1.3 Objectives

ID	Objective	Achieved S1	Achieved S1
01	Input a picture from any camera source into system	-	-
02	UI built within the unity engine	-	-
03	Single object recognition from picture	-	-
04	Isolate the desired object from a complex picture	-	-
05	Recognise the depth of object in relation to the camera	-	-
06	Recognise the global rotation of desired object	-	-
07	Output position, rotation, and depth of object in relation to camera	-	-
08	Use a JSON structure for data transfer between components	-	-
09	Build API to allow the application to offload processing of image	-	-
10	Use Azure/AWS VM with a public IP for server hosting	-	-
11	Offload ML model training to the cloud	-	-
12	Generate 3D scene from JSON structure output by API	-	-
13	Create a selection of models to substitute for objects in 3D scene	-	-
14	Organise system into single application pipeline	-	-

2. Literature Review

2.1 Part one

To complete this project in a way which furthers the AI community in NI and push the boundaries of automated image processing, the author of the project has conducted preliminary research into several topics including R-CNN and pose detection.

2.1.1 RCNN

Before getting into the reasoning of using an R-CNN within this pipeline, a brief overview of the research that led the author to the conclusion that an R-CNN would be useful. The R-CNN (Regions with CNN features) structure was proposed as an effort to bridge the gap that sat between object classification and detection [2]. The basic R-CNN structure split the process into 3 stages or modules:

- Extract region proposals
- Compute CNN features
- Classify regions

Using an RCNN we essentially generate *r* number of regions of varying size across the picture input. These regions are category-independent. There are many different ways to generate these regions as objectiveness[3], multiscale combinatorial grouping [4], and object proposals [5].

After the initial paper outlining the use of RoIs and CNNs together, the next major upgrade to the R-CNN format came in the form of Fast R-CNN (the background of Fast R-CNN is required for the comparisons in figure 9). This methodology proposed higher efficiency of the R-CNN method by implementing feature sharing and multi-task loss during training among other advancements [6]. Researchers developing this method had shown up to 18.3x increased training speed than R-CNN and SPPnet [7], another computer vision method [6]. Unfortunately, Fast R-CNN uses object proposal as a method for determining RoIs. Due to the nature of this project, the author decided it best to look for a

different method for determining the RoI rather than manually specifying which object to look for each time the application was executed.

Another route to follow when aiming to fill the object recognition, classification, and isolation was to use the FCN (Fully convolutional network) methodology [8]. Although this is not an R-CNN, it could in theory fill the same section for image recognition and classification. The author's reasoning against using FCN was due to the work required to adapt the methodology to the current program. Although in terms of standalone semantic segmentation FCN could be a great contender, the product theorised by the author would require the instancing of each class. The FCN methodology alone shows to work poorly for this [9] and would require further work to allow this. In the future, once the pipeline is built, there may be a case for transitioning the component into an FCN if experiments show a speed increase.

The theorised product would require instance segmentation as previously mentioned. Recently work was published detailing mask R-CNN. This is a variation of an R-CNN with a focus on instance segmentation [9]. This is an extension of the previously mentioned Fast R-CNN and as such is fast to implement and train. This relives time pressures on the Author to implement this component.

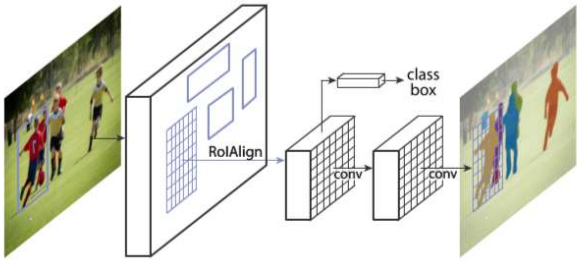

Figure 1 - Mask R-CNN framework [9]

In terms of the object recognition component of the product, the system would require the recognition of an object within the input photograph. Once an object is recognised, we would then require the separation of this object from the remainder of the picture on a pixel-by-pixel basis. This combines both semantic segmentation and instance segmentation. The only CNN purpose-built to achieve this task in an efficient manner is the mask R-CNN. Figure 2 below shows mask R-CNN currently outperforms all previously mentioned models.

