OPEN ACCESS
IOP Publishing
Measurement Science and Technology
Meas. Sci. Technol. **34** (2023) 075001 (16pp)
https://doi.org/10.1088/1361-6501/acc59e

Developing digital twins of multi-camera metrology systems in Blender

C Pottier[1,*], J Petzing[1], F Eghtedari[2], N Lohse[1] and P Kinnell[1]

[1] Wolfson School of Mechanical, Electrical & Manufacturing Engineering, Loughborough University, Loughborough, United Kingdom
[2] Manufacturing Technology Centre (MTC), Coventry, United Kingdom

E-mail: c.pottier@lboro.ac.uk

Received 26 September 2022, revised 3 January 2023
Accepted for publication 20 March 2023
Published 29 March 2023

Abstract

Blender is an open-source three-dimensional animation software, which can be used as a simulation tool in metrology, to build numerical models that can be used in the design and optimisation of camera-based measurement systems. In this work, the relevance of using Blender to model camera-based measurement systems was explored. Two experiments were conducted in real-world and Blender modelled environments, one using individual cameras for a simple measurement task, the other considering multi-camera position optimisation. The objective was to verify whether the virtual cameras created in Blender can perceive and measure objects in the same manner as the real cameras in an equivalent environment. The results demonstrate that in its native modelling format Blender satisfies the optical metrology characteristics of measurement, but the correlation between Blender output and real-world results is highly sensitive to initial modelling parameters such as illumination intensity, camera definitions and object surface texture.

Keywords: Blender, camera metrology systems, digital twin, MATLAB, photogrammetry, software accuracy, reprojection error

(Some figures may appear in colour only in the online journal)

1. Introduction

Today's manufacturing industries are constantly evolving and integrating new technologies to meet their persistent needs for production efficiency, flexibility, competitiveness and sustainability. The concept of the fixed production line is being challenged by new paradigms such as modular production with dynamic reconfiguration [1–4]. Robotics and automation are being explored for difficult and or repetitive tasks, with hybrid human–robot collaboration being developed for improving decision making and productivity [5–8].

In this contemporary industrial setting and infrastructure, remote monitoring systems are and will be increasingly needed and used for; ensuring the safety of workers, monitoring different production technologies, production optimisation, and collision avoidance [9–12]. Motion capture systems have already been developed and are an implemented solution as illustrated in 2018 by Ford, in its production site in Valencia, by using body tracking technology [13, 14]. Based on 15 small movement tracking light sensors, connected to a wireless detection unit, the skin-tight suit tracks how a person moves at work to enhance worker's postures and productivity, and design tailor-made workstations.

However, such systems need to have cameras accurately placed around the work volume, and the workers have to wear markers on themselves or wear a cumbersome suit.

* Author to whom any correspondence should be addressed.

The deployment of this type of technology breaks the natural process of the factory by stopping the production system and can subconsciously bias the responses and movements of the staff.

Other solutions exist based on markers which can be reflective or light emitting. For instance, StarTracker developed by Mo-Sys [15] uses retro-reflective stickers covering an entire scene, for tracking purposes. This system is comprised of a single camera, that reports in real time, the position and orientation of the studio camera to the rendering engine, according to the position of the retro-reflective stickers detected.

Other familiar and widely used examples of this type of technology are the Vicon and the Nikon K-CMM systems. The Vicon technology is based on passive LED calibration, i.e. the retroreflective markers are tracked by infrared cameras [16]. In contrast, the Nikon K-CMM is based on active LED calibration, i.e. the LED markers emit light which is tracked by special cameras. Moreover, it is the most flexible and common technology used in industry [17]. They are used in diverse areas across various industries [18], including clinical settings [16] but their deployment and tracking volume are quickly limited by the localisation of the markers in the measurement environment due to the camera's field of view (FOV) and the line of sight along with associated occlusions, similar to all solutions based on markers [19].

Markerless motion capture technology has been developed to overcome these issues. One of the most notable solutions is Microsoft's Kinect widely used for different applications within, biomechanical, industrial and medical settings [19–21]. Based on an red, green, blue plus depth data (RGB-D) sensor, this technology uses synchronised colour and depth images for tracking and detection applications [19].

Multi-camera network systems used with photogrammetry are an alternative to the Kinect. Photogrammetry is used as a three-dimensional (3D) measuring tool in industrial and engineering works by recreating a 3D model of an object/room from a set of camera images [22]. Employed as a measurement tool for tasks such as; calibration, inspection, monitoring [23], photogrammetry evolved with industrial need and advancing camera technology. Comparisons have been made between Microsoft's Kinect v2 and photogrammetry [24], with Kinect v2 accuracy being cited as approximately 10 mm (distance dependent) noting issues of ease of use, with the detection being limited by the size of the object and outdoor work also limited. In comparison, photogrammetry is typically limited by the choice of the camera used, as well as uncertainty about ideal location of individual camera nodes.

In the factory of the future, monitoring systems have to be flexible, agile, space-saving, optimised and reliable. However, the current physical available solutions are not necessarily optimised, cameras are installed according to geometry specificities such as being collinear [25], or through learning trial-and-error processes and mathematical models [26, 27], with some algorithms helping to improving results [27, 28].

Due to the recurrent nature of the optimisation problem, and the necessity for smart manufacturing to connect the physical and virtual spaces [29], digital models or even potentially digital twins are being investigated to solve this problem [30, 31], allowing the possibility for rapid investigation of installation and positioning issues with significantly reduced collateral and financial overheads at the investigation stage. In addition, such modelling would allow for rapid changing of camera specifications as a further system optimisation.

In this context, the concept of digital twins is quite recent and some questions concerning reliability, the quality of the digital model of a scene/object, and the analysis of the simulation results, are addressed and answered by characterising such digital modelling environments [32, 33] through a metrology process. In these studies, the development of the digital twin environment is considered, whereby motion tracking camera positions can be flexibly and rapidly modelled to determine optimum locations—thus significantly reducing practical set-up times within the work place. However, the solutions presented are predicated on designing a whole virtual camera from the original camera sensor through to the radiometric characteristics of a real camera. This is a complex process, that is not necessarily accurate and may be time-power consuming.

Other alternatives exist such as the virtual cameras used for rendering in 3D animation software. A wide range of such software platforms are available such as; Autodesk 3ds Max, Autodesk Maya, Blender, Cinema 4D, SOLIDWORKS, and Unity. However, most of these are specialised in discrete aspects of the 3D animation market, and thus force the user to have to switch between different platforms depending on work context [34].

For instance, Autodesk Maya is a reference tool for 3D animation within the film and special effects industry thanks to its customisation capabilities to adapt to different pipelines. In comparison, Autodesk 3ds Max has powerful polygonal modelling tools within its 'Graphite' module and a large library of 3D content, making it more suited for architectural visualisation and real-time 3D [34]. An alternative environment is provided by Blender which is a multi-tool 3D modelling software providing more generalist processing/modelling environment suitable for many different types of application.

Blender is a free open-source computer graphics software toolset used for creating animated films, visual effects, interactive 3D and virtual reality applications. It combines the functionality to model the interactions of scenes with illumination sources, and generate simulated images based on modelled cameras using its internal rendering engine based on ray-tracing and Python scripting options.

Ray tracing allows the modelling of illumination sources and the interaction of the light with 3D objects in a visually realistic way, to create photorealistic scenes. The Python scripting capability, based on API (application programming interface) commands, can be used to customise the application and write specialised tools, bringing the freedom to manipulate and automate the created scene. This enables many variants of a simulated scene and the associate network of cameras and light sources, that form the measurement system, to be automatically created.

This provides a potential environment for the automated exploration of design choices when creating a camera-based 3D measurement system. Recent digital applications using Blender relevant to metrology have been developed, such as BLAINDER [35], an add-on for Blender allowing different depth sensors to be loaded from pre-sets; customised sensors can be implemented and different environmental conditions (e.g. influence of rain, dust) can be simulated, and BlenderProc [36], a procedural pipeline helping in generating realistic looking images for the training of convolutional neural networks. The applications for this work are numerous, such as segmentation, depth, and normal and pose estimation.

However, there is very limited evidence that Blender itself has been considered as a viable metrology compliant environment for multi-camera system modelling. Specifically, there is an opportunity and a need to determine if modelled multi-camera-based systems for object measurement simulated in Blender correlate well with the equivalent real-world multi-camera measurement systems. If such correlation were determined then this would better promote the use of platforms such as Blender for virtual modelling of metrology systems and hence better optimisation of camera system placement in the real world.

In this context the investigation of metrology compliance within Blender requires assessment of performance metrics whilst executing virtual measurements, in addition to considering the potential of the virtual metrology solution. The research presented here considers the initial investigation and suitability of using Blender to model multi-camera measurement systems, with the eventual aim of allowing rapid positional optimisation of in-factory multi-camera measurement systems. The research has specifically considered two scenarios—these being; the measurement of ideal objects, and camera position optimisation.

2. Two experiments presentation

To evaluate Blender, two real-world measurement scenarios were defined, with replica measurement configurations developed and simulated in Blender. The first was the measurement of a sphere typically using individual cameras whilst the second involved the position of a multi-camera measurement system relative to camera calibration artefacts.

2.1. Measurement of a sphere diameter with individual cameras

2.1.1. Methodology. The first measurement scenario consisted of measuring the diameter of a sphere using three Raspberry Pi V2 cameras, located on the *x, y* and *z* unity axes, at different distances from the sphere. The Raspberry Pi V2 was chosen because it is commonly available, cheap, straightforward to set up and use, with fixed optics that are characterised by the pinhole camera model. It is noted here that a consequence of the design means that the camera has a very large depth of field although best focus for this camera version is typically found in the first few metres of the image. The wide FOV (horizontally 62°, vertically 49°) makes this camera very suitable to internal confined space work hence its selection.

