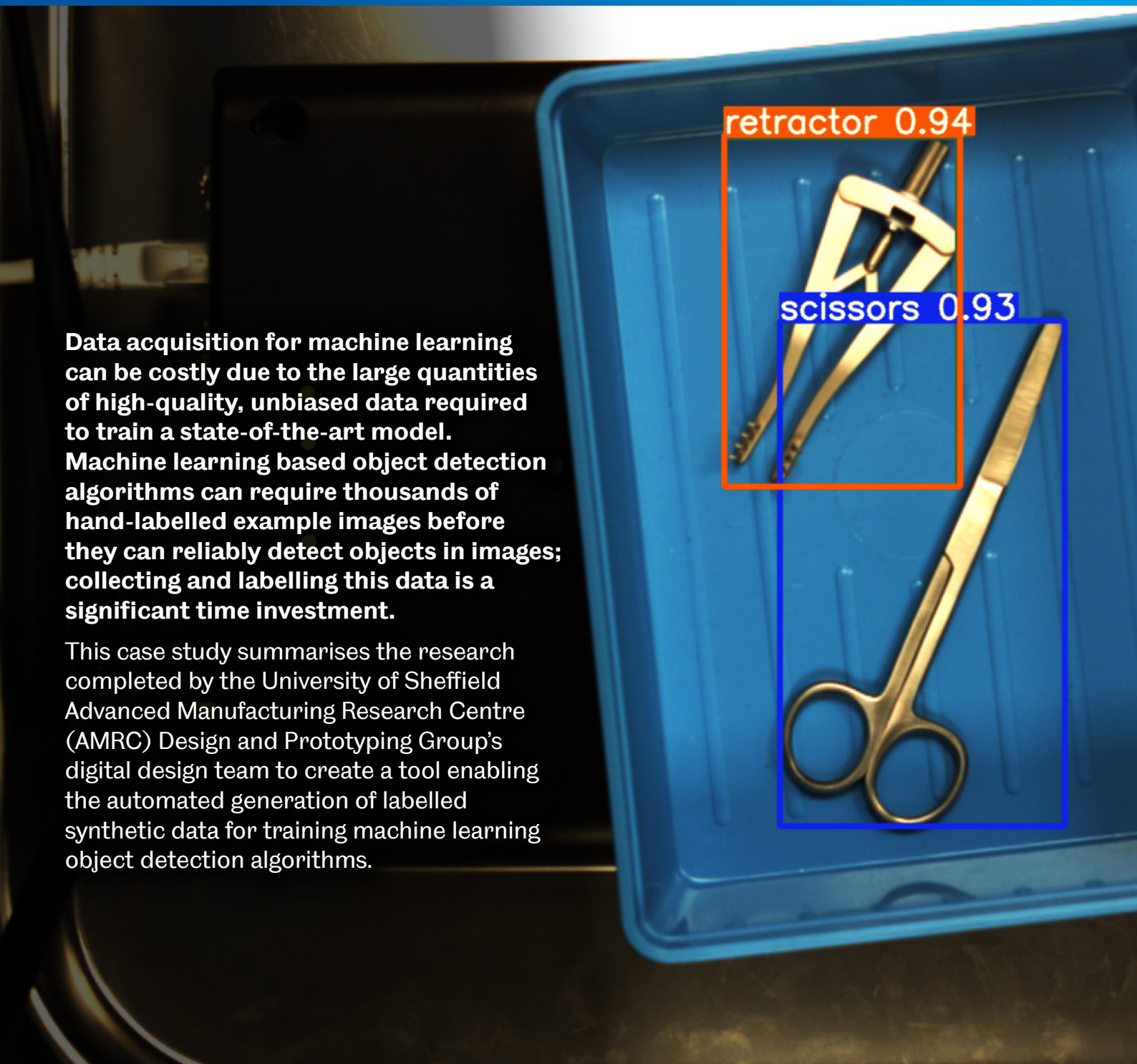
The University Of Sheffield. AMRC Advanced Manufacturing Research Centre

Design and Prototyping Group
Case Study

Generating synthetic data to train accurate AI object detection



Data acquisition for machine learning can be costly due to the large quantities of high-quality, unbiased data required to train a state-of-the-art model. Machine learning based object detection algorithms can require thousands of hand-labelled example images before they can reliably detect objects in images; collecting and labelling this data is a significant time investment.

