Autoferry Gemini: a real-time simulation platform for electromagnetic radiation sensors on autonomous ships

K Vasstein, E F Brekke, R Mester and E Eide
Norwegian University of Science and Technology, Høgskoleringen 1, 7491 Trondheim, Norway
E-mail: kjetv@stud.ntnu.no, edmund.brekke@ntnu.no, rudolf.mester@ntnu.no, egil.eide@ntnu.no

Abstract. Testing that ships are compliant to specified safety requirements have traditionally relied on real world data, which is not scalable and limited to testable scenarios due to financial and ethical reasons. Low fidelity simulations have been used to counteract some of these problems, which is sufficient for emulating simpler systems such as radar detectors, but not for testing complex systems as found in computer vision. In the automotive industry the use of game engines have shown to be a great testing platform due to their customizability, and combination of real-time physics with computer graphics to create large volumes of high fidelity images. In the work presented here, development of an open-source maritime platform named Autoferry Gemini based on the Unity game engine is used to simulate sensors in real-time. Utilizing simulated optics and general purpose GPU programs, the render pipeline is capable of modeling lidar, radar, visible-light and infrared camera sensors simultaneously. Results from visible-light cameras and lidar have already proven to satisfy other research activities on sensor fusion for autonomous ship technology. Infrared cameras motivates further research in gathering empirical data, while GPU algorithms have made it possible to simulate 3D radar models and multiple lidar types in real-time.

1. Introduction
Traditionally, testing control and navigational systems for ships have to a large extend been done through real world testings. In addition, such systems have relied on being operated or supervised by human pilots, thus simple systems have been sufficient. Due to this simplicity, the use of idealized or analytical mathematical models for simulating control and navigation systems, have been a good substitution for real testings. With the development towards autonomous ships, these systems have in contrary become more complex due to the introduction of probabilistic models and other methods from the field of artificial intelligence (AI), such that traditional simulation techniques is less capable of determining if the control and navigation is safe, stable and predictable. In addition, advancement in fields such as machine learning, computer vision and the integration of these with conventional model based control and navigation, such as optimal-control, target tracking, localisation and mapping, requires data from other sensors than just IMU, GPS, and simple radar detectors. This includes sensors with high data rates such as lidars, but also more fidelity sensitive sensors such as visible-light (VL) and infrared (IR) cameras. This have led to an increasing demand for sensor data with high quality and volume. As real world sensor data from both cars and ships have proven to be costly, moreover difficult to receive for research outside companies, there is a demand for high fidelity simulations.

High fidelity video games, which goes by the nickname AAA-Games have traditionally used game engines developed in-house. However, in the last decade multiple third party game engines have enabled
several simulators such as AirSim created with the Unreal Engine [1] and LGSVL created with Unity [2] to simulate sensors for both drones and cars. Such game engines have also been used in the maritime industry, primarily for training and educating human operators in a real-time environment [3]. Among these, the Offshore simulation center (OSC) stationed in Ålesund are the best known high fidelity maritime simulator to the author. In addition for being used in education and training, it have been used for research on human behaviour in offshore operations [4, 5], control designs related to cranes operating in difficult environments [6] and testing multi-sensor systems for enhancing situational awareness (SITAW) [7]. Unfortunately OSC’s simulator is not open source, and since it focuses on VL cameras, other sensors of maritime interest such as IR, lidar and radar have received little attention.

Simulators for IR images have primarily focused on modeling IR materials on specific objects such as aircraft [8] and oceans [9], while various video games have in contrast relied on artistic implementations to emulate IR textures. More recent studies have shown results of generalizing IR for various scene objects with the use of GPU shaders [10] and examples can be seen in game engines [11] that have previously only used shaders for VL simulations.

Simulating VL cameras, have been the primary reason for the use of game engines in simulators such as Carla [12], LGSVL [2] and AirSim [1]. LG’s automotive simulator [2] was early to adapt Unity’s new high definition render pipeline (HDRP) which demonstrated its high fidelity capabilities for cameras on autonomous vehicles, and the Unreal engine have been well known for its fidelity in the AAA-game marked for over a decade. The use of these simulators for autonomous control have been vast, but mainly focused on computer vision techniques such as Visual Odometry [13].