Spheres are widely used in metrology applications for camera calibration [37, 38], in addition the use of a sphere allows for simulated light interactions with 3D objects. A sphere is also convenient because it is geometrically speaking always observed as a two-dimensional (2D) circle no matter which orientation it is viewed from. This characteristic compensates for potential imperfections of the real and virtual experimental setup, for instance physical camera misalignment, allowing this experiment to be reproduced without adding new sources of error such as detection problems coming from such misalignment.

Whilst a sphere is straight forward in concept the simple geometry still introduces consequences for the measurement of the sphere diameter using a 2D camera. Figure 1 shows a sphere albeit in this case illustrated as a circle. Point C represents the camera's optical centre whilst points A and B are the intersection points between the extremities of the FOV and the circle. These extremities are tangential to the circle.

The segment [AB] is not equal to the real sphere's true diameter, but represents the diameter perceived by the camera because these points are the interception points between the camera's FOV, and the sphere. The angle θ will increase with increasing object distance between the sphere and the camera. Consequently, as the object distance tends to infinity then θ approaches 90°, and the segment [AB] as recorded by the camera would then approximate to the true diameter. However, it is also noted that as the object distance tends to infinity, then the spatial resolution of the camera image will degrade as a function of reduced pixel density thus introducing alternative error terms.

The outcome criterion from this real and virtual experimentation was the comparison between deviation of the measured observable sphere diameter and θ to the theoretical values. The theoretical observable sphere diameter and θ were calculated using the geometry shown in figure 1, and Pythagoras' theorem (equation (1)).

2.1.2. Real world sphere measurements. The real-world experiment was comprised of a 0.09 m diameter white sphere, two Raspberry Pi V2 pinhole cameras located on the *x* and *y*-axes (the *z*-axis measurement was obtained by rotating the sphere by 90° on the *y*-axis). The cameras were located at an object distance varying from 0.2 m to 2 m, in steps of 0.2 m from the sphere. Measurements were non-sequentially repeated three times to provide understanding of potential variance as a function of camera placement.

The experiment took place in a controlled environment (illumination intensity control, noise coming only from the cameras, etc), with two light sources, one coming from the ceiling, and one coming from underneath the sphere to avoid shadows, as shown in figure 3. The top one was a 46 W LED light unit and, the bottom one was a 25 W LED light unit. A dark background was used to create high contrast between the white room and the white sphere.

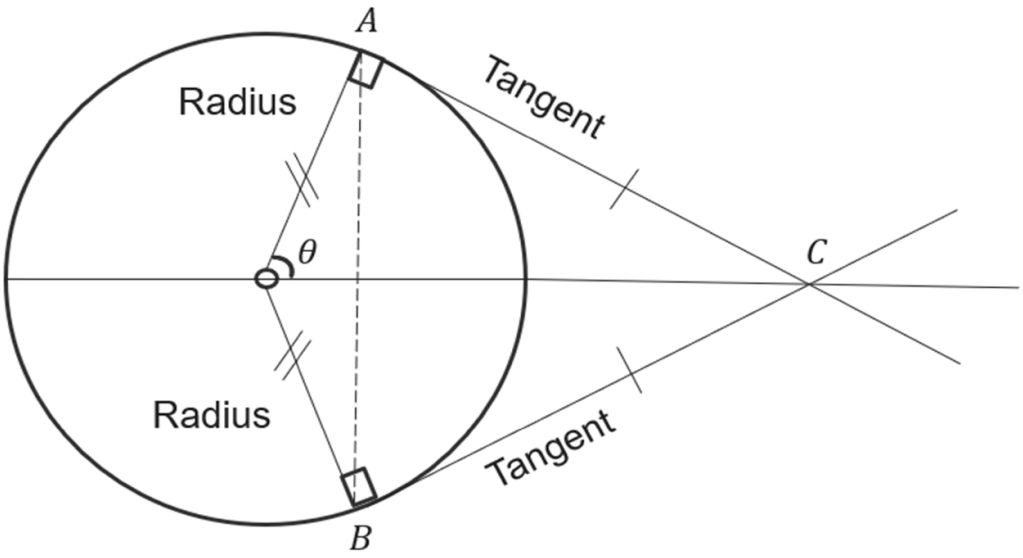

Figure 1. Relationship between a sphere and camera FOV.

A series of MATLAB functions were developed to detect the sphere within the FOV, and return key data elements. Different functions in MATLAB exist to transform a RGB image into a binary image, and to detect a sphere in an image. This research used the *im2bw(originalImage, 0.4)* function for the binary transformation, and the *regionprops* (in the script *regionprops(logical(binaryImage), originalImage, 'all')*) function for the sphere detection, based on the accuracy of the *regionprops* function in detecting a sphere in an image, and returning key values such as the diameter of the sphere in pixels. The camera image was cropped manually (examples shown later in figure 7), by drawing a rectangle around the sphere.

The processing steps followed were:

- Crop the image around the sphere to remove undesirable background elements.
- Load the cropped image.
- Transform the RGB image to a binary image.
- Detect the sphere in the binary image.
- Measure the diameter of the sphere in pixels.
- Calculate the diameter of the sphere in metres:

The diameter in pixels found with *regionprops* was converted to metres using equation (1):

$$\text{diameter (m)} = \frac{\text{distance}_{\text{camera - sphere}}}{\text{focal length}} \times \frac{(\text{sensor}_{\text{width}} \times \text{diameter (pixel)})}{\text{Image}_{\text{width}}}. \quad (1)$$

2.1.3. Blender sphere measurements. The real-world experimentation was then recreated in the virtual environment. The experiment again consisted of measuring the diameter and centre of a 0.09 m diameter white virtual sphere from pictures taken by the Blender simulated virtual pinhole camera located at (again) object distances varying from 0.2 m to 2 m, in steps of 0.2 m from the sphere. The whole scene was generated in Blender using a Python script as shown in figure 2. For the purposes of visualisation within this report, figure 3 does not include the black background that was used to maximise object contrast. Multiple repeats ($n = 3$) of the experiments were completed to determine any variance in the output.

A dedicated camera model was created in Blender and used to simulate the Raspberry Pi V2 pinhole camera. The parameters of both the real and virtual cameras were; focal length of 3.03 mm, image resolution of 1640 × 1232 pixels (using 1.12 μm square pixels), and a sensor size of 3.68 mm × 2.76 mm, as defined in the camera datasheet [39].

The Blender model sphere was modelled with 32 polygons on a black background, noting that objects created in Blender are all based on a polygon mesh. Hence, the number of polygons represent the resolution of the sphere geometry, consequently the more polygons the sphere is modelled with, then in theory the more 'spherical' the sphere will be. Polygon density was considered in a separate series of background experiments with resolutions of 100 and 1000 polygons, but any differences of results compared with 32 polygons were found to be negligible.

In Blender, illumination intensity is defined depending on the light source. For instance, sunlight is defined in terms of watts per square metre but when considering artificial lighting (spot or area lights) then the lighting is specified in watts noting that this relates to radiant flux and not to electrical energy [40]. Consequently, it is difficult to directly correlate illumination intensity between a real-world environment and the equivalent Blender model environment. In this series of experiments, two square area lights with default values initially fixed at 10 kW were placed at 1 m above and below the sphere to illuminate the whole sphere homogeneously. Whilst 10 kW would be a significantly large value in the real-world scenario, within Blender this allowed for even illumination with minimal shadowing.

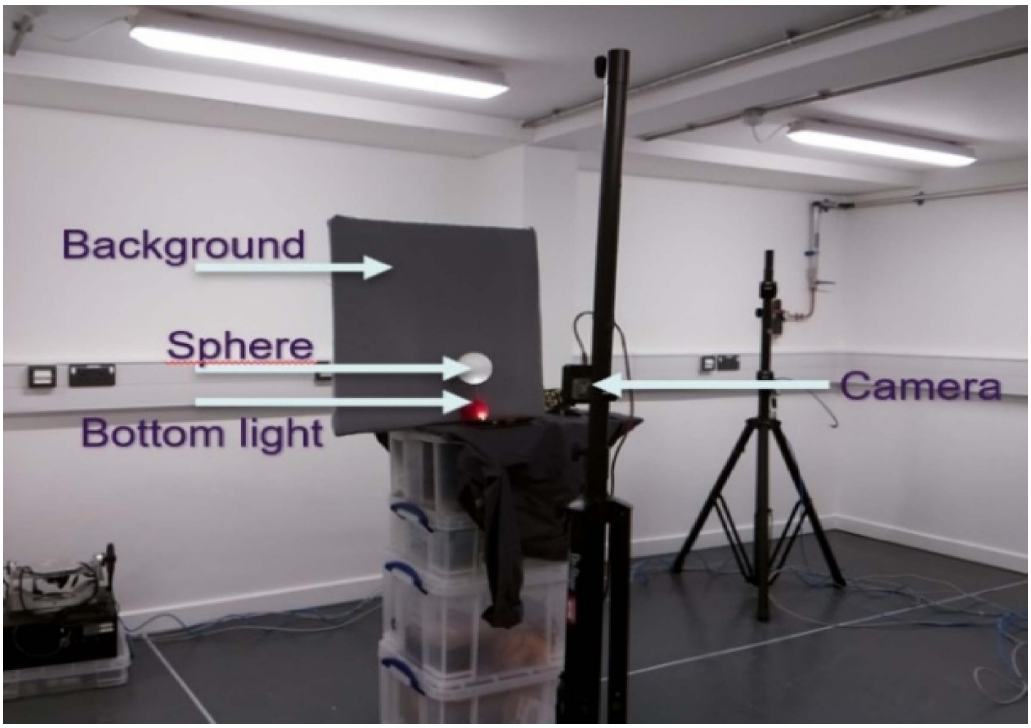

Figure 2. Real-world experiment.

