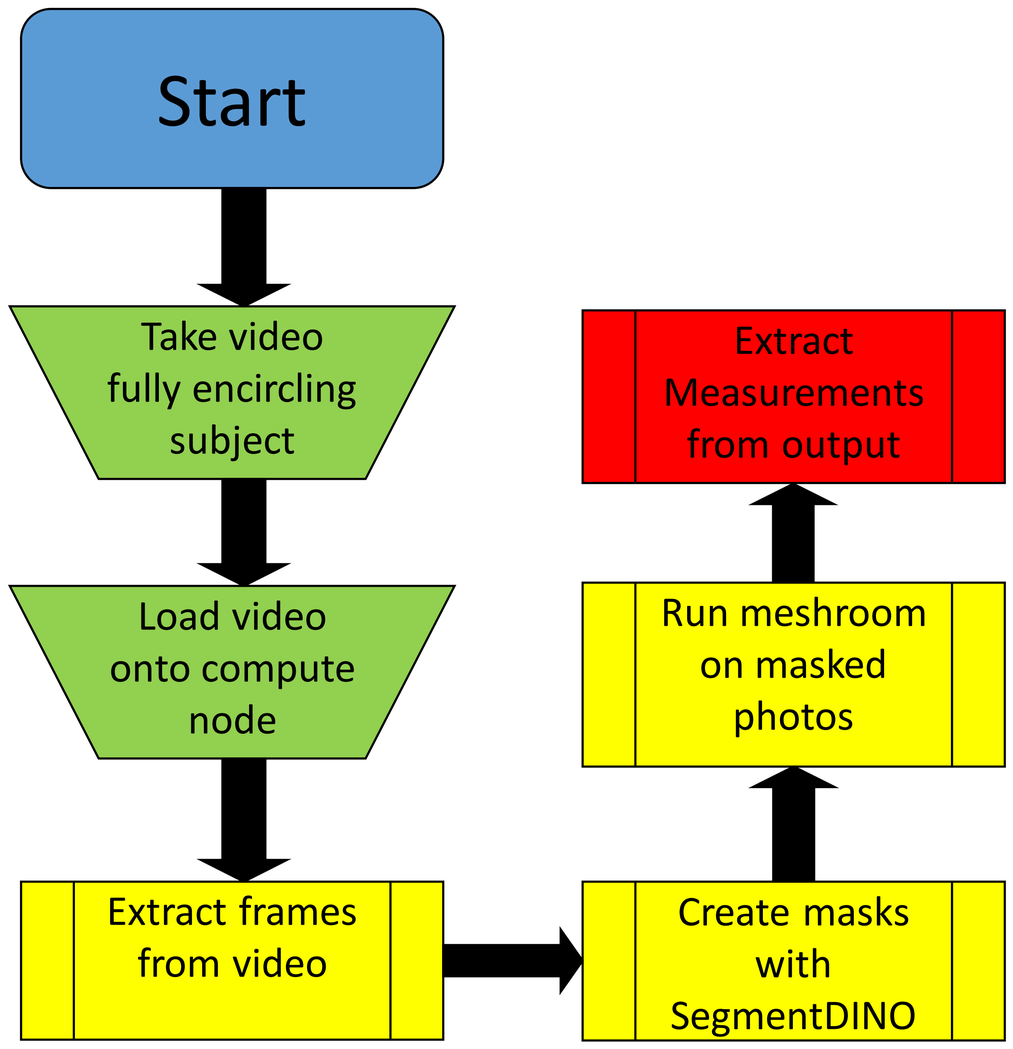
Authors: Robin Holmes, Wen Guo, Cong Sun, Peter Kinnell

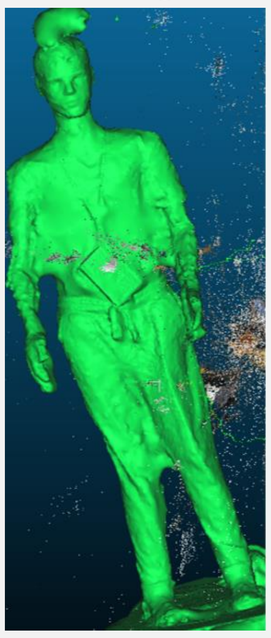
MIDLANDS CENTRE For Data-Driven Metrology