Lidar simulations using GPU was also demonstrated in LGSVL at the end of 2018, using depth-buffers from Unity’s older built in render pipeline (BRP) to simulate dense point clouds from virtual lidars running in real-time. Previous research on virtual lidars [14] have also been known to use depth-buffers, which are among the two most common depth techniques used in render pipelines, dating back to 1974 [15]. Ongoing research focuses on enhancing the lidars fidelity by introducing effects such as reflection, refraction, clutter and raydrops with conventional modeling techniques [16] and AI [17].

Radar simulations using game engines are to the author not known, except for simple collision sensors for cars seen in various simulators [1, 12, 2]. The vast selection of different radar types have probably not made the task any easier, such as the need for dedicated hardware to simulate synthetic aperture radars [18]. For conventional long range radars for ships, there are however non real-time solutions such as Carpet [19], that functions as a separate radar program capable of handling atmospheric disturbances, Doppler and polarization effects, antenna radiation pattern etc.

In a student project at NTNU during the spring of 2019, implementation of several EMR sensors was done using a CPU based lidar with raycasts from Unity’s physics engine, and virtual VL camera from Unity’s BRP [20]. This was done for the autonomous ferry Milliamper, which gave a demonstration of the possibility of simulating sensors in a virtual environment. The study showed the possibility to run virtual sensors simultaneously, with relatively high fidelity, in a maritime environment considering hydrodynamic forces, vessel control and human interaction. This have since been used in research regarding situational awareness, such as feature-based pose estimation of ships in monocular camera images [21], which demonstrated the use of data from game engines for testing computer vision on ships.

The main contribution of this paper is to present the open source maritime simulation platform Autoferry Gemini, with the sensor models for IR, radar, VL and lidar. The paper builds upon the student projects [20, 22], which can be obtained from contacting the first author. By utilizing the Unity game engine, HDRP is used to model IR and VL sensors, while depth-buffers and GP-GPU algorithms are used to simulate lidar and radar. We believe from the mentioned use cases [13, 7, 6, 21] that this can give researchers a new platform for obtaining EMR sensor data from virtual marine vessels that have previously been spread across multiple platforms. As a summary, this entails:

• Creating an open-source maritime simulator emphasising autonomous ships.
• Concurrent simulation of radar, lidar, IR and VL sensors.
• Real-time simulation by utilizing the GPU for sensor modelling.
2. Sensor simulation methods
The core idea behind our method is to utilize Unity’s real-time render pipeline to generate data. By configuring it with the use of command buffers [22], creating several separate GPU data pipelines based on virtual cameras graphic-buffers (G-buffers), which contains information of surface normals, materials and depths from the deferred shading technique [23], one guarantees that all of the sensor data comes from the same state. This also makes it possible to run all of the sensors concurrently, as Unity schedules the pipelines running on the GPU.

![Figure 1. GPU data pipeline using HDRP to simulate VL and IR, and a custom pipeline to simulate radar and lidar. Blue boxes cover the implemented work, red the resulting EMR sensors, while purple is implemented by Unity.](image)

2.1. HDRP
Unity’s HDRP are used to implement the VL and IR camera sensors. Unity’s Lit shader are used to display most of the surfaces for the VL sensor by utilizing physically based rendering (PBR) materials from third party sources. Custom shaders are used for the ocean created with HDRP’s Shader Graph tool, while clouds are created using particles from the Visual Effects Graph (VFX).

A culling mask is created by using Unity’s layer system, in order to create the separate pipeline for the IR camera shown in Figure 1. In addition, Shader Graph is used to implement the thermal radiation from Stefan-Boltzmann law [24] with an emissivity texture $\epsilon$ which gives the render pipeline the emissivity at different surface locations on a 3D model:

$$W(T) = \epsilon \sigma T^4.$$  \[W \text{ m}^{-2}\]  \(1\)

Due to the lack of empirical data, artistic emissivity textures are used. Assuming that the 3D models are close to the ambient temperature $T \approx 300K$, one would get from Wiens’s displacement law [24] that the radiations peak wavelength $\lambda_{peak} = \frac{h}{\pi} \approx 9.7\mu m$ is in the infrared spectrum. Due to this, Stefan-Boltzmann law becomes a good first order approximation, since close to ambient temperature ranges, infrared imaging devices are classified as total-radiation radiometers [25], where 98% of the total black body radiation described by Planck’s law are in the infrared spectrum.
2.2. Depth-Buffer to Point Cloud

A camera’s depth-buffer is fetched from the G-buffer with the use of a command buffer from a custom pass class in Unity seen in Figure 1. This is sent to a sensor script that dispatches the depth-buffer along with other sensor parameters such as the sensor’s max target distance, resolutions, etc, to a compute shader that creates a point cloud using the GPU.