	Backbone	Bounding Box AP
Faster R-CNN+++	ResNet-101-C4	55.7
Faster R-CNN w FPN	ResNet-101-FPN	59.1
Faster R-CNN w G-RMI	Inception-ResNet-v2	55.5
Faster R-CNN w TDM	Inception-ResNet-v2-TDM	57.7
Faster R-CNN, RoIAlign	ResNet-101-FPN	59.6
Mask R-CNN	ResNet-101-FPN	60.3
Mask R-CNN	ResNeXt-101-FPN	62.3

Figure 2 - Comparison between R-CNN models in regards to bounding box accuracy percentile on single models [9]

2.1.2 Pose Detection

In order for the final product to work as expected, we need to not only figure out what the object is with regard to classification (as discussed above) but we also need 3 pieces of contextual information: position, rotation, and depth. In order to achieve this, a pose detection algorithm will need to be put in place. The main issue with pose research is that many of the papers found by the author focused on algorithms to determine human positions where the product would require object poses.

Modern pose detection processes were first presented in 2005 as pictorial structures for object recognition [10], which is still considered a starting point when developing pose recognition systems [11]. The pictorial structures specify a model that will distinguish between instances of an object based on

training models. Again, this paper uses human pose to test and validate the algorithms but it is worth a mention here as although the author does not require human pose, the graphical tree based approach given by the paper may be required knowledge when developing the object pose model.

From the pictorial structures, we can follow pose detection through hierarchical models [12], non-tree models [13], [14], to convolutional models [11] which are commonplace today. The author wishes to use a pose model which allows us to define which way an object sits in the image. For instance: what way the handle of a cup faces as it sits on a table. The convolutional pose detection model would be able to achieve this however there are several downsides the author had found.

The first of these downsides is the keypoint locations. Specifically, the denotation and training of these locations. In the previously mentioned papers, the methods outlined require the manual specification of each key point for each training image. The author believed that there is a better way to undertake training using other software which will automatically find the key points for itself. This would allow for faster training of models and therefore faster addition of more objects to the system. Shortening training time later in development was seen to be a worthy decision even considering the risk of the required upfront development time [Section 4].

The second downside of these models is the output of point depth in relation to the camera. One major issue when developing a system such as the product proposed, is that depth is notoriously hard to gather from a single point perspective for several reasons. Depth cameras often make use of either infrared or stereoscopic cameras to define depth as each of these technologies allows us to make easy distinctions between locations of objects in relation to the camera. The author however, has specified that the product uses single monoscopic images to create the scene which complicates the process. Rather than taking the depth data direct from the camera, we need to instead analyse the context of the object [15].

In relation to the high-level system architecture the author has proposed in section 3.5 of this report, we need to pass the pose components just the masked image theoretically. This proposal may require changing down the line as with a masked object image we remove the global context. We get around this by using recent research into latent key point analysis. This research proves that using a trained model, we can output the depth of key points within the object image [16].

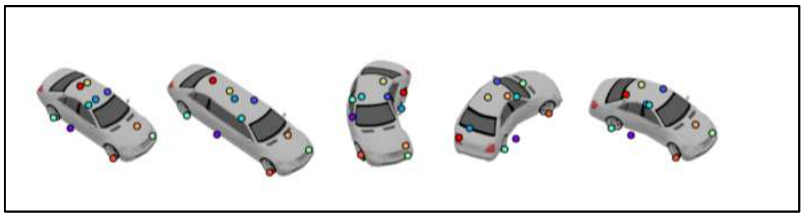

Figure 3 - Keypoints on a deformed car [16]

The main thing that drew the author to this model for depth and pose is the automatic generation of key points. This means that as the system trains, it will define its own key points to use meaning it will always use the most efficient points. The system also analyses and outputs the depth of each key point which can be theoretically used to position the object in relation to the camera. Using this system will reduce the training required for adding objects to the system however does come with a large upfront development cost due to it being experimental technology.