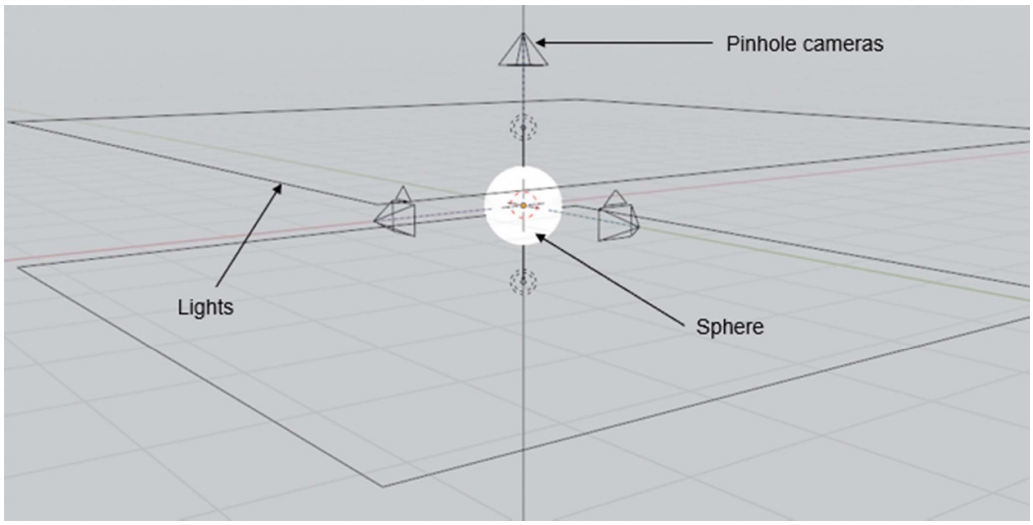

Figure 3. Virtual sphere and camera set-up in Blender.

Three pinhole cameras were set up on the *x, y* and *z* axes to view the sphere from different points of view. The Blender modelling environment was set up to be perfect (no noise, distortion, object surface texture, etc were introduced). The same MATLAB functions and data processing developed for the real-world scenario were used.

2.2. Camera location relative to a calibration artefact

2.2.1. Methodology. Whilst the first series of experiments was to assess camera performance in Blender in the context of object (sphere) measurement, the second series of experiments was designed to evaluate the ability of Blender to predict the camera location relative to a camera calibration artefact, using photogrammetry techniques. This consisted of comparing the reprojection errors of the detection of the centre of five zone circles printed on a calibration board using an array of eight Raspberry Pi V2 cameras—both in the real-world environment and the Blender virtual environment.

Photogrammetry uses triangulation to rebuild a 3D scene from multiple separate images, requiring a minimum of two cameras. Background experiments were completed with a number of cameras (ranging from three to eight), to determine which combination of cameras were likely to give lower

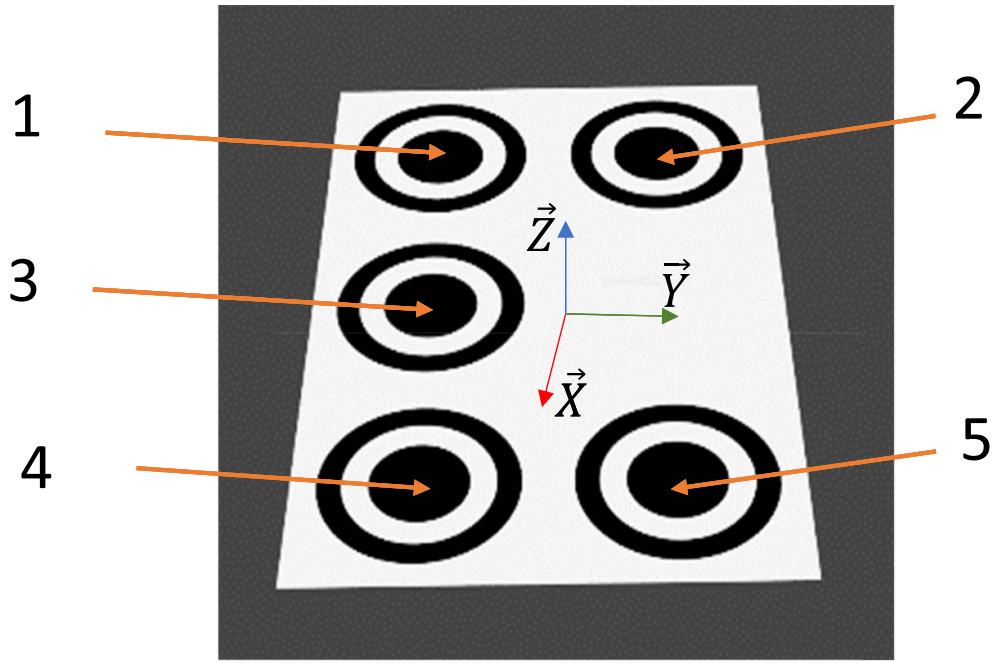

Figure 4. Artefact used for camera location tests.

reprojection errors. According to the background results, eight were chosen.

The reprojection error [41] is a geometric error corresponding to the image distance between a projected point and a measured point. In this instance, based on the standard deviation of set of measurements, the reprojection error is used to quantify the extent to which the estimation of a 3D point recreates the real projection of the point. In the case of detection, this error makes it possible to quantify the quality of circle (centre) detection by a set of cameras and to observe the variations in detection as a function of the positions of the cameras in the scene. It is noted that quantification of reprojection errors is inconsistently reported in literature and can also be reported as a function of the root-mean-square of all the reprojection norms, mean of the reprojection errors, and, median of the reprojection errors.

With respect to equation (2), the standard deviation represents the variation or dispersion of a set of samples around the mean on the basis that the data is parametric and follows a Gaussian or Normal distribution. In this instance it is used as a measure for the reprojection error, but does not allow specific identity of error components or individual contributions

$$\text{Error} = 1.96 \times \frac{\|\sigma\|}{\sqrt{N_e}}. \tag{2}$$

where; σ is the standard deviation of the error mean, N_e is the number of errors, and 1.96 the multiplication factor to obtain a result at a coverage factor of $k \approx 2$ (95% confidence limits based upon a Normal distribution).

Circular targets have been widely used in image processing and computer vision for their invulnerability to partial occlusion and their ease of access, which increases their use in optical measurement systems [42–44]. However, the accuracy of circle centre detection is sensitive to distortion, centre deviation and other stochastic factors in the imaging process [42]. To minimise the impact of these sensitivity factors, the pattern used was comprised of five zone circles. For better detection and localisation in the image by MATLAB, an alternation between black and white zone circles was used.

There is significant debate within the computer image community about the performance characteristics of zone circle-based calibration artefacts versus checkerboard-based artefacts with performance dependent on the nature of the pixel processing algorithms used. In this instance, reliable results were obtained with the zone-circle pattern because all pixels on the periphery of the circles can be used to accurately determine the circular shape of the pattern. In addition, the zone circles enabled three estimates of the target centre to be made by considering the central black circle, the white circle that surrounds it and finally the outer back circle.

In figure 4, the zone circles cover almost the entire FOV, which facilitates detection from different viewpoints. Circle 3 positioned at 0 mm from the x-axis was used as a reference to check the measurements, and the placement of the pattern on the virtual board. The origin of the frame is placed in the centre of the artefact, as shown in figure 4, along with the zone circle order.

The zone circles were 0.1 m, 0.15 m and 0.2 m respectively in diameter on a white diffuse substrate measuring 0.5 m $\times$ 0.75 m $\times$ 0.025 m. However, due to the overall size of the zone circle artefact, it was challenging to cause the entirety of the object to be within the FOV of each camera at 0.2 m, 0.4 m, 0.6 m and 0.8 m object distance—the artefact being partially cropped within the images at these object distances. A smaller additional artefact was physically

Table 1. Coordinates of the circles for the two zone circle artefacts.