The depth-buffer for camera \( l \) is defined as a 2D pixel array \( Z^l_{nx,ny} \in \mathbb{R} \) of normalized values \( 0 \leq Z^l_{nx,ny} \leq 1 \), and pixel positions \((n^l_{nx}, n^l_{ny}) \in \mathbb{N}^{H \times W} \) where \( H \) and \( W \) are the depth-buffer’s height and width. From this, homogeneous coordinates in image space can be written as:

\[
\bar{X}^i,l = [\bar{n}^l_{nx}, \bar{n}^l_{ny}, Z^l_{nx,ny}, 1]^T
\]  

(2)

where \( \bar{n}^l_{nx} = \frac{2n^l_{nx}}{W} - 1 \), \( \bar{n}^l_{ny} = \frac{2n^l_{ny}}{H} - 1 \) places origo at the depth-buffers center and remaps the pixel coordinates to be values between \((-1, 1)\). To convert from image space coordinates \( \bar{X}^i,l \) to the camera’s view space coordinates \( \bar{X}^v,l \), a projection matrix \( P^v_i \) with parameters for the virtual cameras far plane \( f \), near plane \( n \) and horizontal field of view \( \text{FOV}_h \) seen in Figure 2 is used:

\[
P^v_i := \begin{bmatrix}
cot(\frac{\text{FOV}_h}{2}) & 0 & 0 & 0 \\
0 & \cot(\frac{\text{FOV}_h}{2}) & 0 & 0 \\
0 & 0 & \frac{f + n}{f - n} & -2 \frac{f \cdot n}{f - n} \\
0 & 0 & -1 & 0 \\
\end{bmatrix}^{-1}
\]  

(3)

\[
\bar{X}^v,l = P^v_i \bar{X}^i,l
\]  

(4)

Figure 2. Depth-buffers from virtual cameras used to approximate cylindrical beams with a convex regular polygon. Stitching the depth-buffers together, forms a 2D depth array.
IOP Conf. Series: Materials Science and Engineering 929 (2020) 012032   doi:10.1088/1757-899X/929/1/012032

The results in a circular range equation:

$$X^{\nu;l} = \frac{1}{w^{\nu;l}} \tilde{X}^{\nu;l}. \quad (5)$$

The last equation divides all elements by $\tilde{X}^{\nu;l}$ last element such that it becomes 1 as in (2). From this, a range $R_l(n_x, n_y)$ defined as the distance between camera $l$’s center to an object’s rasterized surface in view space with respect to the depth-buffers pixel positions, can be created from (2)–(5):

$$R_l(n_x, n_y) := \|X^{\nu;l}\|_2 = \left\| \frac{1}{w^{\nu;l}} P_i^\nu \left[ 2n_x W - 1, 2n_y H - 1, Z_l(n_x, n_y), 1 \right]^T \right\|_2. \quad (6)$$

In order to approximate a circular beam, $L$ cameras is spaced around the sensor’s azimuth-axis with vertical field of view $\text{FOV}_v = \frac{2\pi}{L}$ as seen in Figure 2. This creates $L$ depth-buffers $Z_l(n_x, n_y)$ that can be stitched to a 2D depth array $Z(n_x, n_y)$ seen in Figure 2:

$$l := n_x \backslash W \quad (7)$$

$$Z(n_x, n_y) := Z_l(n_x, n_y) \quad (8)$$

where $0 \leq n_x < L \cdot W$ substitutes the camera dependent pixel coordinate $n_x$ seen in Figure 2, and “\" is defined as the integer division. Since each camera is spaced around the azimuth-axis, they have a rotation matrix $R^l$ relative to the first camera which transforms each cameras view space coordinates $X^{\nu;l}$ to the first camera $X^\nu := X^{\nu;0}$:

$$R^l := R_y(\text{FOV}_v \cdot l) := \begin{bmatrix} \cos\left(\frac{2\pi l}{L}\right) & 0 & \sin\left(\frac{2\pi l}{L}\right) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\left(\frac{2\pi l}{L}\right) & 0 & \cos\left(\frac{2\pi l}{L}\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$X^\nu = R^l X^{\nu;l}. \quad (10)$$