One major risk to using this technology for the process is that it has not been wholly proven against real world scenarios. Throughout the paper the authors have used 3D models to train, test, and validate the model. This may require the author of this project to edit the algorithms used or generate new processes for pose and depth of objects.

2.2 Part two

There are some similar solutions to the problem outlined. In terms of the transferring of a real life space into a 3D scene that can be used in 1:1 scale would be photogrammetry. Photogrammetry is defined by [17] as:

> *"the science of obtaining reliable information about the properties of surfaces and objects without physical contact with the objects, and of measuring and interpreting this information."*

When used in the real world, photogrammetry and LIDAR scanning allows engineers and developers to effectively and realistically scan a given real life area in moments. One of the global leaders in photogrammetry products [Matterport] states that their scanners can produce a realistic scan of a $2000ft^2$ area in under 26 minutes [18]. This does not include processing time nor setup time, which increases the total time significantly. These numbers should be taken as anecdotal however, as dealing in scanning the time of the scan depends wildly based on what you need from the location and how many scans you require.

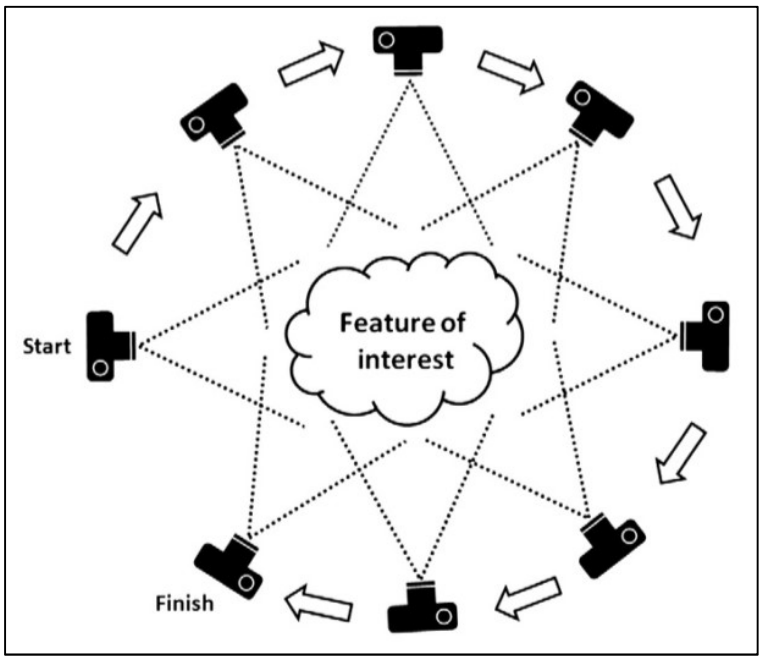

Figure 4 - SfM photogrammetry [19]

The Leica system for LIDAR scanning of an area states a scan time of 3 minutes per single scan [20]. The author has personal knowledge of using this system and understands that a $200ft^2$ area from start to usable 3D image can take up to half a day. This scan time – although relatively fast for the actions – is still too long for quick developments and each comes with issues in changing the 3D scan to suit a developer's requirements.

A big difference in the proposed system would be the use of custom models in substitution of real-world textured models. Where a 3D area would be generated based on the textures and measurements from the real world, the proposed system removes the real-world textures. Our system uses the measurements and locations of objects and theoretically, would generate the area faster because of this.

The systems do however have different applications. A geospatial scan such as a scan from LIDAR or generated using photogrammetry have their uses is realistic representations of areas. This is a use case that is key to preserving history and spaces for the future. The proposed system of this report would be used to generate quick 3D spaces using custom models based off a single image. There may be a use case down the line where, using feature recognition or location mapping we could scan and generate full 3D