Circle number	Coordinates large artefact (x,y) (mm)	Coordinates small artefact (x,y) (mm)
1	−0.125, −0.250	−0.0523, −0.0871
2	0.125, −0.250	0.0523, −0.0871
3	−0.125, 0	−0.0523, 0
4	−0.125, 0.250	−0.0523, 0.0871
5	0.125, 0.250	0.0523, 0.0871

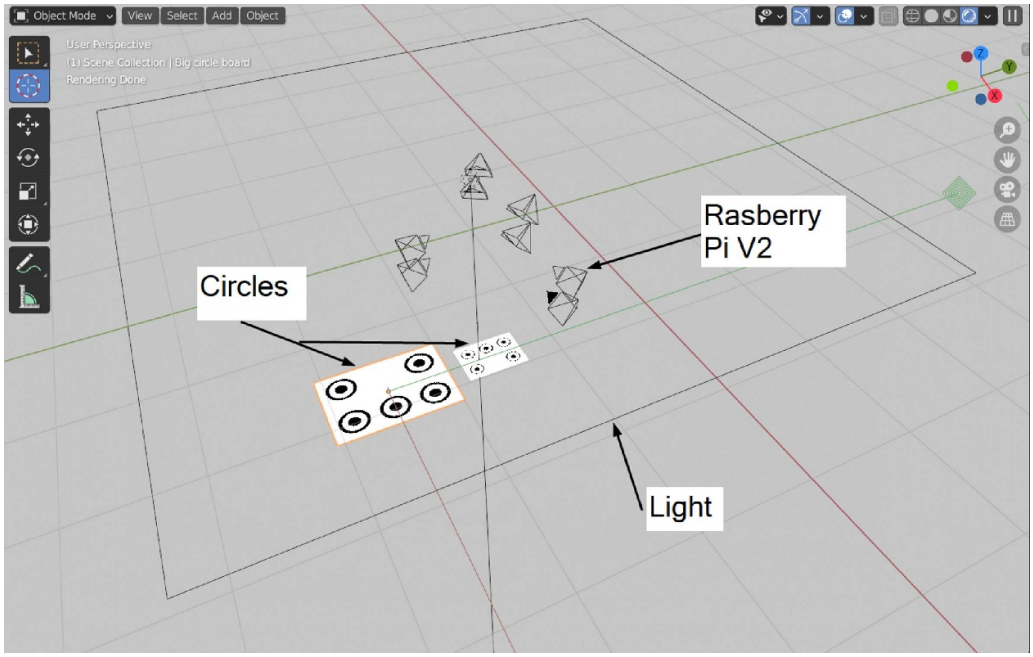

Figure 5. Blender scene using the zone circle artefacts.

developed whereby the zone circles were 0.029 m, 0.048 m and 0.069 m respectively in diameter on a substrate measuring 0.21 m × 0.29 m × 0.005 m. Both artefacts were likewise simulated in Blender by importing the original digital artwork. The (x,y) coordinates of each zone circle for both the small and large artefacts are listed in the table 1 with reference to the origin sited at the middle of each board.

The coordinates of each individual zone circle centre were extracted from each set of camera images, before using triangulation to create a 3D reconstruction of their positions in relation to the camera positions. The reprojection error for each zone circle centre was determined by calculating the deviation between the reference circle centre coordinates and the coordinates determined within MATLAB, with the final answers expressed in terms of the standard deviation.

2.2.2. Real world multi-camera location. Eight real Raspberry Pi V2 cameras (with the same characteristics as those used for the sphere experimentation), were placed around the real zone circle calibration board. Two cameras were used at each position defining a triangular pattern of analysis (in this case illustrated in figure 5 as part of the Blender virtual equivalency). The distance between each camera on the z axis was 0.25 m. The camera-object distance varied from 0.2 m to 1.8 m (limited by physical room dimensions) incrementing in steps of 0.2 m. Multiple repeats ($n = 3$) of the experiments were completed to determine any variance in the output.

The calibration zone circle board was placed in the middle of the measurement volume, approximately equidistant to each camera. A dedicated MATLAB based set of algorithms were developed to complete the data processing and return key data elements. The processing steps followed were:

- Improve the contrast of each picture to allow for better detection with the function *imadjust(Image, [0.6, 0.7])*.
- Define the focal length and sensor size of the camera according to the datasheet [39].
- Calculate the intrinsic matrix with *cameraIntrinsics(focal, principalPoint, imageSize)*.
- Filter the noise in the image with the function *medfilt2(Image, 'indexed')*.
- Transform the RGB image to a grey scale image with the function *rgb2gray(DenoiseImage)*.
- Transform the grey image to a binary image with the function *imbinarize(GrayImage, 'adaptive')*.
- Calculate the complementary image with the function *imcomplement(BinaryImage)*.
- Detect the centre of each circle with the functions *regionprops(Image, 'basic')* on the binary and complimentary

images before using k-means *(centers, 5)* to find the centre of the cluster created.
- Check the order of the circle centre.
- Circles 1-3-4: The cross product of the vectors created between circles 1-4, 1-2 is calculated, and its sign checked. If the sign is null, the centre of the board is detected correctly; if the sign is positive circle indices 1 and 4 are correct;

  Circles 2-5: The angles between circles 1-4, and 1-2 is calculated, as well as the angle between 1-5, and 1-4, using simple trigonometric functions. If the circle order is correct, the angle formed between 1-4, and 1-2 is higher than the other one.
- Calculate the camera parameters (rotation matrix and translation vector) with the function *extrinsicsToCameraPose(centres, reference centres, intrinsics parameters)*.
- Calculate the 3D coordinates of each target centre by triangulation using the point reprojected in the image frame, and the camera positions.
- Calculate the reprojection error.

The reprojection errors were calculated as based on the description given in section 2.2.1.

2.2.3. Blender based multi-camera location. Within Blender, an equivalent set of eight virtual Raspberry Pi V2 cameras were placed in a triangular set up around the board, to mimic the real-world scenario as shown in figure 5. Camera definitions were those as used for the sphere measurements with data processing following the MATLAB algorithms developed for the real-world scenario (section 2.2.2).

3. Results

3.1. Sphere measurement with individual cameras

3.1.1. Real-world sphere analysis. The results from the real single camera measurement of the 0.09 m sphere are shown in figure 6 demonstrating the deviation of the measured sphere diameter from the actual diameter. The x-axis scale represents the distance between the camera position and the sphere, the left side y-axis scale is the difference between the theoretical and measured diameter observable (deviation or residual), and the right y-axis scale is the area of the sphere within the FOV in camera pixels. The mean value of the multiple repeat results from the three camera positions on the orthogonal sphere axes are shown. The standard deviation of the trial data with respect to the diameter deviation was 0.59 mm, 0.61 mm, and 0.53 mm respectively for Trial 1–3, whilst the standard deviation across the trial repeats at each object distance ranged from 0.0002 m to 0.013 m, the latter value occurring at 0.4 m object distance and attributed to geometry error as a function of figure 1.

The data demonstrates that the three trials generate similar results, with similar data trends. The absolute deviation of the sphere diameter decreases with increasing object distance. The initial significant deviation from the true diameter is a function of the camera FOV relationship when viewing a sphere as defined in figure 1. The deviation improves with increasing object distance as noted with three sets points in the data summarised in table 2. Here it is noted that there is a discrepancy between the Trial 1 deviation data and that for Trials 2 and 3. This has been attributed to subtle differences of camera set-up between experiments (noting that the three trials were not completed contiguously) and noise elements within the experiment.

The size of the sphere defines the number of camera pixels that constitute the sphere within the FOV. As expected, as the camera object distance increases, the sphere is resolved using fewer pixels thus potentially reducing the accuracy of measurement and hence the accuracy of sphere diameter measurement at longer distances with all three camera positions showing this trend. Moreover, the camera resolution decrease causes increasing image blur as shown in figure 7, likewise affecting the quality of measurement although it is also noted that the V2 version of the Raspberry Pi camera tends to suffer from defocus issues at longer object distances as a function of the original factory build and configuration.

3.1.2. Blender environment sphere analysis. The results from the Blender modelled single camera measurement of the 0.09 m modelled sphere are shown in figure 8. This likewise demonstrates the deviation of the measured sphere diameter from the actual modelled diameter, and decreasing camera pixel resolution as a function of object distance using the same axis descriptions as for figure 6. Unlike the real environment experiments, the Blender analysis did not use three repeats. This is because the algorithmic nature of Blender means that repeating the same analysis multiple times with the same data input leads to negligible variance of data output. This was previously determined through background experimentation.

Initially, the sphere diameter deviation decreases up to 0.6 m object distance, but then unlike the real-world experiment the deviation negatively increases as the object distance progresses to 2 m. However, it should be noted that the left side y-axis scale factor in figure 8 is an order or magnitude smaller than the equivalent in figure 6, and if plotted on the same scale as figure 6 would approximate to a straight line. In the first instance, the deviation comes from the sphere geometry, and the diameter measurement is function of the camera FOV (see figure 1). The subsequent negative increase of diameter deviation is due to the lighting and the resolution of the camera. In addition, it was determined that the same blurring process occurred in the virtual model (similar to figure 7). It is noted here that during further background experimentation it was determined that too much light within the Blender environment exaggerated the environment of the sphere yet did not seem to saturate the virtual camera image sensor—this being a key limitation of real cameras yet not noted as being a characteristic of the Blender camera model.

Figure 9 illustrates this point. Using the Blender setup described in section 2.1.3, and the methodology given in section 2.1.1, seven light powers (5 kW, 1 kW, 0.5 kW, 0.1 kW, 75 W, 50 W and 46 W) were simulated to determine

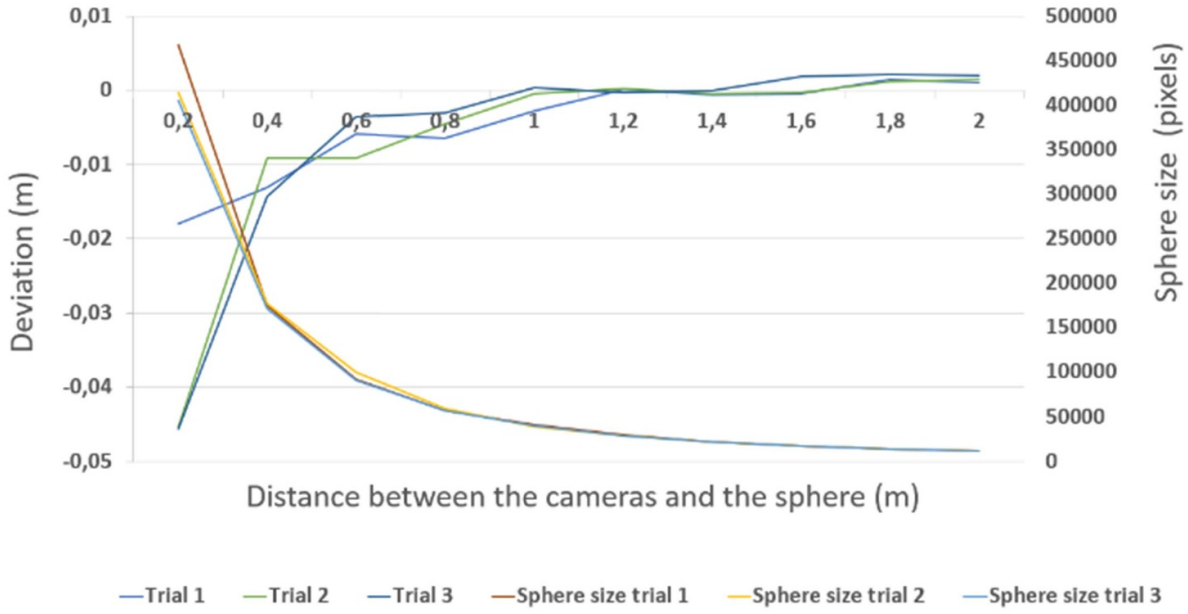

Figure 6. Deviation of sphere diameter measurement for the real environment.