This results in a circular range equation:

$$R(n_x, n_y) = \|X^\nu\|_2 = \|R^l X^{\nu;l}\|_2 = \|R^l\|_2 \|X^{\nu;l}\|_2 = \|R^l\|_2 R^l(n_x, n_y)$$

$$= \left\| \frac{1}{w^{\nu;l}} P_i^\nu \left[ 2n_x W - 1, 2n_y H - 1, Z_l(n_x, n_y), 1 \right]^T \right\|_2$$

$$= \left\| \frac{1}{w^{\nu;l}} R^l P_i^\nu \left[ 2n_x W - 1, 2n_y H - 1, Z(n_x, n_y), 1 \right]^T \right\|_2. \quad (11)$$

Spherical coordinates can be transformed from pixel coordinates in the following manner:

$$\theta = \frac{2\pi n_x}{L \cdot W}, \quad \phi = \text{FOV}_v \left( \frac{2n_y}{H} - 1 \right) \quad (12)$$

$\theta$ is the azimuth angle: $0 \leq \theta < 2\pi$,

$\phi$ is the elevation angle: $-\text{FOV}_v \frac{2}{L} \leq \phi < \text{FOV}_v \frac{2}{L}$.

$\text{FOV}_v$ is the sensor’s vertical field of view.
2.3. Radar Spokes Equation

A single radar spoke, which contains cells of power at different target ranges, can be described with the radar range equation for an antenna used for both transmitting and receiving [26]:

\[ P_r = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \]  

with radar wavelength \( \lambda \), antenna gain \( G \), power \( P_t \), target range \( R \), and radar cross section area defined as \( \sigma = \lim_{x \to \infty} 4\pi R^2 \frac{E_i}{E} \) from Maxwell’s equations [26].

To simplify the problem, we assume \( \sigma = 1 \) and use the range \( R(n_x, n_y) \) from (11). The antenna gain is created by introducing a normalized radiation pattern \( G(n_x, n_y) \) with an amplitude \( |G_{\text{max}}| \) such that \( G = |G_{\text{max}}|G(n_x, n_y) \), where we create a single main lobe as an ellipse with the height spanning the maximum of the main lobe. With these representations (13) becomes:

\[ P_r(n_x, n_y) = \frac{P_t \lambda^2 |G_{\text{max}}|^2 G(n_x, n_y)^2}{(4\pi)^3 R(n_x, n_y)^4}. \]  

To create a spoke, we bin this power by introducing the spoke range \( r \in \mathbb{N} \) with a resolution of \( r_{\text{max}} \):

\[ r = \frac{R(n_x, n_y)}{f} r_{\text{max}} \]  

where \( f \) is the far plane distance from (3) and \( r \) rounds to the closest integer. Any range larger than \( r_{\text{max}} \) due to the beam-shape error is dropped. With this, we marginalize (14) to form a single spoke equation that takes into account the 3D surroundings:

\[ P_{\text{spoke}}(r) = \frac{P_t \lambda^2 |G_{\text{max}}|^2}{(4\pi)^3} \sum_{n_x=0}^{L} \sum_{n_y=0}^{H} G(n_x, n_y)^2 \frac{R(n_x, n_y)^4}{R(n_x, n_y)^4}. \]  

As typical ship radars have multiple spokes arranged around the azimuth-axis, a finite number of spokes \( h \) is sampled from the total sweep resolution \( h = \frac{LW}{n}, n \in \mathbb{N} \), with a spoke index \( 0 \leq k < h \in \mathbb{N} \), resulting in \( h \) spokes displaced around the radar by offsetting the radiation pattern:

\[ P_{\text{spokes}}(r, k) = \frac{P_t \lambda^2 |G_{\text{max}}|^2}{(4\pi)^3} \sum_{n_x=0}^{L} \sum_{n_y=0}^{H} G(n_x + k \cdot \frac{LW}{h}, n_y)^2 \frac{R(n_x, n_y)^4}{R(n_x, n_y)^4}. \]  