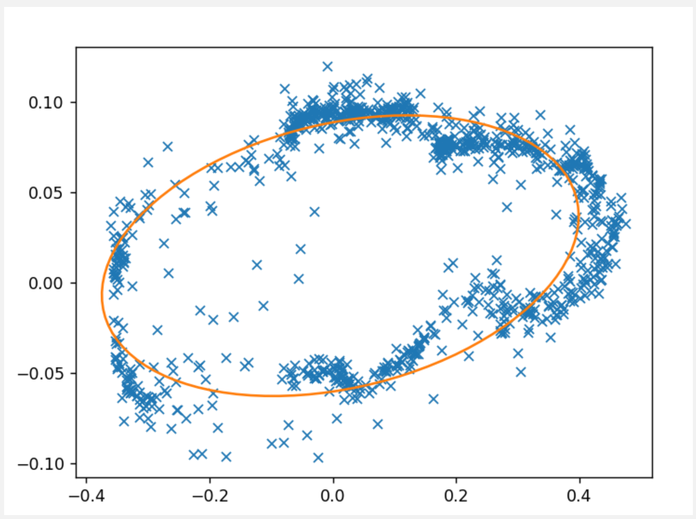
Loughborough University
Intelligent Automation Centre

How to measure someone with a smartphone:
There are two major ways to get a 3d model from a smartphone, LIDAR scanning and photogrammetry. LIDAR scanning requires a LIDAR sensor (sometimes also called a time-of-flight sensor) which is limited to certain recent I-Phone and I-Pad models as well as three now discontinued Samsung phones. Photogrammetry on the other hand requires only the camera, something all modern phones have. By taking this model and essentially slicing it into strips, measurements can be acquired from the whole body.

A quick introduction to photogrammetry:
Photography is the process of projecting a 3d object onto a 2d plane, photogrammetry is an attempt at reversing this process. It's often quite a computationally and data intensive process, requiring a fairly large dataset, including photos from all angles, to create a successful model. Photogrammetry is also sometimes called "Structure From Motion (SFM)" as by using frames from a video you get a near perfect dataset.



Colmap	Meshroom	Visual SFM	Regard 3D	Kiri Engine	Polycam
					
The result here is extremely noisy and lacks detail, the whole scene having just 102k points. Looking at the arms in particular, some areas are entirely impossible to measure.	This model is objectively the best for the application, note the clearly defined limbs without holes and the relatively high 164k points. Some areas are still somewhat sparse (E.G. the shirt)	Visual SFM failed to produce a viable model, instead giving less than 2 thousand data points. Some recognizable features such as the lanyard are still visible, but nothing that can be used.	Insanely noisy, results that almost make Colmap look good. 527k points means it's not lacking in detail, but it also took the longest to run by far.	While the model may look like one of the best at first glance, this is actually one of the least useful, note the fusing between the arms and legs making measuring impossible. Also, there's a strange object ejecting itself from the top of the model that isn't on the real object.	This model is not actually a point cloud, this is a textured model making it much harder to pull usable data from. Also note the fusing between the legs and the fringing on the arms - the model is basically useless.



Measurements from model:
By slicing the model on the XY plane we can generate a series of rough ovals of data points, and these data points can then have an actual oval fitted to them. The code that generates this best fit oval gives the semi major and semi minor radii which is everything we need to find the perimeter.

By inputting these results into the slicer, we have seen results within 5% of the expected value (after scaling by a predetermined known factor).

Final steps:
Finding the scaling factor of the model is currently a challenging and very involved process and getting this wrong can cause results to be entirely useless.

Using a checkerboard to automatically scale would probably be the fastest and easiest solution, but more testing is needed.