Table 2. Deviation of real environment sphere diameter and pixel resolution for an object distance of 0.2 m, 0.6 m and 2.0 m.

Object distance	0.2 m	1.0 m	2.0 m
Deviation Trial 1 (mm)	−18	−2.70	1.13
Deviation Trial 2 (mm)	−45	−0.37	1.42
Deviation Trial 3 (mm)	−46	−0.38	2.02
Sphere size Trial 1 (pixels)	467 382	41 115	11 272
Sphere size Trial 2 (pixels)	414 135	38 280	11 232
Sphere size Trial 3 (pixels)	405 048	40 261	10 961

the impact of light power on the sphere diameter measurement in the virtual environment. However, to further remove the geometry error influence illustrated in figure 1, a 0.09 m diameter circle was modelled instead of a sphere—placed on a black background to achieve similar white/black high contrast associated with the previous sphere measurements.

Figure 9 shows the diameter deviation is greater for high power (5 kW, 1 kW and 0.5 kW), than for low power (0.1 kW, 75 W, 50 W and 46 W). Thus, the more powerful the light and illumination, the more the details of the circle are exaggerated, resulting in errors in the detection of the circle (and consequently sphere) exacerbated as a function of increasing object distance. This background element of work confirms that object specific illumination thresholds may exist but can only be determined on a case-by-case basis.

The initial choice of 10 kW illumination was originally and correctly justified to maintain even illumination across the entire sphere and remove shadowing (section 2.1.1) but this still generated errors when detecting the diameter of the sphere. It has been identified that in Blender, the illumination set-up and characteristics does not necessarily give the same result as the same illumination in the real world, this partly being a function of the colour transform used in Blender to generate the rendered image. By default, Blender uses the sRGB colour transform, which was originally designed to approximate the response of a cathode-ray tube monitor [45].

Table 3 also reveals that the size of the sphere is not the same between the real and the virtual image even though the camera parameters were set-up as per the manufacturer's specification. For instance, at 0.2 m for Trial 1 (table 2), the size of the sphere in the real image is 467 382 pixels compared to a virtual equivalent size of 305 561 pixels (table 3). As the object distance increases then the difference in pixel count becomes more obvious. This is related to the manner in which Blender generates images, whereby pixel count is linked to the output of the rendered scene rather than necessarily relating it directly to the number of pixels at the camera image plane.

In addition, this difference may be due to the grey scale of the real-world image versus the Blender equivalent. In Blender, the sphere is perfectly white [255, 255, 255], and the background perfectly black [0, 0, 0], whereas in the real-world case the white and black elements of the image are not perfectly white and black but contain grey scale components of varying values. In MATLAB, to detect the sphere in an image, a binary image process is employed and uses a threshold to determine the grey scale components of the image based on the Otsu method [46]. Due to the different grey scale values in the image, fewer pixels are used to detect the sphere in MATLAB.

Figure 7. Sphere cropping and blurring at 0.2 m, 1 m and 2 m.

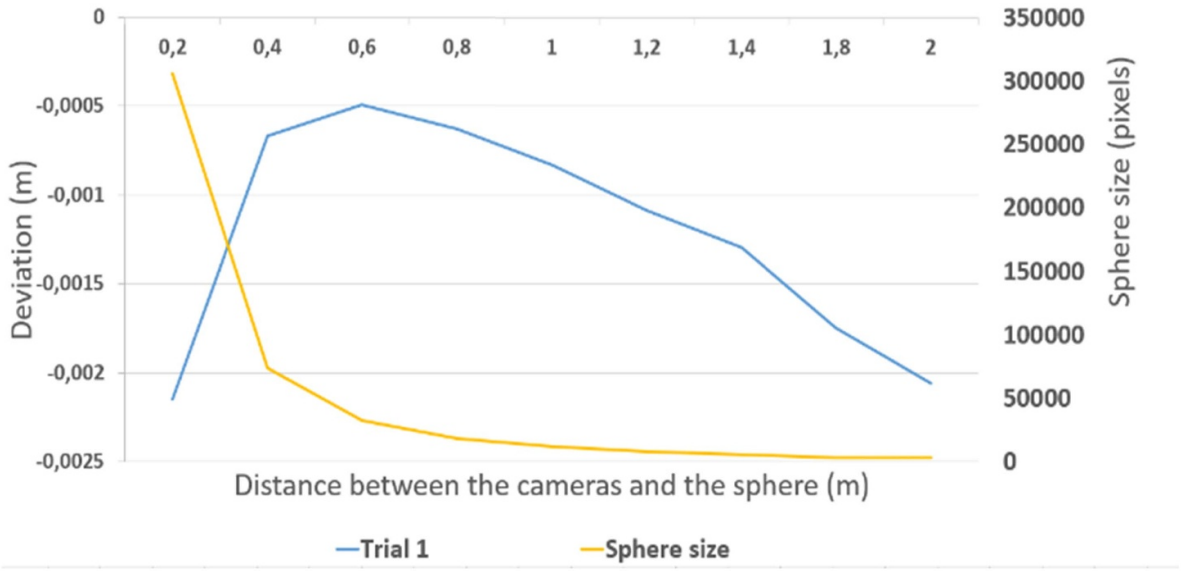

Figure 8. Deviation of sphere diameter measurement for the Blender environment.

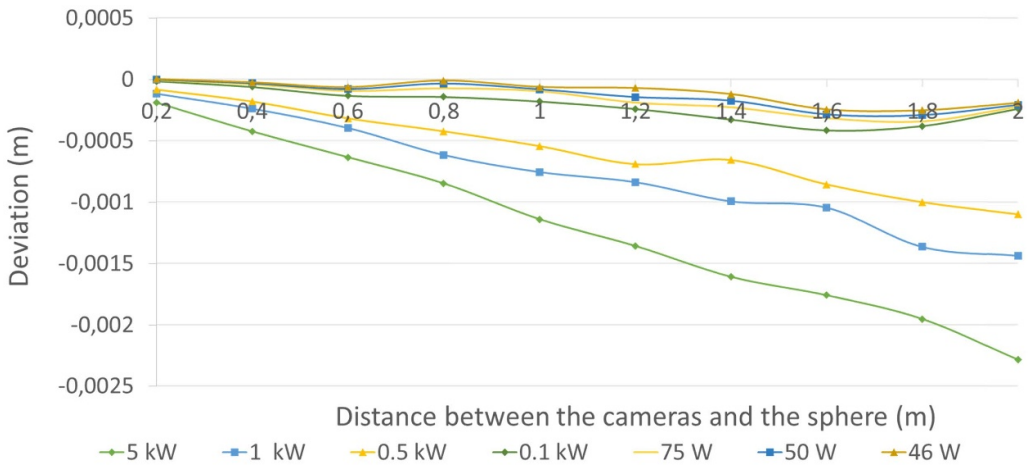

Figure 9. Light variation in Blender for a 90 mm circle.

3.2. Camera position related to a calibration artefact

3.2.1. Real-world reprojection error analysis. The results from the real-world multiple camera measurement of the calibration zone circles are shown in figure 10 noting that again for the real-world investigation there were three repeats of the experimentation. The x-axis represents the distance between the cameras and the calibration artefact and the y-axis represents the reprojection error of the centre of each circle detected. Due to data density as a function of

Table 3. Deviation of Blender environment sphere diameter and pixel resolution for an object distance of 0.2 m, 1.0 m and 2.0 m.