This case study summarises the research completed by the University of Sheffield Advanced Manufacturing Research Centre (AMRC) Design and Prototyping Group's digital design team to create a tool enabling the automated generation of labelled synthetic data for training machine learning object detection algorithms.

CATAPULT High Value Manufacturing

Credit: Faungg's photos. Flickr. (Changes made to image, right). www.flickr.com/photos/44534236@N00/

Figure 1: An example of semantic segmentation.

The problem

The training of an accurate artificial intelligence (AI) object detection and classification model requires an extensive dataset consisting of hundreds of labelled image examples in various environments and contexts. One of the most famous public datasets, COCO (Common Objects in Context) contains over 200,000 labelled images containing over 1.5 million objects across 80 different classes. Training a custom dataset is normally done via transfer learning of a model already trained on a dataset such as COCO and thus requires far fewer examples to fine tune to the new classes. Many example images are still required to accurately differentiate and detect new objects added to the dataset.

Creating and labelling hundreds of images per class is a very time consuming task, especially if the end use case or implementation has a large number of unique objects to detect. The images must also be hand labelled with either bounding boxes or semantic pixel labels by a person before the data can be used for training a model at the expense of more development time and cost.

The AMRC Design and Prototyping Group's digital design team have developed a proof-of-concept tool to address this problem. The tool is capable of generating the data required for training a model using only CAD data and some background images to put the object in context of where it will be detected in the final application of the machine vision system.

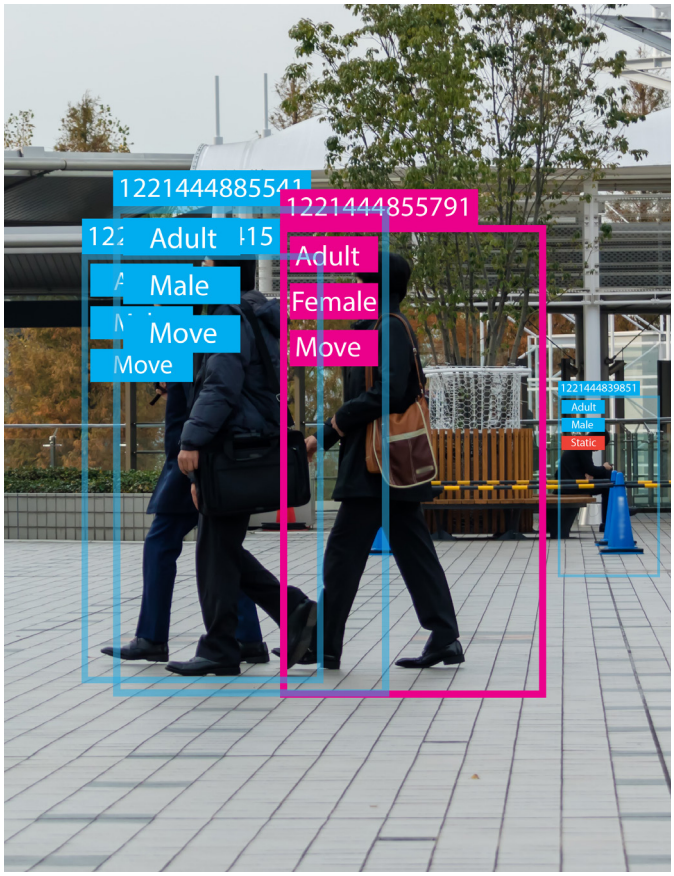


Figure 2: An example of AI image detection and classification on people.

What makes a good dataset?