Robin Holmes
LinkedIn:

Automatic Assembly System based on an Object Detection and Localization Algorithm

Ahmad Abdin, Cong Sun, Claire Guo, Masoud Sotoodeh-Bahraini, Peter Kinnell

MIDLANDS CENTRE For Data-Driven Metrology

Loughborough University
Intelligent Automation Centre

Introduction

In automatic assembly systems, object detection is a necessity for the process to be completed, hence the reason behind the use of point cloud stitching and PartFinder.

Why point cloud stitching?
- Adaptable
- Accurate
- Can interface using python
- Easily evaluated

Why PartFinder?
- Flexibility.
- Fast process.
- Can interface using python
- Able to detect a variety of objects

Point Cloud Stitching

Point cloud stitching was accomplished via two methods:

Method #1: Manually by using a point cloud processing software (CloudCompare)

Method #2: Through a python code which consists of 4 sections (input, pose graph, optimization, output).

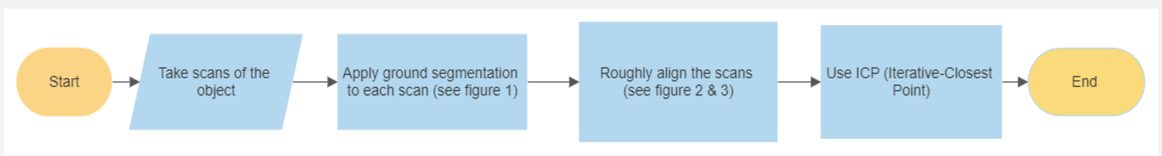


Figure 1. Workflow of point cloud stitching and multiway registration

Figure 2. Scans after the ground segmentation process.

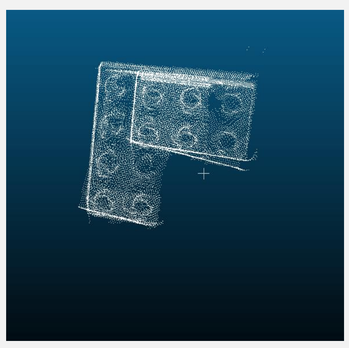


Figure 3. Rough alignment of point clouds through CloudCompare functions (manual).

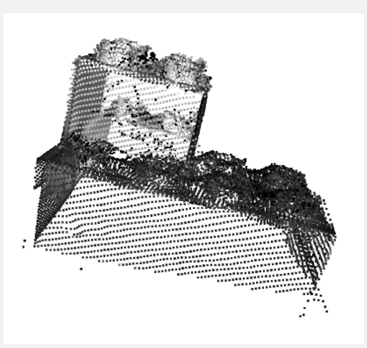
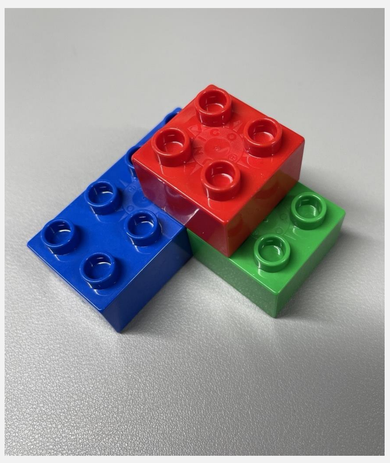
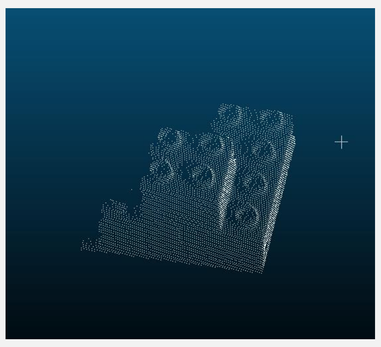
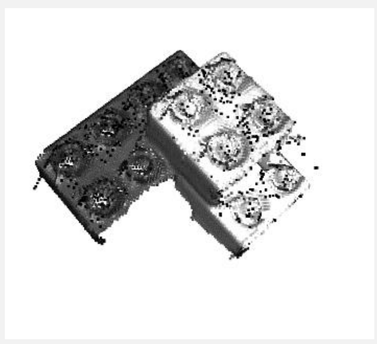
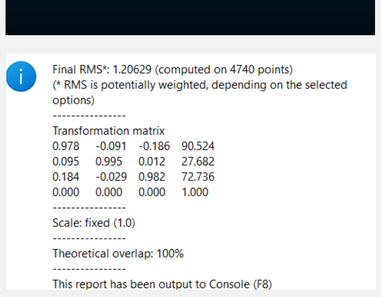
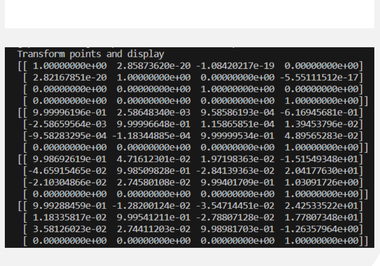