A first order approximation of the radar range resolution is done by taking the radar bandwidth \( B \) into account. This is done by convolving a discrete Gaussian function \( f(x) \) with \( P_{\text{spokes}}(r, k) \), creating a finite summation with window size \( 2M \), where \( M = 50 \) turned out to be sufficient for experiments:

\[ f(x) = \frac{1}{\sqrt{2\pi s^2}} e^{-\frac{x^2}{2s^2}}, \quad s^2 = \frac{c}{2B} \]  

\[ P_{\text{spokes}}(r, k) = (P_{\text{spokes}} \ast f)(r) = \sum_{m=-M}^{M} f(r - m) P_{\text{spokes}}(r, k) \]  

where \( c \) is the speed of light in a vacuum. Combining (17) - (19) the radar spokes equation with radar range uncertainty becomes:

\[ P_{\text{spokes}}(r, k) = \frac{P_t \lambda^2 |G_{\text{max}}|^2}{(4\pi)^3} \sum_{m=-M}^{M} e^{-\frac{(r-m)^2}{\frac{c}{2B}^2}} \sum_{n_x=0}^{L} \sum_{n_y=0}^{H} G(n_x + k \cdot \frac{LW}{h}, n_y)^2 \frac{R(n_x, n_y)^4}{R(n_x, n_y)^4}. \]  

For implementation, (15) and (17) is used to create the first compute shader kernel, while a second kernel uses (19) to compute the final radar spokes results from (20).
2.4. Beam-Shape Error

The use of $L$ depth-buffers creates a convex regular polygon beam that approximates a circle as seen in Figure 2. This creates an error as a function of both distance and angle relative to the depth-buffers center for a camera $l$:

$$
\epsilon(\alpha, d) = d - d = \frac{d}{\cos(\alpha)} - d = d \left( \frac{1}{\cos(\alpha)} - 1 \right)
$$

From Figure 2, it can be seen that the maximum error is located where the depth-buffers meet: $\alpha_{\text{max}} = \frac{\text{FOV}}{2} = \frac{\pi}{L}$. Distance $d$'s largest value is the virtual cameras far plane $f$, giving a maximum beam shape error:

$$
\epsilon_{\text{max}} = f \left( \frac{1}{\cos(\frac{\pi}{L})} - 1 \right).
$$

Further, an average beam shape error with respect to distance can be calculated as:

$$
\epsilon_{\text{avg}}(d) = \frac{1}{\alpha_{\text{max}}} \int_{0}^{\alpha_{\text{max}}} \epsilon(\alpha, d) d\alpha = \frac{L}{\pi} \int_{0}^{\frac{\pi}{L}} d \left( \frac{1}{\cos(\alpha)} - 1 \right) d\alpha
$$

and the final average beam shape error becomes:

$$
\epsilon_{\text{avg}} = \frac{1}{f} \int_{0}^{f} \epsilon(d) \, d = \frac{L}{f} \ln \left( \frac{1 + \sin(\frac{\pi}{L})}{\cos(\frac{\pi}{L})} \right) - 1 \left[ \frac{d^2}{2} \right]_{0}^{f}
$$

3. Evaluation

Sensor parameters used to obtain the results are based on specifications from a Simrad 4G for radar, Velodyne VLP-16 for lidar, while IR and VL are chosen arbitrarily, if not other is specified. As studying how the sensor outputs could be transferable to the real-world was beyond the scope of the paper, only a qualitative evaluation is made.