A good dataset not only contains a large number of images, but those images also have to be representative of objects to be detected in the context you want to detect them in. For example a training set of scissors only on a red background may not generalise well to scissors on other coloured backgrounds. This is also true of lighting conditions, object orientation, occlusion, and size; however, some of these issues can be addressed in training via augmentation. The dataset should also have an even spread of examples for each class, if the network is 90 per cent one class then it could achieve 90 per cent accuracy by always predicting that class. Skewed class frequency can lead to low confidence values on the underrepresented classes and confusing items for the more common classes.

The subject of the image does not necessarily need to be clear and obviously the focus of the image. In fact, it is quite the opposite for some object detection applications. The subject of the detection should appear as it would in an end use case scenario. If the object you are attempting to detect is often half obscured then the training examples should also be half obscured. So that an accurate and unique feature map of the object can be learned during training a sample of the full range of possible characteristics should be present in the training set. This is true of all characteristics, such as lighting conditions, image contrast, object posing and subject size.

It is important that if an object can be posed, then all possible poses of the object are captured in the training data. For example with a pair of surgical scissors, if none of the images in the training set captured the scissors fully opened, the model cannot be expected to recognise the opened pair as the same object. This can also be extended to object orientation so the object can be recognised from all angles, objects that come in different shapes or colours, and objects of varying size.

Lighting conditions in particular can prove problematic due to reflections, lens flares or coloured lighting disrupting the detection. If ideal lighting conditions are used when capturing the training data the model will have trouble detecting the objects in other environments. Effort should be made to capture images of the object with various levels of contrast and light sources of varying brightness, although this can be somewhat accounted for with digitally altering the contrast later.

Image quality and resolution are less of a problem as long as the object is large enough to be resolved. In most image based AI detectors the image is downsampled to a fixed smaller resolution to maximise processing speed and efficiency. This also combats problems of low detail on objects as the model will learn other characteristics.

The quality of the labelling is a very important aspect of a good data set. Most object detection algorithms use a bounding box detection system where a rectangular box is drawn around the object and it is given a class label. Other systems, such as semantic segmentation, give a label to each pixel in the image. The process of labelling a dataset involves manually drawing the bounding box around each class and giving that box a class label, eg. “car”. There are numerous tools available for image labelling and differing levels of detail. The fastest way is a rectangular bounding box dragged around each class such as the images in Figure 2, however more complex polygons or exact pixel level labeling can be done for high-precision applications such as biomedical imaging [1].

Augmenting a dataset is done to improve its quality and increase model performance. Augmenting a dataset means perturbing the image in some way, this is done as it is passed into the network, with the goal of making the model more robust. An example of augmentation is adjusting the contrast on an image during training to make the network more capable of dealing with various lighting conditions. Other forms of augmentation include cropping, skewing, rotating, or adding noise to the image. Some of these changes such as skewing or rotating also require the image labelling to be adjusted in the same way to match the newly augmented image.

Ronneberger et al 2015 pg. 5 [1] states why image augmentation was needed for their specific needs:

“Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available”

Augmentation is invaluable for bespoke applications where large datasets are not available.