Figure 4. Rough alignment of point clouds through python scripted pose graph estimation.

Results:

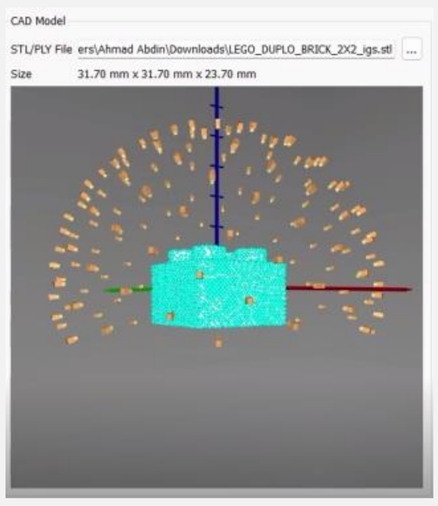
The images below include the real-world LEGO brick structure, the final point cloud images and the transformational matrices of the scans after the Iterative-Closest-Point (ICP) is accomplished for both methods.

Real Image	Method #1	Method #2
		
		

PartFinder

The Ensenso PartFinder allows users to detect objects by uploading a CAD file of the item to their NXView software. This module was used, placing an Ensenso N30 camera on a tripod in order to angle it on top of an area that included a number of 2×2 and 2×4 LEGO bricks placed randomly with different rotations. By doing this we could investigate the software's efficiency in object detection.

Step #1: Model Generation

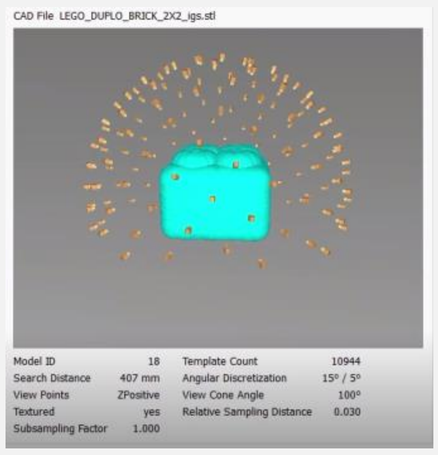


This section allows the user to modify how the camera views the object itself. This is done through working on many important parameters such as:

Angular discretization:
simplification of the possible orientations or angles at which an object can be detected.

Camera viewpoints:
controlling the sides at which the camera would be able to identify.

Step #2: Finding the Model



The next stage of the process includes the parameters of how the software will help the camera detect the object, based on the object's characteristics.

Distance ratio:
changes the detection rate when searching over a distance.

Coverage threshold:
selects the threshold where the points are considered to be on the surface.

Hypothesis clustering:
groups educated guesses through object characteristics.

Results:

The results provide the user with the object's 6 degrees of freedom along with the score (confidence of the software that the object represents the CAD model), surface coverage, texture score, and coverage threshold.

