Generation of Synthetic Stereo Calibration Patterns for Digital Image Correlation using Blender - Final Report

Jeremy Hill
CS231a
jlhill@stanford.edu

Abstract

Advancements in camera technology and computational power have made optical methods popular in mechanics research. However, these methods often require complex post-processing of images to extract data like displacements. Real-world experiments present challenges like limited depth of field and noise, making it difficult to develop and validate processing algorithms. Synthetic images offer a controlled environment for algorithm development, with access to a known "true" solution. However, existing synthetic image generation tools are often limited in accessibility. Blender, a free and open-source 3D software, offers scene creation, rendering, and Python scripting capabilities. This combination allows for automated image generation and complex analyses, potentially making Blender a more versatile tool than current options. This work will explore Blender's suitability for generating synthetic test images and demonstrate its accuracy with the commercial DIC software from Zeiss.

1. Introduction

There is a new paradigm in mechanical testing of materials that combines full-field deformation measurements like digital image correlation (DIC) and inverse identification to extract mechanical constitutive parameters from complex stress states. Known as Material Testing 2.0 (MT2.0), this approach opens doors to exploring new materials with complex or non-uniform properties. It combines experimental data in the form of DIC measurements with simulation results from finite element analysis to better model material behavior. Most approaches involve some form of minimizing a cost function between experiment and simulation. An area of current research deals with how we map one dataset onto the other in order to calculate that cost function. One approach is to use synthetic images generated from finite element analysis simulations to aid in this mapping.

Developing and testing image processing algorithms with real-world photos can be difficult. Real images often have limitations - they might be blurry due to shallow depth of field, have low contrast, or contain noise. Additionally, camera calibration errors can further complicate analysis. Synthetic images offer a solution. They allow researchers to explore these algorithms in a controlled environment, free from the complexities and expenses of real-world experiments. Furthermore, synthetic images can provide a ground truth, or "perfect" solution, which is usually absent in real-world scenarios. Most synthetic image generation tools are either commercial software or custom research code bases. Blender is an open source 3D software tool that has potential to be used as synthetic image generation tool for engineering purposes.

Digital image correlation refers to a group of non-contact methods for acquiring images of an object, storing the images in digital format, and performing image analysis to obtain full-field shape and deformation measurements [1]. DIC is considered a subset of digital image registration techniques. The method requires at least one-high resolution digital camera and, for 3D displacement measurements, two cameras (stereo DIC) are required.

This work aims to generate and evaluate synthetic stereo calibration patterns for DIC using the open source software Blender.

1.1. Prior Work

Pierron introduced the concept of MT2.0. It utilizes full-field deformation measurements and inverse identification to extract material properties from complex tests (non-statically determinate) [2]. This paper reviews existing MT2.0 research on engineering materials, excluding biological tissues due to their specific deformation measurement methods and material behavior. Following a brief introduction to full-field measurements and inverse identification, the paper categorizes existing research by material type. It then addresses key issues like uncertainty quantification, test design, and standardization before concluding.

DIC data offers valuable information for finite element analysis validation, but inconsistencies between the two require careful handling. Lava et al explore two methods: direct interpolation (addressing some inconsistencies) and DIC leveling (addressing all) [3]. The authors find DIC leveling provides more accurate validation, especially when considering model form errors and data filtering, and recommend it for generating meaningful quantitative comparisons.

Rohe and Jones demonstrated the suitability of Blender for generating synthetic test images for digital image correlation and showed its' accuracy is comparable to a commercial software package [4]. Their study demonstrated that Blender can generate images with sufficient accuracy (around 0.3 millipixel error) to be used for simulating optical DIC tests, even those involving subpixel motions [5]. Notably, the quality of Blender's synthetic images rivaled those produced by commercial DIC software. The researchers noted that Blender could be used to plan experiments, save time and resources by predicting outcomes beforehand. Additionally, Blender facilitates the development and validation of motion extraction techniques through comparisons with analytical data, often unavailable in real-world experiments.