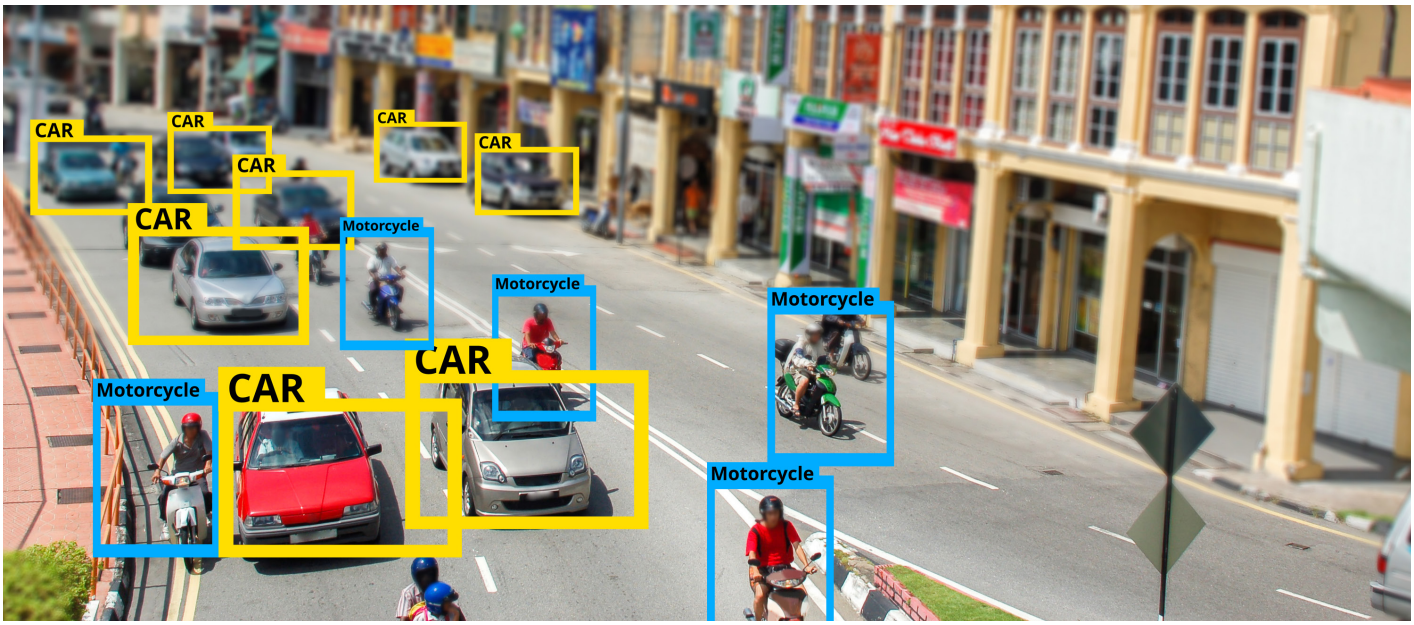


Figure 3: An example of object detection and classification on traffic. This data can be useful for smart adaptive traffic systems.

The standard process

The traditional process of dataset generation typically begins with capturing the photographs of the objects that will be the subject of the vision system. As previously mentioned, the objects would ideally be in similar context to what they will be detected in, and in a variety of poses, positions, and lighting conditions, which can be time consuming for a large number of images. In some cases video can be taken and each frame of the video can then be extracted for labelling which can speed up the acquisition of images. This still takes up a substantial amount of time post-processing and may not be ideal as it is unlikely to accommodate changes in lighting conditions and similar environmental effects.

Once the images are captured they must be labelled before they can be used for training, this is normally completed manually. Depending on the models that will be used, the labelling will be slightly different. A simple classifier model will only require a label for the entire image, such as a classifier that determines if the image is that of a dog or a cat. Common object detection networks will place bounding boxes and classes on individual objects within the image, therefore they require objects to be labelled using bounding boxes which, as explained above, involves manually drawing the box and labelling each object in each photo captured.

One further step up from this would be labelling for semantic segmentation models. These models give a label to each pixel in the image and so require precise class labels to be painted onto the image. An example of semantic segmentation labelling is shown in Figure 1. Pixel labelling can also be used as rectangular bounding box labelling.

Augmentation is the final step once a dataset has been labelled. This augmentation can be done explicitly before training to increase the size of the dataset or it can be done during runtime, automatically modifying the image as it is used as a training example. The former method allows the data to be inspected before training and requires the augmented images to be stored but the latter ensures each example is unique across many training runs and does not require any additional storage space. The examples being generated at runtime is a double-edged sword, while it means the examples are always new, which is useful for additional training or fine tuning, it also means the dataset is not available for debugging or troubleshooting later.