	Score	Surface Coverage	Coverage Threshold	Texture Score	X	Y	Z	Rot X	Rot Y	Rot Z
1	0.75	0.78	5.0	0.72	130.06	78.29	375.15	59.61°	-70.58°	-27.29°
2	0.69	0.84	5.0	0.59	53.54	51.13	388.52	148.44°	30.70°	49.35°
3	0.69	0.74	5.0	0.65	158.16	-7.66	417.51	172.36°	4.40°	162.14°
4	0.67	0.71	5.0	0.64	-28.80	-36.84	408.65	175.92°	2.15°	-23.80°
5	0.67	0.73	5.0	0.62	62.84	-67.95	406.67	25.99°	-169.93°	117.64°

Conclusion & Future work

After working with point cloud stitching and PartFinder, it is deduced that point cloud stitching does have an advantage due to its accuracy. However, whilst PartFinder provides faster results and is more user-friendly, factors like the angle of the camera and object, and the colour and size of the object, affect the results. To further improve the findings and increase the understanding of the methods, the next stage is:

- Integrate hand-eye calibration to the point cloud stitching API code
- Increase the number and variety of objects tested with both methods
- Incorporate both methods into robots with the aim of allowing it to scan and detect objects without the need of a person to direct it

Ahmad Abdin
LinkedIn:

www.mcddm.ac.uk
www.intelligent-automation.org.uk

Object Detection Using an Automated Blender Simulation

MIDLANDS CENTRE For Data-Driven Metrology

Loughborough University
Intelligent Automation Centre

Authors: Uche Okwese, Peter Kinnell, Wen Claire Guo and Cong Sun

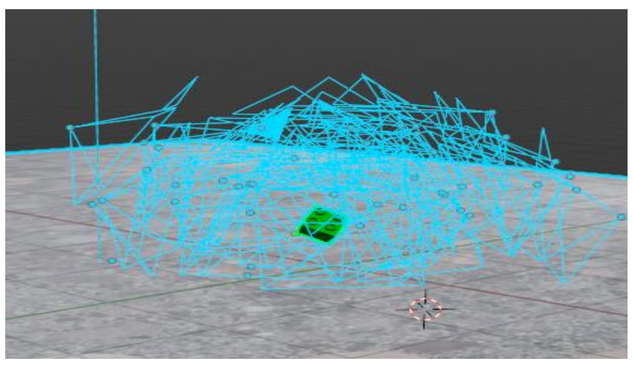
ABSTRACT

-Generating a 3D Model library data set in Blender using Python Scripting to replicate work environment to be used by a collection of Neural Networks

-Determining optimal rendering resolution, image sample size and simulated environment correlation to work environment needed for an effective object recognition neural network

1) METHODOLOGY

Camera Simulation:



- Cameras generated using a Blender Python script function

- Camera generation location is randomized within a set maximum and minimum distance from LEGO Brick

Computer Specifications:

Processor: AMD Ryzen Threadripper 2950x 16-Core Processor 3.50 GHz

RAM: 64 GB

GPU: NVIDIA GeForce GTX 1660 Ti

Rendering Time (In Seconds):

Per Image: **[Resolution]:**

0.8s (Without Background) & 1.5s (With Background) **[300 X 300]**

5.8s (Without) & 9s (With) **[640 X 640]**

7.2s (Without) & 17s(With) **[1080 X 1080]**

Annotation Directory:

- Each render is split into 2 directories (Image & Annotations), annotations contain image data for the computer to interpret what a human visualizes.

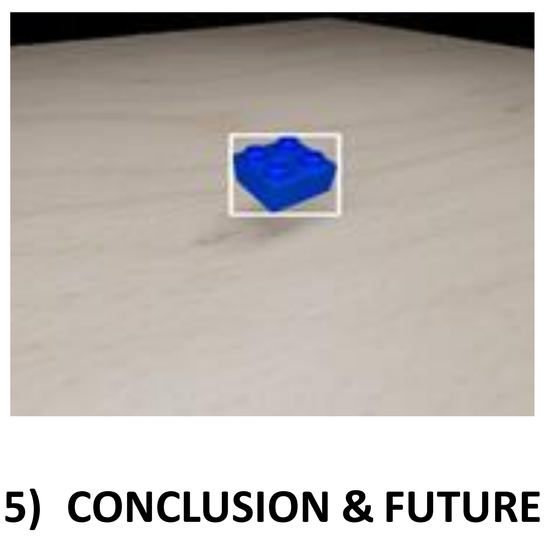
- Directories are defined in the Python Script and image data can be returned by calling the variables and functions in which they are stored.