Object distance	0.2 m	1.0 m	2.0 m
Cam on x (mm)	−2.15	−0.83	−2.06
Cam on y (mm)	−2.15	−0.81	−2.03
Cam on z (mm)	−2.08	−0.74	−2.15
Sphere size (pixels)	305 561	11 870	3743

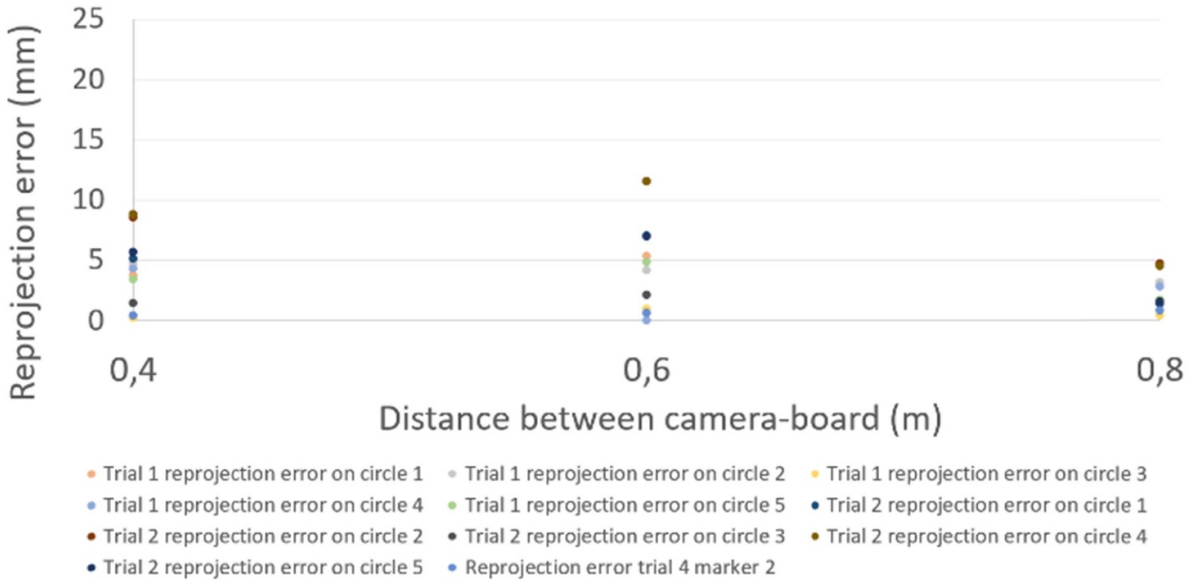

Figure 10. Real environment reprojection error (0.4–0.8 m).

five repeat experiments for each trial, only Trials 1 and 2 are shown here to illustrate both magnitude and trend characteristics.

Two sets of data are presented for each experiment using the two separate zone circle artefacts. The smaller artefact was used for an object distance varying between 0.4 m and 0.8 m (figure 10), the larger artefact was used for object distances varying between 1.0 m and 1.8 m (figure 11).

The data (also shown in tables 4 and 5 respectively) demonstrates that the three trials generated similar results, with similar data trends for both versions of the artefacts, but not with the same magnitude of results. For instance, for zone circle 1 at 0.4 m, the reprojection error was 5.317 mm for trial 1, 7.002 mm for trial 2 and 0.272 mm for trial 3 respectively. The difference between the results comes from the script used for the detection, and the physical setup. Because the tripod mounted cameras were moved manually from 0.4 m to 1.8 m, it was found to be very challenging to maintain consistent camera rotation and positional parameters (with six translational and rotational degrees of freedom available) during repositioning at each object distance. In addition, due to the variation in camera configuration, the light on the surface of the sphere was not perceived by the camera in the same manner, which led to errors in the detection of the sphere in the image during processing in MATLAB.

With respect to figure 10 and table 4, the reprojection error increases between 0.4 m and 0.6 m, before decreasing. The reprojection error is larger for zone circles 1, 2, 4 and 5, whilst zone circle 3 has the smallest value. This difference is attributed to the fact that the zone circles are paired (1 with 4, 2 with 5) but 3 is an individual zone circle. With increasing object distance, zone circles 1 and 4 start to become more difficult to identify, likewise for 2 and 5. Moreover, zone circle centre detection might not be the true geometric centre (for each zone circle) due to optical changes of perspective and depth of field. In addition, the circles are 'ovalized', changing slightly the centre coordinates and the height of the board.

It is noticeable that Trial 2 gives a larger reprojection error than Trial 1 at 0.4 m and 0.8 m, a function of the physical setup, the camera location and orientation, and, the lighting condition. In addition, the circle detection script may be sensitive to variation, as a function of incorrect detection of the circles caused by the image contrast and the illumination environment.

With respect to larger zone circle artefact (figure 11 and table 5) the reprojection error decreases between 1.0 m and 1.6 m, albeit before slightly increasing at 1.8 m for Trial 1 and 2. For Trial 3, the reprojection error increased from 1.2 m to 1.4 m, before slightly decreasing. For zone circles 1, 2, 4 and 5 the reprojection errors were larger, whilst zone circle 3 again had the smallest value.

Similar trends for Trials 1 and 2 are observed across both sets of data shown in tables 4 and 5. However, the results for zone circle 3 are of similar values for both trials, and the results

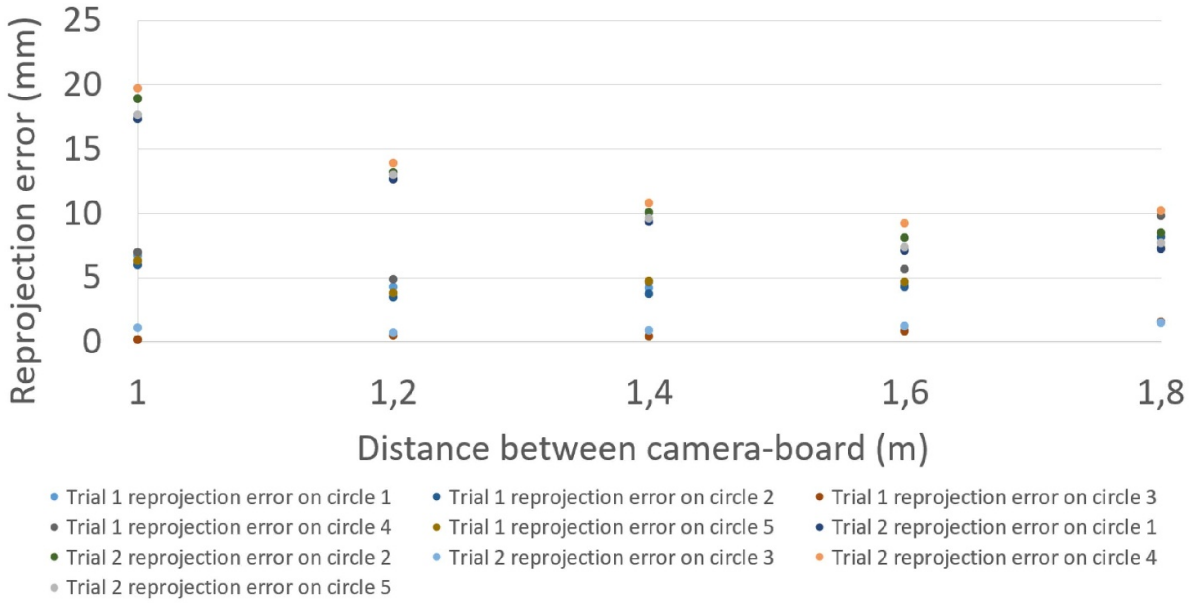

Figure 11. Real environment reprojection error (1.0–1.8 m).

Table 4. Real environment reprojection error (0.4–0.8 m).

Zone circle (Trial 1)	Object distance 0.4 m	Object distance 0.6 m	Object distance 0.8 m
1 (mm)	5.32	1.61	4.10
2 (mm)	4.18	3.15	1.16
3 (mm)	0.92	0.39	2.17
4 (mm)	3.95	2.78	1.36
5 (mm)	4.82	1.71	3.87
Zone circle (Trial 2)			
1 (mm)	7.00	1.36	12.06
2 (mm)	11.55	4.66	6.27
3 (mm)	2.09	1.56	3.95
4 (mm)	11.50	4.47	6.75
5 (mm)	6.92	1.49	13.01

Table 5. Real environment reprojection error (1.0–1.8 m).

Zone circle (Trial 1)	Object distance 1.0 m	Object distance 1.4 m	Object distance 1.8 m
1 (mm)	4.23	4.21	6.87
2 (mm)	3.47	3.71	8.16
3 (mm)	0.50	0.43	1.53
4 (mm)	4.81	4.65	9.83
5 (mm)	3.78	4.70	7.31
Zone circle (Trial 2)			
1 (mm)	12.63	9.35	7.23
2 (mm)	13.18	10.03	8.44
3 (mm)	0.72	0.91	1.48
4 (mm)	13.91	10.80	10.17
5 (mm)	12.99	9.59	7.71

for the large artefact analysis shown in table 5 are larger than in table 4 showing that reprojection errors increase with object distance. This is to be expected due to changing pixel resolutions as a function of object distance and orientation of the camera.

3.2.2. Blender reprojection error analysis. The analysis of the reprojection errors determined within the Blender virtual environment is presented in a similar fashion to the real-world analysis within figure 12 (short distance) and figure 13 (long distance), supported by tables 6 and 7 respectively.

With respect to both figures and tables of data, the reprojection error for zone circles 1, 2, 4 and 5 increases with increasing object distance, whilst zone circle 3 is fairly consistent (and small) in its reprojection values. This is similar overall behaviour to that seen in the real-world experiments. Zone circle 3 may be exhibiting the best results because of its position within the coordinate framework of the artefact and analysis. In addition, texture adjustment (see below), camera distortion, and board size were all found to have varying effects on the y-axis component of the measurements, when defining the height of the array.