Review of similar methods

The data acquisition problem has been addressed both commercially and in academia previously with various machine vision goals in mind. Planche et al. 2017 [2] demonstrated a synthetic dataset generation pipeline specifically for estimating depth in images using CAD data. Their method focused on recreating the characteristics and artefacts from real world depth scans in their synthetic data. They demonstrated that their method was more capable, in some cases, than the state-of-the-art at the time for training these models.

Wong et al. 2019 [3] also created a data synthesis pipeline with the goal of training a machine vision system to recognise supermarket products in a warehouse setting. Their framework used photogrammetry to create the 3D data which was then used to create the synthetic images for training. They started with photographs of real objects, generated a 3D representation using these photographs, and then synthesised the 2D images in the datasets from these. Similarly to Planche et al 2017 [2], they showed that generating this data saved time and effort compared to manually capturing and labelling a traditional dataset.

There are certain applications where synthetic data is much more desirable due to the inability or difficulty of acquiring human labelled data. Mayer et al. 2018 [4] shows just an example with stereo video data for optical flow or stereo disparity.

Mayer et al 2018 pg. 1 [4]:

"The dominant data acquisition method in visual recognition is based on web data and manual annotation. Yet, for many computer vision problems, such as stereo or optical flow estimation, this approach is not feasible because humans cannot manually enter a pixel accurate flow field"

Hence synthetic data being the only option for some applications and perhaps opening the doors to many new applications.