A past CS231a course project investigated 3D reconstruction in Blender using stereoscopic vision [6]. The authors went from a pair of stereo images of a single human in a particular pose to a rendering of a 3D model with the same pose. The team employed OpenPose to extract 2D key points from human figures in stereo images. These key points were then used for 3D reconstruction via triangulation, leveraging pre-solved camera projection matrices. The resulting 3D joint positions were imported into Blender for rendering human models. While successful reconstructions were achieved for static images, even with occlusions, applying the pipeline to videos exposed limitations. The technique struggled with noisy data and inconsistencies between frames due to a lack of consideration for key point movement over time.

Pivonka and Preucil describe a method for generating synthetic stereo images to aid development of a car undercarriage scanner project [7]. The scanner serves for a security inspection of cars. The real scanner uses multiple cameras to capture a car's undercarriage, but real-world testing is expensive. Their method utilized Blender to create virtual cameras and simulate stereo camera motion. This allowed the generated images to be used for debugging the car scanner system throughout its development cycle. However, the authors found a limitation in that the synthetic images could not account for all real-world complexities or errors caused by incorrect camera calibration.

2. Problem Statement

2.1. Dataset

This work relies on a dataset of 132 experimental calibration pattern images and 20 camera parameters provided by the International Digital Image Correlation Society (iDICs). iDICs is composed of members from academia, government, and industry, and is committed to training and educating users of DIC systems and the standardization of DIC practice for general applications.

iDICs has released several challenges with the purpose of advancing the practice of DIC through collective efforts that point to optimum methodologies. The goal of the Stereo-DIC challenges is to provide a framework within which all codes can be tested, validated, and improved for their intended use.

Please see the following link for more information on current and completed iDICs challenges as well as links to the dataset used in the project. `https://idics.org/challenge/`.

Two Point Grey 2.3 Megapixel (Grasshopper 3 GS3-U3-23S6M 5.86 micron pixel size) cameras were used to acquire the dataset. The cameras had a stereo angle of approximately 19 deg using 35 mm Edmund Optics machine vision lenses. Calibration was done using the Correlated Solutions 14x10-7mm calibration target. Note that there are two extra smaller dots that are used by the software to automatically identify the target.

2.2. Project Goals

This course project primarily focuses on implementing the method introduced by Rohe and Jones to generate synthetic calibration patterns using intrinsic and extrinsic camera parameters. Their work is extended to evaluate generated patterns for use with the commercial DIC package Zeiss Inspect. As part of the evaluation process, a stereo calibration script in python is developed in order to compar calibration matrices calculated from both synthetic and experimental calibration panels.

The results are split into roughly three parts:

1. Starting with provided intrinsic and extrinsic camera parameters from iDICs dataset, generate and render synthetic calibration pattern images using Blender.

2. Develop stereo calibration script in python to confirm experimental and synthetic images generate similar camera parameters. This is done using numpy and OpenCV libraries. This is necessary due to the Zeiss software not providing access to the camera parameters in the calibration file generated within their software.

3. Demonstrate that the synthetic images can be successfully imported into Zeiss Inspect and used to generate a calibration file.

2.3. Evaluation Metrics

This work uses three metrics to evaluate success in generation of synthetic stereo calibration patterns for DIC.

1. The difference between camera parameters estimated from experimental images and those estimated from synthetic calibration patterns.
   (a) Confirm experimental dataset and parameters using custom camera calibration scripts developed in Python using OpenCV library.
   (b) Compare difference in experimental and synthetic camera parameters estimated from Python calibration script.
2. The import into Zeiss Inspect is evaluated on a pass/fail criteria.
   (a) Thirteen calibration images of a GOM calibration panel are generated from Blender. These images are imported in the Zeiss software. Success is defined as the software generating the proprietary calibration file.
   (b) A series of experimental test images are imported along with the calibration file to evaluate displacement and strain.
3. Images with sub-pixel rigid translations are created, and the accuracy of the DIC displacement results is evaluated. This is an indirect method for evaluating the rendering as it combines the render accuracy with the DIC software accuracy, but is nonetheless relevant to the use of Blender for creating synthetic DIC images.
   (a) DIC errors and statistics for rigid translations of synthetic images produced in Blender with a nominal set of rendering parameters.

3. Technical Approach

3.1. Camera Model

The approach for this project is to use Blender and OpenCV for camera calibration. The problem of estimating the extrinsic and intrinsic camera parameters is known as camera calibration and is accomplished by solving for the intrinsic camera matrix K and the extrinsic parameters R, T. OpenCV uses the pinhole camera model. To connect these three camera matrices to the parameters estimated in OpenCV and required for defining a stereo pair in Blender, some variables need definition.