The data presented in tables 6 and 7 further demonstrates that the Blender environment reprojection error appears to be influenced by the size of the calibration board. At 0.4 m the reprojection error is between 0.02 mm and 3 mm, at 1.2 m, the reprojection error is between 0.05 mm and 0.7 mm. It is also clear that there is a step change in magnitude between the use of the two boards, with the smaller board causing larger reprojection errors. It is anticipated that the small board is more difficult to detect than the larger one due to its size, and

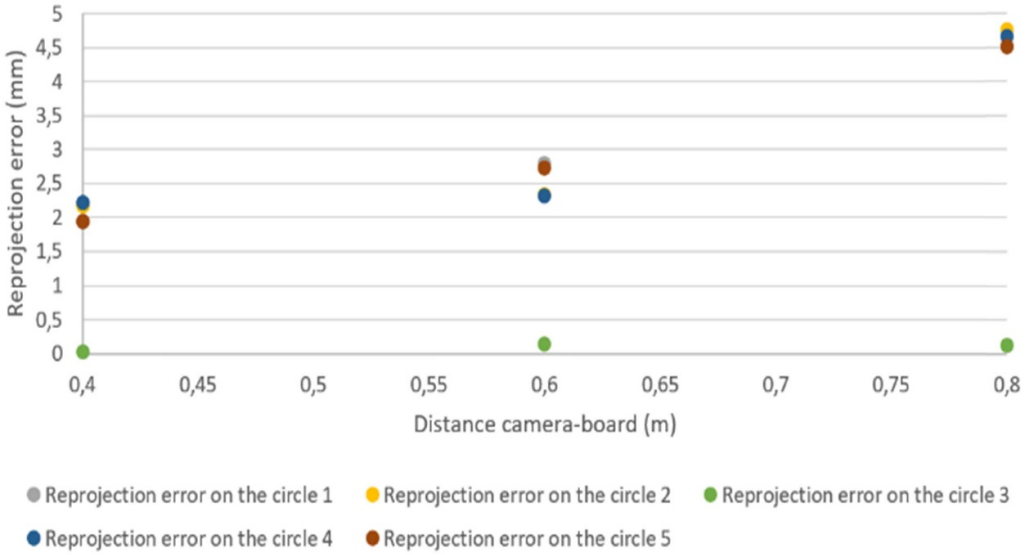

Figure 12. Blender environment reprojection error (0.4–0.8 m).

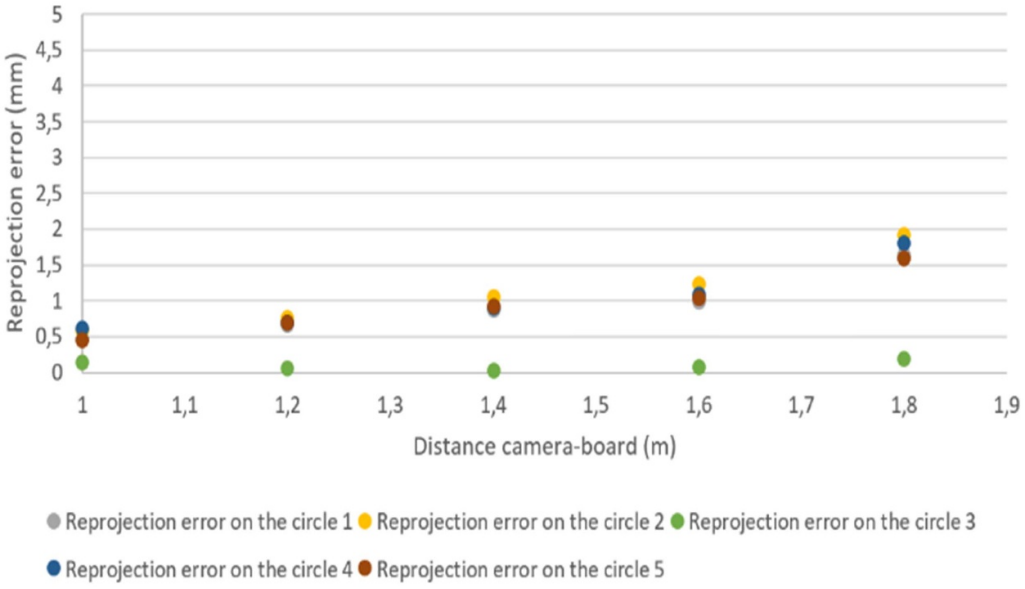

Figure 13. Blender environment reprojection error (1.0–1.8 m).

Table 6. Blender environment reprojection error (0.4–0.8 m).

Zone circle	Object distance 0.4 m	Object distance 0.6 m	Object distance 0.8 m
1 (mm)	2.64	2.80	4.52
2 (mm)	2.48	2.34	4.76
3 (mm)	0.03	0.03	0.13
4 (mm)	2.54	2.21	4.66
5 (mm)	2.67	1.95	4.51

consequently there is more potential for the merging of the circles within the camera images as a function of increasing distances and reducing pixel resolution. Clearly, this does not follow the real-world equivalent trends for the two sizes of artefact.

Analysis of the data and also reflecting on the requirements for establishing the virtual models in Blender, reveal that there are (as expected issues with matching the model to the real-world. For instance, in the real-world—the zone circle artefact is printed on flat laminate board which has a natural element

Table 7. Blender environment reprojection error (1.0–1.8 m).

Zone circle	Object distance 0.4 m	Object distance 0.6 m	Object distance 0.8 m
1 (mm)	0.67	0.87	1.64
2 (mm)	0.77	1.05	1.91
3 (mm)	0.05	0.02	0.19
4 (mm)	0.67	0.91	1.81
5 (mm)	0.67	0.92	1.59

of (diffuse) surface texture—likewise the spheres in the diameter measurements exhibit texture. These real textures can be explicitly measured and quantified in terms of 2D or 3D surface texture parameters.

However, in Blender the menu options used in the set-up of the model provide texture descriptors that are considerably more broad ranging in composition and interpretation (based on nodes, light absorption and reflection properties) that are significantly subjective and qualitative in nature, and do not appear to directly relate to the quantified metrics [40]. Texture adjustment can be achieved with the (Blender) size tool, or with the smart match tool. It was observed that iterative adjustment of the object texture caused a change of the coordinates of the centres of the zone circles, which led to incorrect detection and inflated reprojection error terms. Hence it is difficult to directly correlate the texture on the modelled zone circle board with real-world numerical values (likewise the sphere in the diameter measurement scenario).

4. Discussion and conclusions

This research has been instigated to help start to determine if a 3D virtual reality modelling environment such as Blender can be used to fully model camera-based metrology systems, and eventually support camera-based measurement system placement within industrial environments. The research has used two exemplars to explore and highlight the potential but also the issues found; firstly, the measurement of a sphere using individual cameras, and secondly, the measurement of a camera calibration artefact used in the case of multiple camera positions (in this instance eight cameras).

With respect to the sphere experiments, the virtual and real-world responses have significant similarities but also differences. Trend wise the two responses behave similarly from 0.2 m to 0.6 m camera-object distance, but differ thereafter. The initial error input at very close object distance (visible in both environments), predominantly comes from the camera/sphere geometry relationship shown in figure 1. The second error input (visible in both environments), predominantly comes from the camera/sphere geometry relationship shown in figure 1. The second error input (visible particularly in the virtual environment) comes from the decreasing image pixel resolution as a function of object distance. This explanation is further facilitated by the size of the sphere being smaller (based on pixel count) in the virtual environment than in the real environment even though geometrically speaking the virtual sphere was the same specification (0.09 m diameter). This means that the virtual sphere in the virtual image, for the same image size and resolution, is composed of fewer pixels than in the real image. This is partly caused by the manner in which image generation is defined in Blender rather than the resolution of a camera.

The real-world scenario and results are less accurate than the virtual environment. This is shown in terms of the order of magnitude difference for the real results (tens of millimetres) compared to the virtual results (millimetres). This disjuncture between the two environments and the data is predominantly a feature of two issues. Firstly, the real-world will contain a range of error sources (e.g. camera parameters, differences between cameras, precision and accuracy of camera location, lighting, etc) which will not have been encoded into the virtual model. And secondly, the modelling within the virtual environment is dependent on the menu options provided and the quality of the model boundaries defined by the operator (e.g. light power and design of light sources, illumination wavelengths, object texture simulation, specular and diffuse reflections, camera definitions, etc).

In the camera location and calibration experiments, the results from the virtual and real experimentation are found to be different for similar camera-object definitions. With reference to the data tables, the real world reprojection error results are between 0.43 mm and 14 mm, whilst the virtual results are between 0.03 mm and 4.8 mm. Furthermore, the results present differently as a function of object distance. As explained above, the definition and repeatability of the set up in both environments will be sources of error and performance differences for these experiments.

A persistent outcome from these two exemplars has been that (in the majority) the scenes created in Blender gave better results than the real scenes, noting that significant care and attention has been given to both the set-up of the real-world and virtual experiments. Whilst differences are clear—overall trend behaviour of the virtual cameras is close enough to the real cameras to be able to conclude that the laws of optics are being respected. An example of this is illustrated by the geometric behaviour of the real and virtual systems incorrectly measuring sphere diameter when too close to the sphere (figure 1).

However, whilst Blender has been shown to perform better than reality and shows significant promise as a tool to aid investigative multi-camera metrology systems, this performance capability is in itself a source of concern. The work demonstrates that it is inadvisable to use Blender in a raw or default environment configuration (light interactions, texture, camera characterisation, etc) to design a digital twin of

reality from an observable point of view, because it is lacking a 'realism' that correctly captures the vagaries of the real-world environment.

In other science and engineering modelling scenarios this would be termed as correct/incorrect definition of modelling boundary conditions. Yet aspects of a real-world environment being measured by camera systems are at times exceedingly subjective as opposed to be easily quantified. And this also assumes that the modelling software provides menu options and variable control that have the capability to correctly mimic such real-world variables from a metrology sense, albeit they may model very well from a visual artistic sense. A case in point found throughout these two exemplars has been the difficulties found with consistent and corresponding illumination of the two environments, and definitions of surface texture of objects. Surface textures are important because of light-object interactions. As shown in figure 9, light has an impact on the detection of the sphere. It is further anticipated that the surface texture will also have an impact on the measurement and should be taken into account.