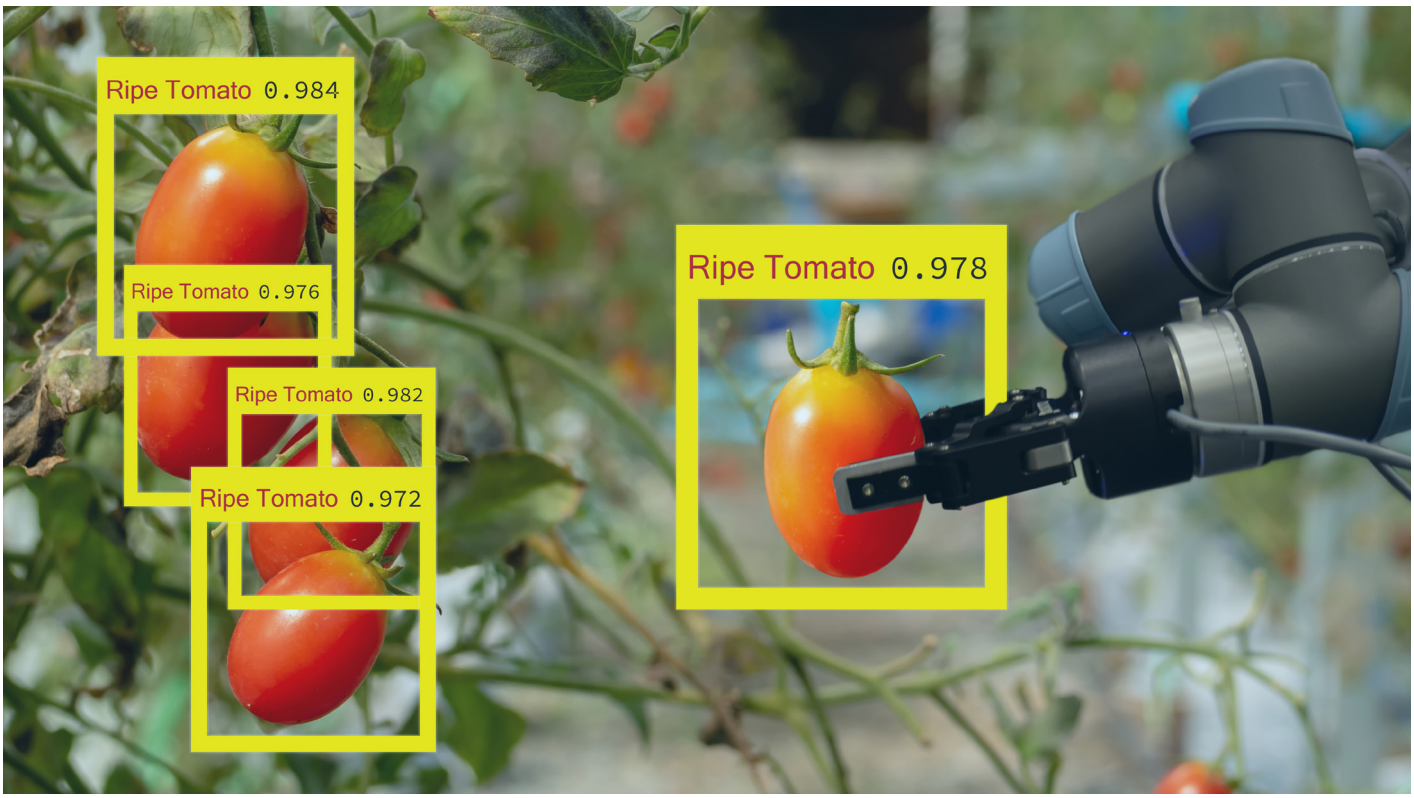


Figure 4: An example of classification and bounding box data generated by detection.

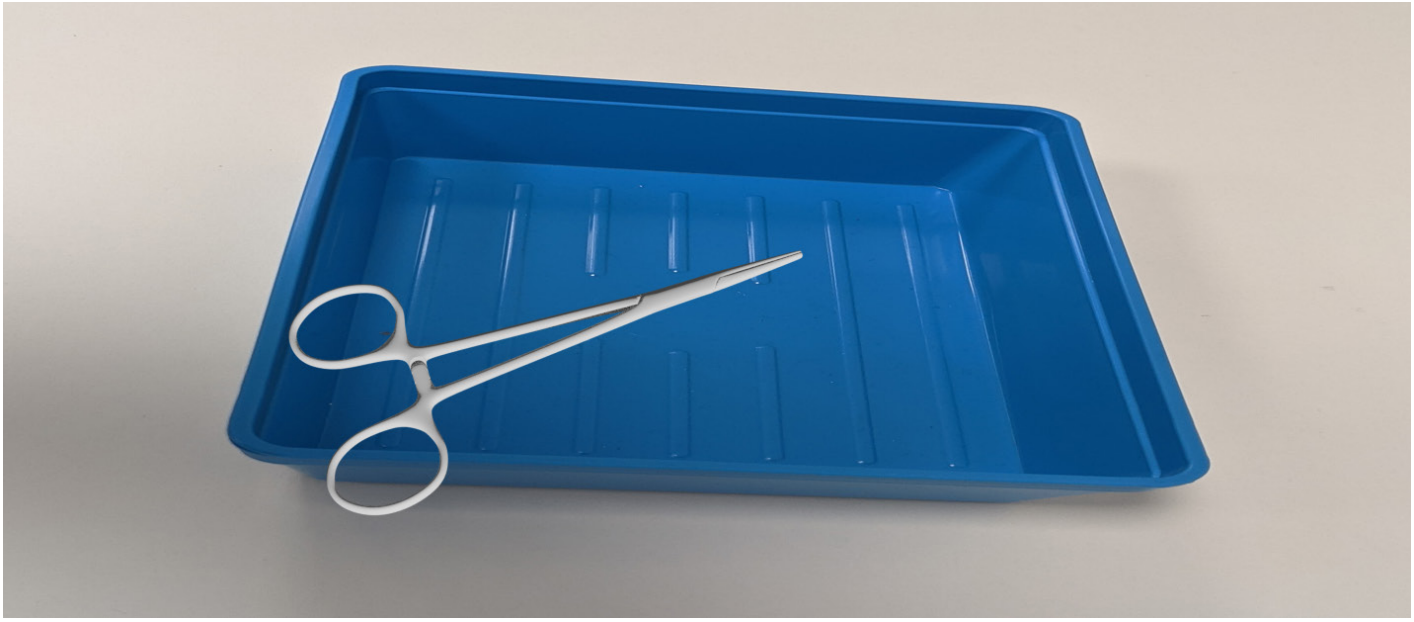


Figure 5: Example image of a render of forceps generated via the dataset generation tool.