An image pixel (u, v) is generated from real world coordinates (x, y, z) through a 3x4 matrix using a projective transformation and homogeneous coordinates.

$$cx = PX$$

where c is a constant, $x = (u, v, 1)^T$, $X = (X, Y, Z, 1)^T$, and P can be further expressed as:

$$P = K\,[I|0] \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = K\,[R|T]$$

The $[K]$ and $[R|T]$ matrices represent the intrinsic and extrinsic camera parameters respectively and transform the physical coordinates XYZ to image coordinates xy.

$$K = \begin{bmatrix} \alpha_u & s & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

$$[R|T] = \left[\begin{array}{ccc|c} r_{XX} & r_{XY} & r_{XZ} & t_X \\ r_{YX} & r_{YY} & r_{YZ} & t_Y \\ r_{ZX} & r_{ZY} & r_{ZZ} & t_Z \end{array}\right]$$

$$c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [K][R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where f is the focal length, s is the skew factor, (u_o, v_o) are the principal point coordinates, and the other intrinsic parameters are defined as follows:

$$\alpha_u = \frac{f}{pixel\ width}$$

$$\alpha_v = \frac{f}{pixel\ height}$$

Figure 1 shows the Blender scene used to render the calibration panel images. A test article (calibration panel or speckle pattern) is placed at the origin of the scene. Left and right cameras are placed and oriented to match the settings from iDICs testing. Two area lights are placed above and behind the cameras to illuminate the test article.

Within Blender, the camera sensor size and focal length are defined by the intrinsic parameters. The stereo angle and baseline distance are defined by the extrinsic parameters.

3.2. Calibration Panels

Two types of calibration panels are utilized for this work. Both employ rectangular dot grid patterns. For the calibration done in Python, a simple 14 x 10 grid is utilized. In order to calibrate within the Zeiss DIC software, specific calibration panels are required as the real world points are pre-coded within the software. Figures 2a and 2b show the original Zeiss calibration panel image and the Blender rendered image from the perspective of the Right camera, respectively.

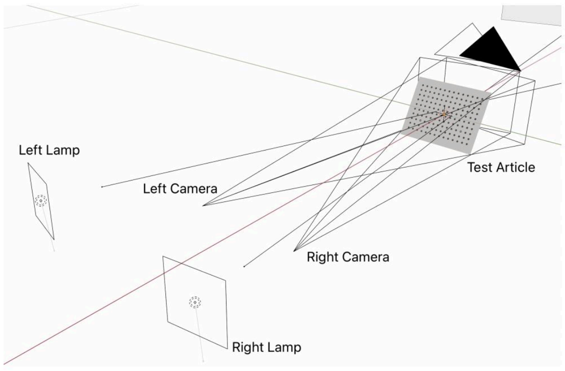
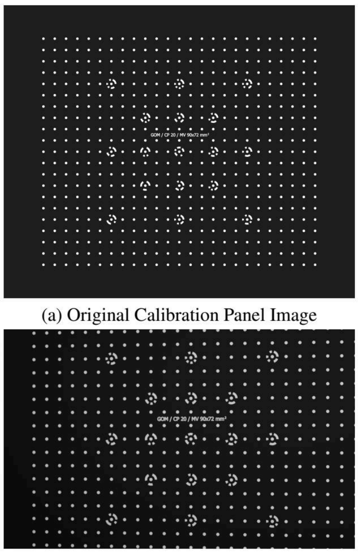