The research has demonstrated that the current Blender environment needs to be more 'realistic' whilst at the same time it is also recognised that such work needs to also minimise real-world experimental errors. Solutions for improving the Blender environment are to better characterise the real and virtual cameras, determine a correlation between real object surface texture and equivalent virtual objects, and, understand how to correctly mimic lighting design (for instance; wavelength bandwidth, intensity, diffusivity and reflectivity).

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

This work was supported by the Manufacturing Technology Center (MTC) in Coventry (UK), and Loughborough University in Loughborough (UK).

ORCID iD

C Pottier https://orcid.org/0000-0002-4382-3352

References

[1] Mehrabi M G, Ulsoy A G and Koren Y 2000 Reconfigurable manufacturing systems: key to future manufacturing *J. Intell. Manuf.* **11** 403–19
[2] Setchi R M and Lagos N 2004 Reconfigurability and reconfigurable manufacturing systems: state-of-the-art review *2nd IEEE Int. Conf. on Industrial Informatics, 2004, INDIN '04* pp 529–35
[3] Michalos G, Makris S and Chryssolouris G 2015 The new assembly system paradigm *Int. J. Comput. Integr. Manuf.* **28** 1252–61
[4] Jovane F, Koren Y and Boer C R 2003 Present and future of flexible automation: towards new paradigms *CIRP Ann.* **52** 543–60
[5] Tsarouchi P, Makris S and Chryssolouris G 2016 Human–robot interaction review and challenges on task planning and programming *Int. J. Comput. Integr. Manuf.* **29** 916–31
[6] Mizanoor Rahman S M, Liao Z, Jiang L and Wang Y 2016 A regret-based autonomy allocation scheme for human-robot shared vision systems in collaborative assembly in manufacturing *2016 IEEE Int. Conf. on Automation Science and Engineering (CASE)* pp 897–902
[7] Rahman S and Wang Y 2015 Dynamic affection-based motion control of a humanoid robot to collaborate with human in flexible assembly in manufacturing *Proc. ASME Dynamic Systems and Controls Conf.* p V003T40A005
[8] Gombolay M C, Gutierrez R A, Clarke S G, Sturla G F and Shah J A 2015 Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams *Auton. Robots* **39** 293–312
[9] Liu Z, Xie K, Li L and Chen Y 2020 A paradigm of safety management in Industry 4.0 *Syst. Res. Behav. Sci.* **37** 632–45
[10] Bragança S, Costa E, Castellucci I and Arezes P M 2019 A brief overview of the use of collaborative robots in Industry 4.0: human role and safety *Occupational and Environmental Safety and Health. Studies in Systems, Decision and Control* (Cham: Springer) (https://doi.org/10.1007/978-3-030-14730-3_)
[11] Muehlmann U, Ribo M, Lang P and Pinz A 2004 A new high speed CMOS camera for real-time tracking applications *IEEE Int. Conf. on Robotics and Automation, 2004. Proc. ICRA '04* pp 5195–200
[12] Gawande U, Hajari K and Golhar Y 2020 Pedestrian detection and tracking in video surveillance system: issues, comprehensive review, and challenges *Recent Trends in Computational Intelligence* ed A Sadollah and T S Sinha (London: IntechOpen) (https://doi.org/10.5772/intechopen.90810)
[13] Ford looks to motion tracking bodysuits to aid factory workers 2018 (available at: https://internetofbusiness.com/ford-motion-tracking-bodysuits/) (Accessed 24 August 2022)
[14] Ford improves car making the same way sports stars raise their game—with body tracking 2018 (available at: https://media.ford.com/content/fordmedia/feu/en/news/2018/08/03/ford-improves-car-making-the-same-way-sports-stars-raise-their-g.html) (Accessed 26 August 2022)
[15] Mo-sys StarTracker (available at: www.mo-sys.com/product/camera-tracking/startracker/) (Accessed 1 September 2022)
[16] Petrosyan T, Dunoyan A and Mkrtchyan H 2020 Application of motion capture systems in ergonomic analysis *Armenian J. Spec. Educ.* **1** 107–17
[17] Drouot A, Zhao R, Irving L and Ratchev S 2019 Towards Industry 4.0: the future automated aircraft assembly demonstrator *Precision Assembly in the Digital Age. IPAS 2018. IFIP Advances in Information and Communication Technology* vol 530, ed S Ratchev (Cham: Springer)
[18] Vicon.com What is motion capture (available at: www.vicon.com/about-us/what-is-motion-capture/) (Accessed 6 September 2022)
[19] An Shen T T 2014 Marker-less motion capture for biomechanical analysis using the Kinect sensor *Bachelor's Thesis* Universitat Politècnica de Catalunya
[20] Sturm J, Engelhard N, Endres F, Burgard W and Cremers D 2012 A benchmark for the evaluation of RGB-D SLAM systems *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* pp 573–80
[21] Giancola S, Corti A, Molteni F and Sala R 2017 Motion capture: an evaluation of Kinect V2 body tracking for upper

limb motion analysis *Wireless Mobile Communication and Healthcare. MobiHealth 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* vol 192, ed P Perego, G Andreoni and G Rizzo (Cham: Springer)
[22] Fraser C S and Brown D C 1986 Industrial photogrammetry: new developments and recent applications *Photogramm. Rec.* **12** 197–217
[23] Luhmann T 2010 Close range photogrammetry for industrial applications *ISPRS J. Photogramm. Remote Sens.* **65** 558–69
[24] Lachat E, Macher H, Landes T and Grussenmeyer P 2015 Assessment and calibration of a RGB-D camera (Kinect V2 sensor) towards a potential use for close-range 3D modeling *Remote Sens.* **7** 13070–97
[25] Lee S H, Kim B J and Lee S B 2021 Study on image correction and optimization of mounting positions of dual cameras for vehicle test *Energies* **14** 4857
[26] Georgiou M, David S, Loudos G, Tsougos I and Georgoulias P 2012 Experimental and simulation studies for the optimization of dedicated scintimammography cameras *J. Instrum.* **7** 1011
[27] Leiber D *et al* 2022 Simulation-based layout optimization for multi-station assembly lines *J. Intell. Manuf.* **33** 537–54
[28] Olivier P, Halper N, Pickering J and Luna P 1999 Visual composition as optimisation *AISB Symp. on AI and Creativity in Entertainment and Visual Art* vol 1 pp 22–30
[29] Tao F, Zhang H, Liu A and Nee A Y C 2019 Digital twin in industry: state-of-the-art *IEEE Trans. on Industrial Informatics* vol 15 pp 2405–15
[30] Steil T and Roßmann J 2014 Validating the camera and light simulation of a virtual reality testbed by means of physical mockup data *2nd Int. Conf. on Artificial Intelligence, Modelling and Simulation* pp 143–8
[31] Farrell J E, Xiao F, Catrysse P B and Wandell B A 2003 A simulation tool for evaluating digital camera image quality *Proc. SPIE* **5294** 124–31
[32] Jones D, Snider C, Nassehi A, Yon J and Hicks B 2020 Characterising the digital twin: a systematic literature review *CIRP J. Manuf. Sci. Technol.* **29** 36–52
[33] Gohari H, Berry C and Barari A 2019 A digital twin for integrated inspection system in digital manufacturing *IFAC-PapersOnLine* **52** 182–7
[34] Fremox, Quel Logiciel 3D Choisir? (1/4) 2017 (available at: https://motion-cafe.com/quel-logiciel-3d-choisir-01) (Accessed 8 September 2022)
[35] Reitmann S, Neumann L and Jung B 2021 BLAINDER—a Blender AI add-on for generation of semantically labeled depth-sensing data *Sensors* **21** 2144
[36] Denninger M, Sundermeyer M and Winkelbauer D 2019 BlenderProc (arXiv:1911.01911)
[37] Zhang H, Kwan-yee K W and Zhang G 2007 Camera calibration from images of spheres *IEEE Trans. Pattern Anal. Mach. Intell.* **29** 499–502
[38] Zhang H, Zhang G and Wong K Y 2005 Camera calibration with spheres: linear approaches *IEEE Int. Conf. on Image Processing* (IEEE) vol 2 pp II–1150
[39] Raspberry PI (available at: www.raspberrypi.org/documentation/accessories/camera.html) (Accessed 14 September 2022)
[40] Blender manual Light objects (available at: https://docs.blender.org/manual/en/latest/render/lights/light_object.html#power-of-lights) (Accessed 12 September 2022)
[41] Hartley R and Zisserman A 2003 *Multiple View Geometry in Computer Vision* (Cambridge: Cambridge University Press)
[42] Qin L, Feipeng D and Heming H 2010 An improved method of location of the circular target *Int. Conf. on Computational Aspects of Social Networks* (IEEE) pp 251–4
[43] Motai Y and Kosaka A 2008 Hand–eye calibration applied to viewpoint selection for robotic vision *IEEE Trans. Ind. Electron.* **55** 3731–41
[44] Datta A, Kim J S and Kanade T 2009 Accurate camera calibration using iterative refinement of control points *2009 IEEE 12th Int. Conf. on Computer Vision Workshops, ICCV Workshops* (IEEE) pp 1201–8
[45] Blender Guru 2017 The secret ingredient to photorealism (available at: www.blenderguru.com/tutorials/secret-ingredient-photorealism) (Accessed 16 September 2022)
[46] Otsu N 1979 A threshold selection method from gray-level histograms *IEEE Trans. Syst. Man Cybern.* **9** 62–66