Innovation

The AMRC Design and Prototyping Group digital design team have developed a tool to speed up the process of dataset generation by automatically creating labelled renders of objects from their CAD data. These renders can be used as a replacement for photographs for training machine vision models. The tool is capable of generating 100 1080p renders with bounding box data in 11 minutes. The tool uses OBJ files with materials and more optimised model files make for faster renders. The tool is built in Python and uses the Blender API to generate the renders of the CAD models from a random angle then places the render on a randomly selected background image for context.

The process starts by loading the CAD model into a new blender scene via script using the Blender API, then a chosen number of points around the object are selected to be the camera angles for the renders. The camera angles are chosen to be random points on a sphere surrounding the object, where the radius of the sphere is also varied randomly between a minimum and maximum to provide the dataset with renders from various distances.

The light source within the scene is given a random rotation before each render is executed. The bounding box data is then extracted from the scene by converting the vertex coordinates in scene space to 2D screen space coordinates and selecting the furthest most top, bottom, right, and left points. These coordinates are then translated into the desired format required by the model used for training and are saved in a companion file for each image rendered.

After a render is complete it is overlaid on a background image randomly selected from a folder of images and saved. An example of a render can be seen in Figure 5. The background photographs were taken to be suitable context for the use case, however any image can be used. Ideally these background contextual images could also be added to a wider collection or library that can be used for future dataset generation.

The tool was used to generate a dataset of 600 images, 100 for each class for a set of surgical tools. The dataset was then used to train a YOLOv5 object detection model (Ultralytics YOLOv5) successfully. For the examples shown below the model was trained for 200 epochs with a 500/100 train/validation split taking ~1.5 hours using an Nvidia RTX 3080 graphics card. The trained model was capable of detecting the surgical tools in real images despite only being trained on the renders generated by the tool.

Result

Figure 6 shows an example of the model's output for a real image. In this example the forceps were successfully located and classified. Despite the small training set and short training time the model was capable of generalising from CAD renders to real world images extremely well. Unlike real world example images the tool only uses one object per render. This did not prove to be an issue as during training the images are stitched together in batches and randomly cropped and/or shifted to further augment the dataset and aid in model generalisation.

However, the model does not generalise perfectly, Figure 7 shows a miss-classification of a retractor as tweezers. Figure 5 shows a more common problem where an object is given multiple classifications on top of each other. The double classification error is common between scissors and forceps, which are two very similar objects. This confusion between scissors and forceps is a problem that is also present in networks that were trained on labelled photographs The misclassification is unavoidable at certain viewing angles due to the close similarity of their shapes and features.

Figure 6: Image showing the output of the trained model using a real image of forceps as the input. The model successfully locates the forceps and classified them with 92 per cent confidence.

Figure 7: Image showing the model output for a photograph of a retractor. The model misclassified the retractor as a pair of tweezers.

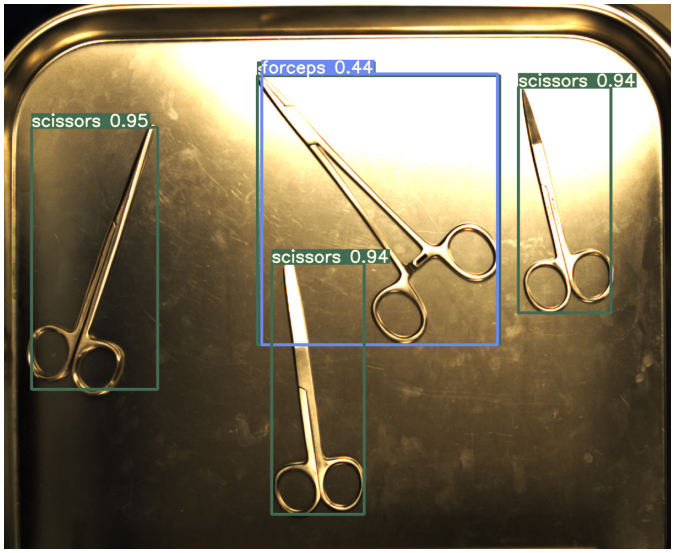


Figure 8: An image showing the output of the model for an image of scissors and forceps. Note the double classification of the forceps in the middle as both scissors (dark green bounding box) and forceps (blue bounding box).