BG1 + Objectannotation_Pose_4_Cam_31.txt - Notepad

File Edit Format View Help

```
Camera Location: <Vector (-0.2541, -0.2731, 0.9397)>
Camera Rotation: <Euler (x=0.1575, y=-0.0000, z=-0.9334), order='XYZ'>
Object Location: <Vector (-0.1765, -0.2157, 0.3315)>
Object Rotation: <Euler (x=1.3600, y=0.8000, z=1.2000), order='XYZ'>
Class: LEGO_DUPLO_BRICK_2X2_igs
Camera Name: Camera.31
```

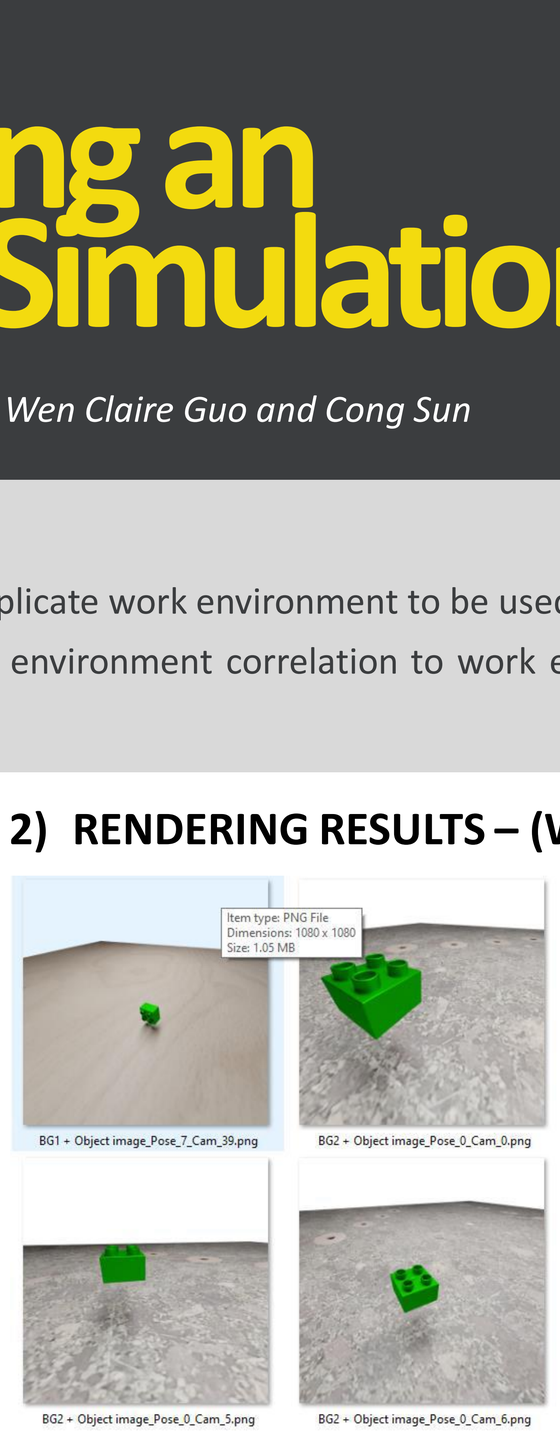
AI Training:



- A bounding box is an imaginary rectangle that illustrates the approximate location and size of an object

- This bounding box is automatically created using an external Python program (or embedded) to return the output bounding box images from the dataset

2) RENDERING RESULTS – (WITH BACKGROUNDS)