The results came from a laptop with the following specifications:

- **OS**: Windows 10 Home, 64-bit
- **CPU**: Intel(R) Core(TM) i7-8565U, X64 Quad Core CPU
- **Memory**: 16 GB RAM, 512 GB SSD
- **GPU**: NVIDIA GeForce GTX 1050, 4 GB GDDR5 VRAM
Table 1. Virtual camera parameters.

| Sensors | W [pixels] | H [pixels] | L | FOV_h [°] | FOV_v [°] | f [m] | n [m] |
|---------|------------|------------|---|-----------|-----------|-------|-------|
| VL      | 1920       | 1080       | 1 | 50        | 90        | 10^5  | 10^{-1} |
| IR      | 1920       | 1080       | 1 | 34        | 60        | 10^5  | 10^{-1} |
| Lidar   | 64         | 16         | 32 | 11.25     | 30        | 100   | 1     |
| Radar   | 64         | 128        | 16 | 22.5      | 20        | 100   | 1     |

Table 2. Additional radar parameters.

| P_t [W] | λ [m] | B [MHz] | G_{max} [dBi] | M | h  | r_{max} | g [°] |
|---------|-------|---------|-------------|---|----|---------|------|
| 20      | λ     | 50      | 1           | 50| 512| 1024    | 5    |

3.1. Results compared to real sensor data

Figure 3. Simulated VL camera of Milliampere 2.0 outside Trondheim to the left, and real VL image of Milliampere 1.0 at Ravnkloa in Trondheim.

High fidelity results for the VL camera with physical correct lightning such as ocean reflection, sub surface scattering in the clouds and colour temperature relative to the time of day, can be seen from Figure 3. Comparing the real and synthetic data for such high quality built scenes, there is not much to remark. A more noticeable effect was discovered by changing the scene content with low resolution models and materials, which resulted in lower fidelity. This could be seen at Milliampere’s operational area at Ravnkloa, where to little details resulted in an unrealistic look, even though the city model of Trondheim seemed realistic at far distances.

Similar observations was made for the IR camera, which had no empirical based emissivity textures to emulate real IR images, which can clearly be seen by comparing the human faces in Figure 4. Lightning effects such as glow and blurriness did however reassemble real IR photographs.

As for lidar, experiments showed the ability to simulate data rates up to 30 million points per second while still having 50 fps as from Table 3. Because of this, it was possible to emulate multiple lidar types between 16-128 lasers, one of which is demonstrated in Figure 5. Using depth-buffers did however introduce a beam shape error which created a trade-off between accuracy and performance seen in Figure 7. Synchronisation errors was also discovered while experimenting with multi-sensor simulations, but no attempts where made to solve it.
Figure 4. Simulated IR camera of a human outside Trondheim using artistic textures to the left, and real IR image from Milliampere 1.0 with a human in the bottom right corner.

Figure 5. Simulated results from the Old Seaport model of a VLP-16 to the left, and real data from [27] to the right.

Figure 6. Simulated radar plot for Simrad 4G at Ravnkloa to the left, with corresponding real radar data to the right. All plots uses a 100m radar range.

Modeling radars using depth-buffers, showed that 3D information from both the radiation pattern and the area of operation could be considered. However, results such as from Figure 6 showed more details in the area than in the real world. Constructing the virtual operational area with sufficient content as in reality, was also an issue. Testing side and back lobes in the radiation pattern created mirrored patterns as in real life, but no further observations towards its fidelity was be made.
3.2. Errors and real-time performance

![Image of observable beam shape error in the Old Seaport model for lidar in the left images, top: L = 4 and bottom: L = 16 cameras. Graph to the right depicts the trade-off between error and real-time performance sampled from using different camera numbers.]

**Figure 7.** Observable beam shape error in the Old Seaport model for lidar in the left images, top: L = 4 and bottom: L = 16 cameras. Graph to the right depicts the trade-off between error and real-time performance sampled from using different camera numbers.

**Table 3.** Real-time results measured in frames per second (fps or Hz) in the Trondheim city model for individual sensors and results from a multi-sensor simulation.

| Sensor         | VL camera | IR camera | Lidar | Radar | All sensors |
|----------------|-----------|-----------|-------|-------|-------------|
|                | 45        | 80        | 50    | 35    | 15          |

4. Discussion

Results showed high fidelity VL images seen in Figure 3, close to what can be observed in existing automotive simulators [1, 12, 2]. However, the dependency on quality and quantity of scene content such as materials and 3D models, shows the current limitation and challenges of using game engines as simulation platforms.

Similar reasoning can be used for the IR camera, where the emissivity texture proved to be the limiting factor, best seen by comparing the human faces in Figure 4. This was however as expected, since artistic textures where used when no empirical data was available, moreover 3D models with both IR and VL textures. Lightning did however reassemble real IR photographs, indicating a potential for further work.

Lidar was capable of generating high volumes of data in real time, while being able to maintain the simulations frame rate, very similar to LGSVL’s GPU accelerated lidar [2]. As expected, a beam shape error from stitching multiple depth-buffers did occur as seen in Figure 7, though the unforeseen trade-off between performance and accuracy in increasing the number of cameras, did limit the proposed lidar model. This shortcoming could be explained by the increased *draw calls* the render pipeline needs to handle as the number of virtual cameras increases.
A proposed 3D sensor model for radar showed potential of running in real-time on a GPU seen in Table 3. Results carried little resemblance to existing stand alone solutions such as Carpet [19], moreover with real world data as seen in Figure 6, some of which can be explained by the lack of virtual content relative to the real world. However, the main contributing factor that limits the fidelity is probably the negligence of RCS.

Table 3 shows acceptable real-time performances for all sensors, especially considering it was running on a laptop. Also the 15Hz performance for multi-sensor simulations are believed to be sufficient for testing virtual vessels using control systems with similar update frequency, such as the autonomous ship Milliampere. As real-time rendering through G-buffers compromises on fidelity versus performance, sensor models is constrained by the poly count in 3D meshes and lightning fidelity in comparison to other rendering techniques such as ray tracing. Because of this, simulators based on other game engines, e.g Unreal [1], that utilizes real-time global illumination based on ray tracing, might be able to improve the sensor models further.

5. Conclusion
In this paper, we showed simulations of EMR sensors with high fidelity VL camera similar to automotive simulators, a customizable lidar capable of generating high volumes of data, a IR camera using lightning based on black body radiation and artistic textures, and a radar taking into account 3D information, all running on models implemented on the GPU. This demonstrated that game engines are capable of simulating multiple EMR sensors of maritime interest, running concurrently in real-time given the proper hardware. From these results in context of related work stated in the introduction, it can be concluded that Autoferry Gemini is the first known simulator that combines VL, IR, lidar and radar simulations in such a way for autonomous ships. Further work will need to emphasize data gathering techniques and modelling to address the lack of EMR content, such as emmissivity and 3D models. Sensitivity analyses could also be carried out to account for real-world imperfections that may occur from actual sensors.

Acknowledgments
A special thanks to Master student’s at NTNU’s Department of Design, for donation of 3D models such as Milliampere 2.0 by Petter Mustvedt [28] and the ferry’s docking stations designed by Peter Glesaaen and Henrik Ellingsen. Many thanks to Ramboll for providing the city model of Trondheim, Renderpeople for providing 3D human models, and Christopher Fantauzzo for his Old Seaport Unity asset. The last thanks goes to Erik Wilthil and Øystein Kaarstad Helgesen that provided real VL and IR images in addition to radar data from Milliampere 1.0.

References
[1] Shah S, Dey D, Lovett C and Kapoor A 2018 AirSim: High-fidelity visual and physical simulation for autonomous vehicles Proc. of Field and Service Robotics ed Hutter M and Siegwart R (Cham: Springer International Publishing) pp 621–635
[2] LG 2018 LGSVL simulator https://www.lgsvlsimulator.com/
[3] Li G, Mao R, Hildre H P and Zhang H 2020 Visual attention assessment for expert-in-the-loop training in a maritime operation simulator IEEE Transactions on Industrial Informatics 16 522–31
[4] Mao R, Li G, Hildre H P and Zhang H 2019 Analysis and evaluation of eye behavior for marine operation training - a pilot study Journal of Eye Movement Research 12 3
[5] Ana M, Li G and Zhang H 2019 A decentralized sensor fusion approach to human fatigue monitoring in maritime operations Proc. of ICCA vol 15 (Edinburgh: IEEE) pp 1569–74
[6] Sanfilippo F, Hatledal L I, Styve A, Pettersen K Y and Zhang H 2016 Integrated Flexible Maritime Crane Architecture for the Offshore Simulation Centre AS (OSC): A Flexible Framework for Alternative Maritime Crane Control Algorithms IEEE Journal of Oceanic Engineering 41 450–61
[7] Sanfilippo F 2016 A multi-sensor system for enhancing situational awareness in offshore training Proc. of CyberSA (London: IEEE) pp 1–6
[8] Li N, Su Z, Chen Z and Han D 2013 A real-time aircraft infrared imaging simulation platform Optik 124 17

[9] Chen X, Zhou X, Qian W, Chen Q, Wan M and Gu G 2018 Infrared simulation and implementation of virtual ocean scene Proc. of SPIE vol 10765 ed Strojnik M and Kirk M S (Bellingham: SPIE) pp 165–172

[10] Dai Y, Chen B and Ming D 2013 Research on real-time infrared simulation of ground scene Proc. of SPIE vol 8917 ed Zhang X and Liu J (Bellingham: SPIE) pp 100 – 106

[11] Zhang F, Wang T, Huang K, Mao J, Li M and Wang Z 2018 One improved real-time infrared simulation system based on unity3d Journal of Computer-Aided Design and Computer Graphics 30 1177

[12] Dosovitskiy A, Ros G, Codevilla F, Lopez A and Koltun V 2017 CARLA: An open urban driving simulator Proc. of Machine Learning Research vol 78 ed Levine S and et al (Zürich: PMLR) pp 1–16

[13] Baldini F, Anandkumar A and Murray R M 2019 Learning pose estimation for uav autonomous navigation and landing using visual-inertial sensor data ArXiv abs/1912.04527

[14] Su H, Wang R, Chen K and Chen Y 2019 A simulation method for lidar of autonomous cars IOP Conf. Series: Earth and Environmental Science 234 012055

[15] Sutherland I E, Sproull R F and Schumacker R A 1974 A characterization of ten hidden-surface algorithms ACM Computing Surveys 6 1–55

[16] Muckenhuber S, Holzer H and Bockaj Z 2020 Automotive lidar modelling approach based on material properties and lidar capabilities Sensors 20 3309

[17] Manivasagam S, Wang S, Wong K, Zeng W, Sazanovich M, Tan S, Yang B, Ma W C and Urtasun R 2020 Lidarsim: Realistic lidar simulation by leveraging the real world Proc. of CVPR pp 11167–76

[18] Li W, Zhang H, Hildre H P and Wang J 2014 An FPGA-based real-time UAV SAR raw signal simulator IEICE Electronics Express 11 20140168

[19] Huizing A G and Theil A 1991 CARPET: A versatile program for radar performance analysis Final Report Physics and Electronics Lab. TNO

[20] Hem A G, Leinebø D, Opheim H V, Vasstein K and Skarshaug T 2019 Digital tvilling Project report from a group in the village Autoferge in Experts in Team at NTNU

[21] Lauvsnes M 2020 Feature-based pose estimation of ships in monocular camera images Master’s thesis NTNU

[22] Vasstein K 2020 Real time simulation of EMR sensors for autonomous ferries Project thesis NTNU

[23] Deering M, Winner S, Schediwy B, Duffy C and Hunt N 1988 The triangle processor and normal vector shader: a VLSI system for high performance graphics SIGGRAPH Comput. Graph. 22 21–30

[24] Young H D and Freedman R A 2016 Sears and Zemansky’s University Physics with Modern Physics (Edinburgh: Pearson Education Limited) pp 1328 – 31

[25] Meola C, Boccardi S and Carlomagno G M 2017 Infrared Thermography in the Evaluation of Aerospace Composite Materials ed Meola C and et al (Cambridge: Woodhead Publishing) pp 57 – 83

[26] Richards M A, Scheer J A and Holm W A 2010 Basic Principles (Principles of Modern Radar vol 1) ed Kay D R and Janelle K (New Jersey: SciTech Publishing) pp 64, 211–19

[27] Morales J, Plaza V, Mandow A, Gomez J A, Serón J and García A 2018 Analysis of 3d scan measurement distribution with application to a multi-beam lidar on a rotating platform Sensors 18 2

[28] Mustvedt P 2019 Autonom ferge designet for å frakte 12 passasjerer trygt over Nidelven Master’s thesis NTNU