Figure 1. Blender scene with 35mm focal length cameras and two area lights



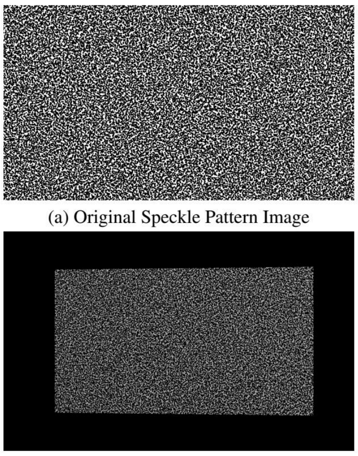
(a) Original Calibration Panel Image

(b) Blender Rendering from Right Camera

Figure 2. GOM Calibration Panels

3.3. DIC Patterns

One fundamental assumption of DIC is that the motion and deformation of the pattern that is imaged exactly replicates the underlying test piece motion and deformation. Sometimes, images of the surface of the test piece itself have a sufficient natural pattern that is adequate for DIC, and no artificial pattern needs to be applied. Most of the time, a pattern must be applied to the test piece surface. Typically, though not exclusively, applied patterns consist of roughly circular "speckles" of a (preferably) uniform size but random locations.Figures 3a and 3b show the original speckle pattern image and the Blender rendered image from the perspective of the Right camera, respectively. This speckle pattern was generated to fit the camera field of view and with speckles of appropriate size for the camera resolution.



(a) Original Speckle Pattern Image

(b) Blender Rendering from Right Camera

Figure 3. Speckle Pattern Images

4. Analysis and Results

A stereo camera calibration python script was developed using the OpenCV library.

Table 1 shows an intrinsic camera parameter comparison between the experimental iDIC dataset and the estimated parameters in Python. κ_n is the n^{th} radial distortion coefficient and P_n is the n^{th} tangential distortion coefficient.

Table 2 shows an extrinsic camera parameter comparison between the experimental iDIC dataset and the estimated parameters in Python.

A review of the camera parameters shows the iDIC dataset and python estimation are reasonably close. The camera model used for the iDICs dataset did not estimate skew or tangential distortion. The tangential distortion coefficients estimated in python are approximately zero, so neglecting them seems like a reasonable assumption. More work is needed however to study differences in the radial distortion parameters. An interesting observation can be made by inspecting the focal length estimates from python. The documentation for the iDICs notes that a 35mm focal length lens with used with a Basler camera having a square pixel size of 5.86 micron. Back calculating the physical unit focal length from the python estimates shows a focal length of approximately 39 mm. This likely represents the effective focal length of the lens/camera system.

Using a combination of the intrinsic and extrinsic camera parameters, a two camera scene was built in Blender.

Camera	Parameter	Dataset	Python
0	α_u	6673.3 px	6674.1 px
0	α_v	6669.3 px	6672.1 px
0	s	0 px	0 px
0	κ_1	0.032	-0.038
0	κ_2	-1.011	7.017
0	κ_3	29.788	-228.896
0	P_1	0	-3.623e^{-3}
0	P_2	0	-6.459e^{-3}
0	u_o	872.2 px	874.2 px
0	v_o	578.0 px	575.3 px
1	α_u	6607.6 px	6589.2 px
1	α_v	6602.9 px	6586.3 px
1	s	0 px	0 px
1	κ_1	0.065	-0.123
1	κ_2	-4.531	15.461
1	κ_3	29.788	-530.466
1	P_1	0	-2.912e^{-3}
1	P_2	0	-5.116e^{-3}
1	u_o	918.0 px	908.2 px
1	v_o	531.6 px	523.9 px

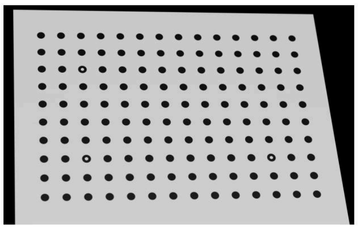
Table 1. intrinsic camera parameters

Parameter	Dataset	Python
t_X	122.2 mm	121.8 mm
t_Y	1.8 mm	1.1 mm
t_Z	17.6 mm	16.0 mm
Θ	0.13°	0.35°
Φ	-19.07°	-18.91°
Ψ	0.28°	0.09°

Table 2. extrinsic camera parameters

Figure 4a shows a rendered calibration panel image from the perspective of the left camera; while Figure 4b shows the same image from the perspective of the right camera. The panel was given a 30° tilt backward; which can be seen in the images. Now that synthetic image generation has been demonstrated, the next step is to generate an image set for calibration using the camera parameters from the iDICs dataset.

Figure 5a and 5b show that in using both synthetic and experimental images, the dot markers can be found by the python scripts. During this work it was noted that the *findCirclesGrid* method in OpenCV was insufficient alone to find all the grid points in the synthetic images, but works for the experimental images. The hypothesis is that the three hollow dots on the panel are causing the method to fail. The python script used in this work implements a combination of image thresholding and contour finding to overcome this limitation.


(a) left camera view

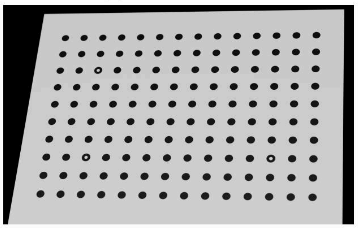
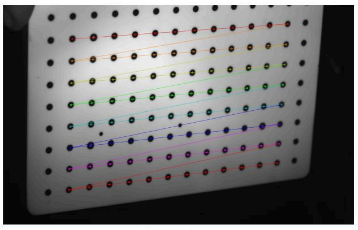

(b) right camera view

Figure 4. Synthetic calibration panel images


(a) experimental image

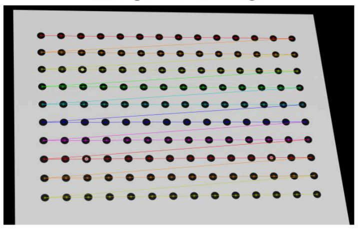
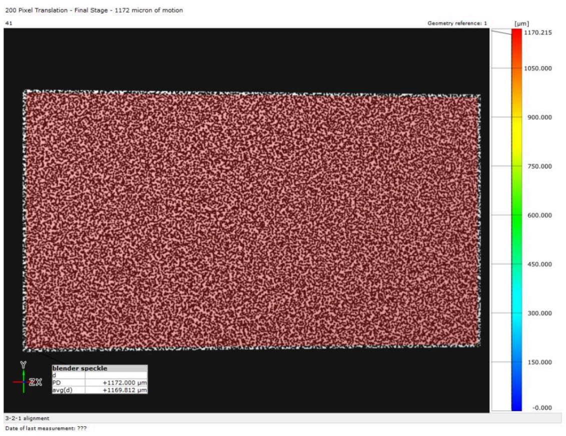

(b) synthetic image

Figure 5. OpenCV *findCirclesGrid* method

The twenty six Blender generated calibration images were successfully imported to the Zeiss DIC software. Thirteen for the left and right cameras each. The thirteen images include positions and orientations that fill the measuring volume and provide points not parallel to the image sensor plane. Three cases of simple rigid body motion were analyzed ranging from very small translations (2 pixels) to larger translations (200 pixels). Based on the camera pixel size, 2 pixels is equivalent to 11.72 micron of displacement. Eighty images (forty from each camera) were generated in Blender breaking the total translation into several incremental movements. For 2 pixels of total displacement, this would equate to 0.05 pixels of movement in each subsequent frame. This allows for the study of sub-pixel accuracy withing the Zeiss software.

Figure 6 shows a contour plot and histogram of lateral translation in the Zeiss software; confirming the images are translating as expected. The image shows the final stage and the estimated displacement is 1170 micron, close to the prescribed displacement of 1172 micron.


200 Pixel Translation - Final Stage - 1172 micron of motion

41

Geometry reference: 1

blender speckle	
d	
PD	+1172.000 µm
avg(d)	+1169.812 µm

3-2-1 alignment

Date of last measurement: ???

Figure 6. Final Stage of 200 Pixel Rigid Body Motion Translation

The plot in Figure 7 shows the the development of displacement error as the test article is displaced. Displacement error is expressed as a percentage and is calculated as the difference between the prescribed displacement in Blender and the estimated displacement from the DIC software. The error is normalized by the prescribed displacement to provide a consistent means to compare the small and large displacements.

$$\delta_{error} = \frac{\mid \delta_B - \delta_{DIC} \mid}{\delta_B} \times 100$$

Displacement errors are low; less than 1 percent and converge even for sub-pixel displacements. The 2 pixel and 20 pixel cases seem to start with similar initial errors, but the 20 pixel case converges to a final error more similar to the 200 pixel case. For context, 0.4 percent error on a 2 pixel translation corresponds to about 0.05 micron.

Literature suggests error should go to zero at integer pixel increments. In other words, error as expected to grow and reduce in a sinusoidal manner at pixel increments. More work is required to bridge the gap between behavior seen in this work and what the Rohe and Jones paper reported.