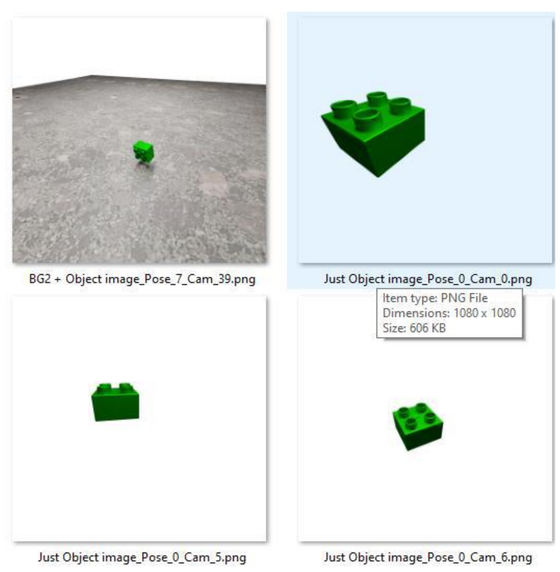


- LEGO CAD Model imported into simulated environment

- Rendered images illustrating the different perspectives of the 2 X 2 LEGO brick

- 2 backgrounds (Wood & Metal surface) added to simulation by importing pictures from the lab to 3D Plan

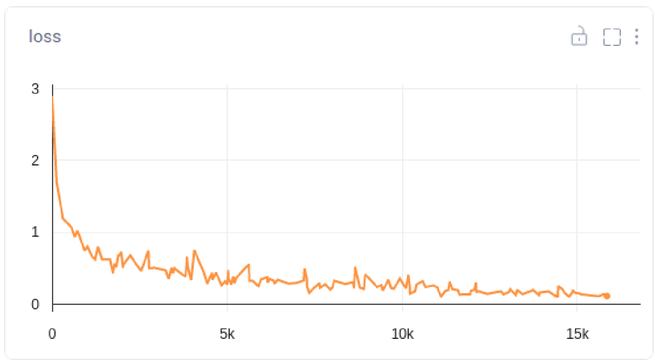
3) RENDERING RESULTS – (NO BACKGROUND)



- Rendered images show appearance of LEGO Brick without a background or background reflection

- No background samples obtained for "Background Subtraction;" a method for localizing an object regardless of its surroundings

4) AI TRAINING RESULTS



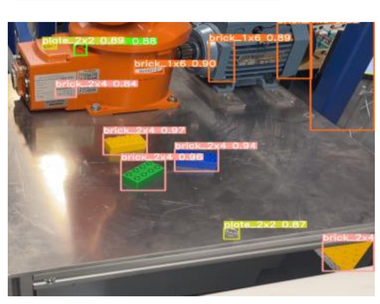
- y-axis represents value of loss function (Lower loss values indicate that the model's predictions are closer to the ground truth labels).

- x-axis is the number of trials.

YOLOv5 (COCO) MODEL:

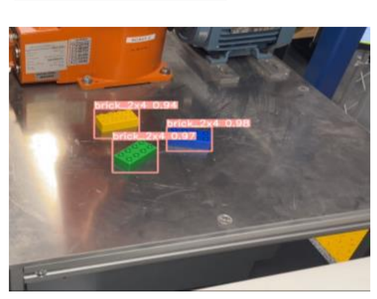
General Opensource object detection library fails to detect our LEGO

GITHUB MODEL:



Pre-existing GitHub model used to build our neural network misinterprets similar objects to LEGO

OUR MODEL:



Using our own custom model for the same Neural Network, the AI can now accurately detect LEGO bricks

5) CONCLUSION & FUTURE WORK:

Monotonous and time-consuming tasks, like bounding box creation, and image and annotation generation, become fully automated.

Training with our custom data makes the model work more efficiently as the simulated environment can be specialised to depict conditions and interactions for the target environment.

In the future this model will be used to combine simulation and a small batch of real data to yield more accurate and precise results for object detection.

Uche Okwese
LinkedIn: