A Motion-Simulation Platform to Generate Synthetic Motion Data for Computer Vision Tasks

Andrew Chalmers
CMIC, Victoria University of Wellington
New Zealand
andrew.chalmers@vuw.ac.nz

Junhong Zhao
CMIC, Victoria University of Wellington
New Zealand
j.zhao@vuw.ac.nz

Weng Khuan Hoh
CMIC, Victoria University of Wellington
New Zealand
wengkhuan.hoh@vuw.ac.nz

James Drown
CMIC, Victoria University of Wellington
New Zealand
james.drown@vuw.ac.nz

Simon Finnie
CMIC, Victoria University of Wellington
New Zealand
simon.finnie@vuw.ac.nz

Richard Yao
Meta Platforms, Inc.
United States of America
richard.yao@fb.com

James Lin
Meta Platforms, Inc.
United States of America
james.lin@meta.com

James Wilmott
Meta Platforms, Inc.
United States of America
jwilmott@meta.com

Arindam Dey
Meta Platforms, Inc.
United States of America
aridey@meta.com

Mark Billinghurst
The University of Auckland
New Zealand
mark.billinghurst@auckland.ac.nz

Taehyun Rhee
CMIC, Victoria University of Wellington
New Zealand
taehyun.rhee@vuw.ac.nz

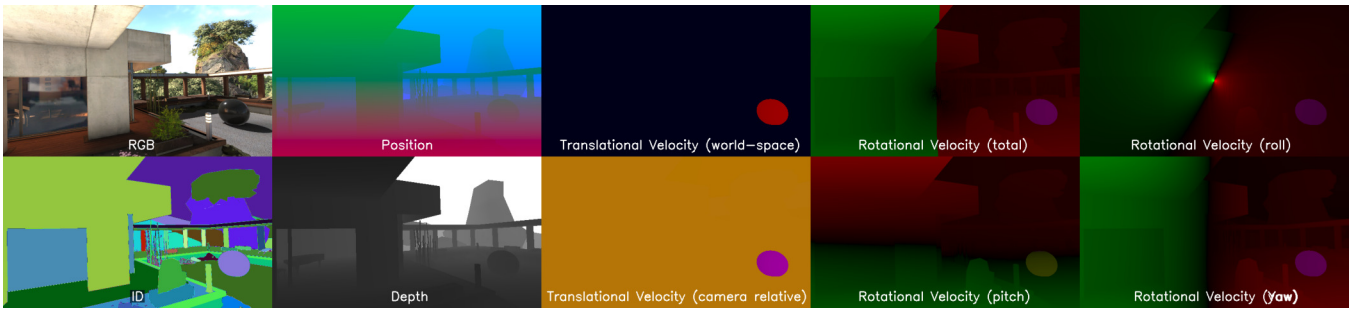


Figure 1: The data extracted from our platform: image buffer data (RGB, ID map, position map, depth map) and motion data (velocities). In the rotation velocity images, the green and red visualize a positive and negative angular value respectively. In this scene, the camera is moving forward while the environment is static except for the sphere which is moving to the right.

ABSTRACT

We developed the *Motion-Simulation Platform*, a platform running within a game engine that is able to extract both RGB imagery and the corresponding intrinsic motion data (i.e., motion field). This is useful for motion-related computer vision tasks where large amounts of intrinsic motion data are required to train a model. We describe the implementation and design details of the *Motion-Simulation Platform*. The platform is extendable, such that any scene developed within the game engine is able to take advantage of the motion data extraction tools. We also provide both user and AI-bot controlled navigation, enabling user-driven input and mass automation of motion data collection.

CCS CONCEPTS

• **Computing methodologies** → **Simulation environments**.

KEYWORDS

motion, simulation, data generation, machine learning, user study

ACM Reference Format:
Andrew Chalmers, Junhong Zhao, Weng Khuan Hoh, James Drown, Simon Finnie, Richard Yao, James Lin, James Wilmott, Arindam Dey, Mark Billinghurst, and Taehyun Rhee. 2023. A Motion-Simulation Platform to Generate Synthetic Motion Data for Computer Vision Tasks. In *SIGGRAPH Asia 2023 Technical Communications (SA Technical Communications '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3610543.3628795

SA Technical Communications '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0314-0/23/12…$15.00
https://doi.org/10.1145/3610543.3628795

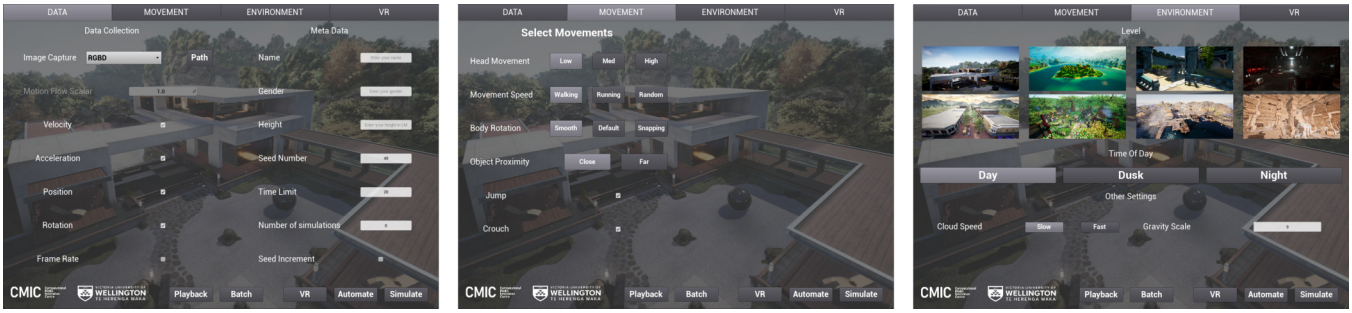


Figure 2: Settings for the *Motion-Simulation Platform*, including the data (left), movement (centre), and scene (right) settings.

1 INTRODUCTION

Intrinsic motion data (Figure 1) is crucial for various computer vision tasks, including optical flow estimation, video-based tasks (e.g., frame interpolation, video stabilization, depth estimation), object tracking, simulator sickness estimation, autonomous vehicles, and imitation learning. Collecting real-world ground truth motion data is very challenging, as it requires geometric understanding of the environment to compute the motion of a given point from one frame to the next. The use of depth sensors or object tracking is required - which is time-consuming, impractical, and error-prone due to limitations in existing hardware and calibration techniques.

Computer-generated imagery (CGI) provides a simulation-based solution, with recent advancements achieving comparable accuracy to real data for deep-learning based computer vision tasks [Wood et al. 2021]. The data will be error-free, support automation, reduce labor, and can be re-run, modified, and adapted to different computer vision tasks. However, prior simulation methods have not emphasized motion data generation.

We introduce our *Motion-Simulation Platform* deployed in Unreal Engine, capable of recording and extracting motion data from a given scene. The platform's extendability allows large-scale motion data generation, beneficial as features for various computer vision tasks to enhance inference quality. We detail the implementation of motion data generation and describe an AI-bot for autonomous navigation, enabling automatic mass data generation. Some computer vision tasks also deal with user-controlled input, as such, we also describe the setup for user input support with replay functionality to reproduce the user's experience.

2 RELATED WORK

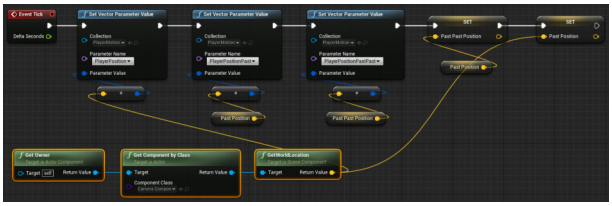
Real-world data collection requires painstaking manual effort, including hardware setup and participant involvement [Lin et al. 2014]. Examples include targetting people [Liu et al. 2015], handwritten digits [Deng 2012], speech [Panayotov et al. 2015], autonomous vehicles [Janai et al. 2020], environmental lighting [Gardner et al. 2017], and many others. These datasets are fixed and are challenging to expand upon. Augmentation strategies (e.g., rotation and scale) are often used to expand the dataset, but cannot greatly improve the variability of the dataset as the underlying data is 2D and rasterized [Sun et al. 2021]. Furthermore, ground truth underlying intrinsic information, such as motion data, is often not available. To avoid such limitations, CGI can be used to generate large amounts of data. Previous work has focused on different types of data for specific application areas [Dowrick et al. 2022; Tremblay et al. 2022; Wang et al. [n. d.]; Wood et al. 2021], while we focus on producing motion data - which is applicable for many computer vision tasks. The use of motion data can be as part of the input to the network (e.g., VR sickness [Hu et al. 2019]), as part of the loss function (e.g., video frame interpolation [Huang et al. 2020]), or as part of the output (e.g., optical flow estimation [Dosovitskiy et al. 2015]). Our simulation platform also supports real user input for those motion-related tasks that are related to user input (e.g., VR sickness [Hu et al. 2019] and imitation learning [Argus et al. 2020]). Unlike previous 3D CGI datasets which operate on intercepting graphics data on existing games [Richter et al. 2016; Shafaei et al. 2016; Wen et al. 2022], we are able to clearly demonstrate extracting motion data since our method operates within the game engine. Working within the game engine allows our method to be extensible and modifiable, where prior work [Mueller et al. 2017] have not described or demonstrated motion flow extraction.

3 MOTION-SIMULATION PLATFORM

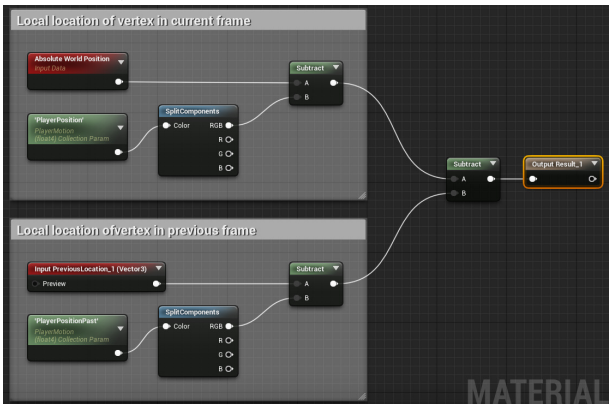
The *Motion-Simulation Platform* consists of three main components: *Scenes*, methods of *navigation*, and a *replay system*. These components enable gameplay simulation and support various computer vision tasks. The platform's user interface (Figure 2) allows flexible customization of settings, supporting manual, sequential, or random selection for user-controlled and automated data collection. The motion-data generation functionality (described in the following section) adapts to the chosen settings, providing flexibility for different computer vision tasks.

For *scenes*, the platform comprises a diverse variety to extract motion data, where additional scenes can be added. Scene selection can be manual, iterative, or randomized, with options to modify attributes like time of day, cloud speed, and gravity.

For *navigation*, this can be AI-bot based or user-controlled. The *AI Character Controller* emulates user behaviors (e.g., rotation, head movement, running) for AI-bot navigation. To mimic real user motion, we optionally add natural head sway animation (captured from pre-motion data) to the virtual camera, with customizable sway intensity. AI-bot navigation incorporates a randomization layer to ensure visual and motion variety, vital for computer vision tasks. The AI-bot spawns at predefined starting positions and navigates to



(a) Storing the world position variable for the next frame.



(b) Difference in position from the previous to the current frame.

Figure 3: Calculating position and velocity in the Unreal Engine material blueprint system.

randomly generated destinations using a *'NavMesh'*. Various movement behaviors (translation speed, head movement speed, body rotation speed, jumping, crouching) can be manually, iteratively, or randomly selected. For user-controlled data collection, we support multiple input methods, including keyboard/mouse, gamepad, and virtual reality (VR) controllers. The behavior settings can limit user functionality, e.g., disabling/ enabling running or jumping.

For the *replay system*, our platform supports the ability to record and replay any AI or user playthrough. Replaying data is useful for retrospection, data analysis, or generating additional data from previous play-throughs. Since the replay is performed "offline" (i.e., users' are not playing live), any additional data generation methods applied in post, aren't restricted by real-time computational requirements. To support the replay system for the AI-bot, we record the random seed and replay the AI-bot navigation from it. For the user-controlled input, we developed a recording system that tracks the transformation matrix of the camera, game objects, and light sources during the users' play-through. These recordings reproduce the exact game state as experienced by the user.

4 MOTION DATA COMPUTATION

We export data for every frame during the gameplay simulation, including six different forms of motion data, as follows. World-space translational velocity:

$$\mathrm{T}_w = O_t - O_{t-1}, \tag{1}$$

camera-relative translational velocity:

$$\mathrm{T}_c = T_t - T_{t-1}, \tag{2}$$

camera-relative total rotational velocity:

$$\mathrm{R}_{total} = \cos^{-1}(N_t \cdot N_{t-1}), \tag{3}$$

and camera-relative decomposed (yaw, pitch, roll) rotational velocity:

$$\mathrm{R}_{yaw} = \cos^{-1}(Nx_t \cdot Nx_{t-1} + Ny_t \cdot Ny_{t-1}), \tag{4}$$

$$\mathrm{R}_{pitch} = \cos^{-1}(Nx_t \cdot Nx_{t-1} + Nz_t \cdot Nz_{t-1}), \tag{5}$$

$$\mathrm{R}_{roll} = \cos^{-1}(Ny_t \cdot Ny_{t-1} + Nz_t \cdot Nz_{t-1}), \tag{6}$$

where O is a point on an object, T and N are unnormalized and normalized direction vectors from the camera to O, and t denotes the frame number. Note that Unreal Engine uses a right-handed coordinate system where z is up. The translational velocity equations capture the movement of the objects within the scene (either in world space, or relative to the camera), while the rotational equations capture the apparent angular motion of the object as observed by the camera. We decompose the rotation into yaw, pitch, and roll as it is often useful to measure each component separately. With these equations, we are able to measure not only the movement of the camera but also that of any dynamic object moving within the scene. The core operation for the equations is to obtain the position of the surface at the current and previous frame. To do this, we utilize Unreal Engine's *'World Position'* variable in the Blueprint system [Unreal Engine 2022] (Figure 3).

In addition to the motion data, we also export the RGB frame data as observed by the camera, the depth buffer, as well as the world-space position of the point on the object at each pixel. An object ID map is also exported to link the per-pixel information back to the scene objects. See Figure 1 for a visualization of all the exported data. All images are exported as 16-bit 4-channel raster images in the OpenEXR format [Kainz et al. 2009], except the ID map which is comprised of unsigned integers. The ID map is particularly useful for data analysis or data modification, as it gives an idea of what parts of the scene impact the computer vision task.

5 EVALUATION

To show that the *Motion-Simulation Platform* extracts the motion data reliably, we perform both a qualitative and ablation study.

The qualitative results are shown in Figure 1. This scene includes a camera moving forward with a dynamically moving sphere. We can observe that the translational velocity in world space is entirely black except for the moving sphere, as expected. The camera relative translational velocity adds the constant motion across all pixels due to the camera motion. For the rotation, the yaw, pitch, and roll angular motion moves in the expected direction, where the green and red pixels indicate a positive and negative angular change respectively. The total rotational velocity radially increases from the centre to the periphery, where there is the most observable change in motion. We show the results in motion in the supplemental video.

The ablation study uses the motion features in practice. As an example, we used a simple U-Net convolutional neural network that is designed to estimate depth maps from RGB video. We used our *Motion-Simulation Platform* to generate the video sequence with ground truth depth maps. The base network will estimate a depth map given an RGB frame of video at time t, where the loss is the

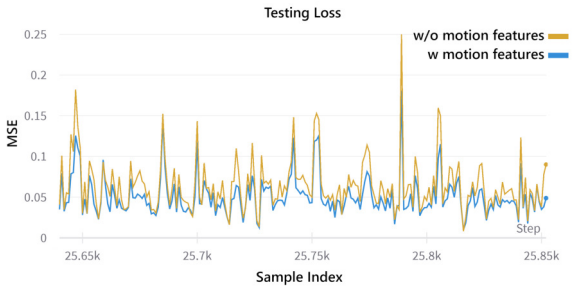


Figure 4: Ablation study results on a computer vision task. As an example, this task aims to estimate depth from RGB video - without (orange) and with (blue) motion data.

(a) Ground truth (b) w/o motion data (c) w motion data

Figure 5: A sample from the test set of the ablation study, including the (a) ground truth depth map, and the estimated depth maps (b) without and (c) with the motion data.

ℓ_2-norm between the estimated and ground truth depth map. The network using the motion data is also a U-Net, and will additionally include the two previous frames at $t-1$ and $t-2$, and the loss also includes the ℓ_2-norm between motion data in addition to the depth map estimate. We trained the network on 10260 samples, and tested on 3392 samples. The overall MSE is 0.09096 and 0.04881 for the network without and with the motion data respectively. The error for each sample is visualized in Figure 4, and an example of the estimated depth map is shown in Figure 5. As you can see, the motion data from the *Motion-Simulation Platform* is reliable and offers a clear benefit to improving the quality of computer vision tasks, such as depth map estimation from RGB video.

6 CONCLUSION

In this paper, we presented the *Motion-Simulation Platform*, a platform that provides researchers with the ability to generate motion data for motion-related computer vision tasks. We described implementation details including the motion-related equations, scene integration, level navigation, and a replay system. The platform is designed to be flexible so that data can be generated and exported from any scene (e.g., free scenes from online repositories, in-house scenes, scenes designed for user studies, etc.). The platform settings make it easy to customize for different computer vision tasks, as well as the ability to target mass automation and user-driven input.

Limitations: The motion data is based on computing differences in positions for objects with geometry. As such, the motion of particle effects will not be captured. For the AI-bot to sensibly navigate the environment, a NavMesh needs to be manually created for the level - future work can consider integrating an automated NavMesh generator. We hope to see researchers expand upon the features of the *Motion-Simulation Platform* to be suitable for more tasks. For example, we focused on human-like behaviors, but our underlying motion data could be adapted to other forms of navigation, such as vehicles.

REFERENCES

M Argus, L Hermann, J Long, and T Brox. 2020. FlowControl: Optical Flow Based Visual Servoing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 7534–7541. https://doi.org/10.1109/IROS45743.2020.9340942

Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.

Thomas Dowrick, Brian Davidson, Kurinchi Gurusamy, and Matthew J Clarkson. 2022. Large scale simulation of labeled intraoperative scenes in unity. *International Journal of Computer Assisted Radiology and Surgery* 17, 5 (2022), 961–963.

Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. 2017. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090* (2017).

Ping Hu, Qi Sun, Piotr Didyk, Li-Yi Wei, and Arie E Kaufman. 2019. Reducing simulator sickness with perceptual camera control. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2020. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294* (2020).

Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. 2020. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision* 12, 1–3 (2020), 1–308.

Florian Kainz, Rod Bogart, and Piotr Stanczyk. 2009. Technical introduction to OpenEXR. *Industrial light and magic* 21 (2009).

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

M Mueller, V Casser, J Lahoud, N Smith, and B Ghanem. 2017. Ue4sim: A photo-realistic simulator for computer vision applications. CoRR, abs/1708.05869. *arXiv preprint arXiv:1708.05869* (2017).

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964

Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 102–118.

Alireza Shafaei, James J Little, and Mark Schmidt. 2016. Play and learn: Using video games to train computer vision models. *arXiv preprint arXiv:1608.01745* (2016).

Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. 2021. Autoflow: Learning a better training set for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10093–10102.

Jonathan Tremblay, Moustafa Meshry, Alex Evans, Jan Kautz, Alexander Keller, Sameh Khamis, Charles Loop, Nathan Morrical, Koki Nagano, and Towaki Takikawa. 2022. RTMV: A Ray-Traced Multi-View Synthetic Dataset for Novel View Synthesis. *arXiv preprint arXiv:2205.07058* (2022).

Unreal Engine. 2022. Coordinates Material Expressions: World Position.

Cheng Yao Wang, Eyal Ofek, Daniel McDuff, Oron Nir, Sai Vemprala, Ashish Kapoor, and Mar Gonzalez-Franco. [n. d.]. CityLifeSim: A High-Fidelity Pedestrian and Vehicle Simulation with Complex Behaviors. ([n. d.]).

Elliott Wen, Tharindu Indrajith Kaluarachchi, Shamane Siriwardhana, Vanessa Tang, Mark Billinghurst, Robert W Lindeman, Richard Yao, James Lin, and Suranga Nanayakkara. 2022. VRhook: A Data Collection Tool for VR Motion Sickness Research. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–9.

Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. 2021. Fake it till you make it: face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF international conference on computer vision*. 3681–3691.