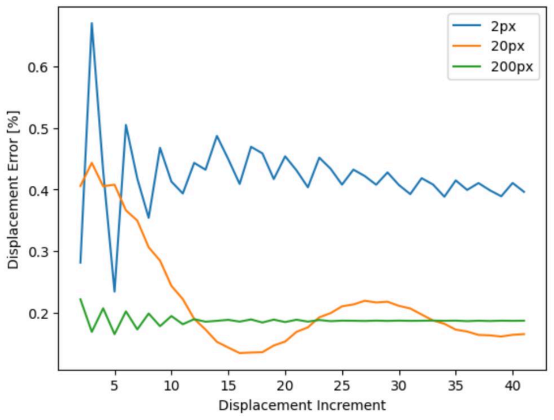


Figure 7. Development of displacement error as the test article is displaced

5. Conclusion

5.1. Milestones

The milestones listed in Table 3 were used to track progress. These milestones were utilized to make consistent and incremental progress on the project throughout the course.

Date	Milestone
04/25	Proposal Due ✓
05/03	Generation of synthetic images in Blender ✓
05/10	Completion of python stereo calibration code ✓
05/17	Milestone Due ✓
05/24	Generation of calibration file in Zeiss ✓
05/31	Completion of final results ✓
06/10	Poster Session ✓
06/12	Final Report Due ✓

Table 3. Project Milestones

5.2. Summary

A custom stereo calibration script was developed in python that enabled the verification of intrinsic and extrinsic camera parameters provided in the sourced dataset. Blender was used to create calibration and speckle images compatible with commercial DIC software. This work has shown that the free, open-source rendering software Blender can produce images that are accurate enough to be used to simulate optical DIC tests, including those with exclusively subpixel motions.

5.3. Next Steps

Moving forward, a deeper study of Blender settings including noise, lighting, scripting is required. For different render settings, First, the noise profile of the images themselves should be evaluated. Blender's python scripting capability should be investigated for automation purposes. Another next step is to utilize Blender to simulate complex strain and deformation modes as shown in Figure 8 to compare with strain calculations from the Zeiss DIC software.

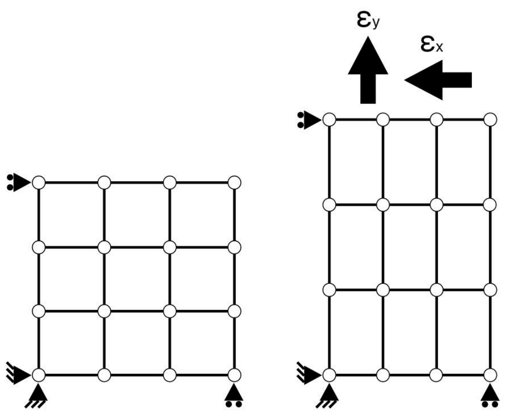


Figure 8. Undeformed and deformed Meshes

5.4. Git Repository

The python calibration script and Blender project file are located in the following repository. `https://github.com/jeremyleehill/cs231a_project_2024`

Please see the following link for more information on current and completed iDICs challenges as well as links to the dataset used in the project. `https://idics.org/challenge/`.

References

[1] M.A Sutton, J.J Orteu, and H. Schreier. *Image correlation for shape, motion and deformation measurements: basic concepts, theory and applications*. Springer Science and Business Media, 2009. 1

[2] Fabrice Pierron. Material testing 2.0: A brief review. *Strain*, 59(3), 2023. 1

[3] P. Lava, EMC Jones, L Wittevrongel, and Fabrice Pierron. Validation of finite-element models using full-field experimental data: Levelling finite-element analysis data through a digital image correlation engine. *Strain*, 56(4):1–17, 2020. 2

[4] D.P Rohe and E.M.C Jones. Generation of synthetic digital image correlation images using the open-source blender software. *Experimental Techniques*, 46:615–631, 2022. 2

[5] The Blender Foundation. https://www.blender.org, 2024. 2

[6] Kegan Kawamura, Neha Konakalla, and Sudeep Narala. 3d reconstruction in blender using stereoscopic vision, 2022. CS231a Final Project Report. 2

[7] Tomas Pivonka and Libor Preucil. Stereo camera simulation in blender. *Modelling and Simulation for Autonomous Systems*, 7th International Conference:206–216, 2020. 2