Figure 9: An example of the trained models being used to run a digital shadow board. The surgical tools are detected and compared against a previously made checklist of tools.

Impact

The proof of concept tool shows promise and the results from the model show that a virtually generated dataset can be used to train a model to detect real objects. Through further training and refinement of the generated renders that are created by the tool, the results can be improved for solely virtual dataset training, or the dataset can be merged with real labelled photographs.

In testing, the model performs comparably to a network trained solely on traditionally labelled photographs. Images with visual characteristics not generated in the renders created by the tool in its current form, such as overexposure or reflections, can be troublesome for the trained model. This is an argument for combining real data with the synthetic generated images, and also highlights potential improvements to the tool.

This tool can be used to save hundreds of hours capturing and hand labelling photographs of objects or the costs involved in outsourcing this work. If the CAD and suitable background images are available, the tool can be used to generate a dataset in its entirety or to bulk out an existing set for a more robust model.

The label data is not subject to human error and the bounding boxes are pixel perfect, depending on the size of the CAD models the hundreds of images can be generated in minutes in comparison to traditional methods of dataset generation which can take hours.

As a specific example, capturing and labelling a dataset of 500 images could take a single engineer 5+ hours depending on the number of objects per image, however, this tool could generate 500 labelled renders in under an hour.

The tool does not require the user's attention during this time, so could be used overnight, out of hours or set up to be created automatically when new parts or products are being designed as an additional output. Even when accounting for the overhead of capturing appropriate background images, the synthetic data can be generated many times faster than a real dataset.

Further work

As stated above, there are certain shortcomings of the tool in its current form, as well as potential applications that require further testing and development. From this point the digital design team would like to take the tool further with more development work and the opportunity to test it in an industrial environment.

The tool's functionality can be expanded to include more expansive render options, more realistic visuals, and more model compatibility. Increased visual fidelity will further increase performance of models trained on the generated data and improve the tools applicability to use cases that require it. The data output formats can also be expanded so that other formats are supported. Currently only bounding box information is saved in YOLO format, however this can be expanded and the choice can be given to the user for what format they require.

Another feature and possibly primary focus could include deforming or adding simulated wear and tear via texturing of the CAD model. This means the tool could be used to train models for defect detection in images on the models given.

As the tool in its current form is mostly driven using command line, the primary goal of further development would be to improve the usability of the tool, by giving the script pipeline a user interface to enable quick and easy use of the tool would be a large improvement and the first step to implementing the tool in an industrial environment. The user interface could be used to modify settings in the render and select the models, background images and other customisation options.

References

[1] Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[2] Planche, B., Wu, Z., Ma, K., Sun, S., Kluckner, S., Lehmann, O., Chen, T., Hutter, A., Zakharov, S., Kosch, H. and Ernst, J., 2017, October. Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition. In 2017 International Conference on 3D Vision (3DV) (pp. 1-10). IEEE.

[3] Wong, M.Z., Kunii, K., Baylis, M., Ong, W.H., Kroupa, P. and Koller, S., 2019. Synthetic dataset generation for object-to-model deep learning in industrial applications. PeerJ Computer Science, 5, p.e222.

[4] Mayer, N., Ilg, E., Fischer, P., Hazirbas, C., Cremers, D., Dosovitskiy, A. and Brox, T., 2018. What makes good synthetic training data for learning disparity and optical flow estimation?. International Journal of Computer Vision, 126(9), pp.942-960.

For further information please contact Scott Herod:

0114 222 9588 | s.herod@amrc.co.uk | amrc.co.uk