Real-time simulation of thermal shadows with EMIT

Andreas Klein[*a], Stefan Oberhofer[b], Peter Schätz[a], Alfred Nischwitz[b], Paul Obermeier[a]
[a]MBDA Germany, Hagenauer Forst 27, Schrobenhausen, Germany, 86529
[b]Munich University of Applied Sciences, Lothstr. 34, Munich, Germany, 80335

ABSTRACT

Modern missile systems use infrared imaging for tracking or target detection algorithms. The development and validation processes of these missile systems need high fidelity simulations capable of stimulating the sensors in real-time with infrared image sequences from a synthetic 3D environment.
The Extensible Multispectral Image Generation Toolset (EMIT) is a modular software library developed at MBDA Germany for the generation of physics-based infrared images in real-time. EMIT is able to render radiance images in full 32-bit floating point precision using state of the art computer graphics cards and advanced shader programs.
An important functionality of an infrared image generation toolset is the simulation of thermal shadows as these may cause matching errors in tracking algorithms. However, for real-time simulations, such as hardware in the loop simulations (HWIL) of infrared seekers, thermal shadows are often neglected or precomputed as they require a thermal balance calculation in four-dimensions (3D geometry in one-dimensional time up to several hours in the past).
In this paper we will show the novel real-time thermal simulation of EMIT. Our thermal simulation is capable of simulating thermal effects in real-time environments, such as thermal shadows resulting from the occlusion of direct and indirect irradiance. We conclude our paper with the practical use of EMIT in a missile HWIL simulation.

Keywords: thermal shadows, real-time infrared, infrared image generation, environment occlusion

1. INTRODUCTION

MBDA Germany has been using simulations with infrared image generation for two decades now to stimulate real and virtual infrared seekers. In order to cover a wide range of climatic and atmospheric conditions, synthetic environments are used. Infrared images with appropriate frame rates are required for closing the guidance and control loop of missile systems in order to feed an IR seeker with images from arbitrary just-in-time computed points of view, i.e. not only IR videos.

In response to experiences with commercial infrared image generation tools as well as our special requirements, MBDA Germany has developed a software infrastructure called EMIT to generate infrared images. Full control over the image generation process is necessary to take into account new project requirements and internal research needs. Another advantage of an in-house development is the usage of the infrastructure as a verification, validation and accreditation tool for the performance evaluation of missiles. Having the possibility of code inspection, the process of validation and verification of the used models can be achieved more easily. Hardware-in-the-Loop (HWIL) simulations for the stimulation of infrared seekers imply the highest requirements to image generation: Supply high and synchronized frame rates with a variety of different synthetic scenarios.

The basis of a 3D infrared image generation system is a heat balance simulation, which performs an unsteady-state heat conduction for the 3D geometry in a scene for a certain time span. The resulting surface temperatures are then converted into radiance values. However, a 4D unsteady-state heat conduction is computational expensive and therefore, important effects are often precomputed or neglected for the use in real-time simulations, such as HWIL simulations.

In a previous publication[1], we incorporated thermal shadows into a real-time 3D infrared image generation system which increases the fidelity of synthetic infrared images. However, this approach has a few limitations. First, we used precomputed temperature curves to estimate an equilibrium temperature with a linear interpolation using the shadow factor. This limited the approach to a single heat source in the 3D scene. Second, we neglected the fact that thermal heating and cooling effects also occurs due to the influence of the irradiance from the environment, such as atmospheric irradiation or reflected radiance. Figure 1 shows examples of such thermal effects. On the left image, the corner and the edges of an office cupboard are subject to more irradiance as the influence of the heat flux from

[*] andreas-herbert.klein@mbda-systems.de

environment is greater. The image on the right hand side shows an image of a ripped surface. In the dents there is a cooling effect occurring from the shadowing of environment irradiance.

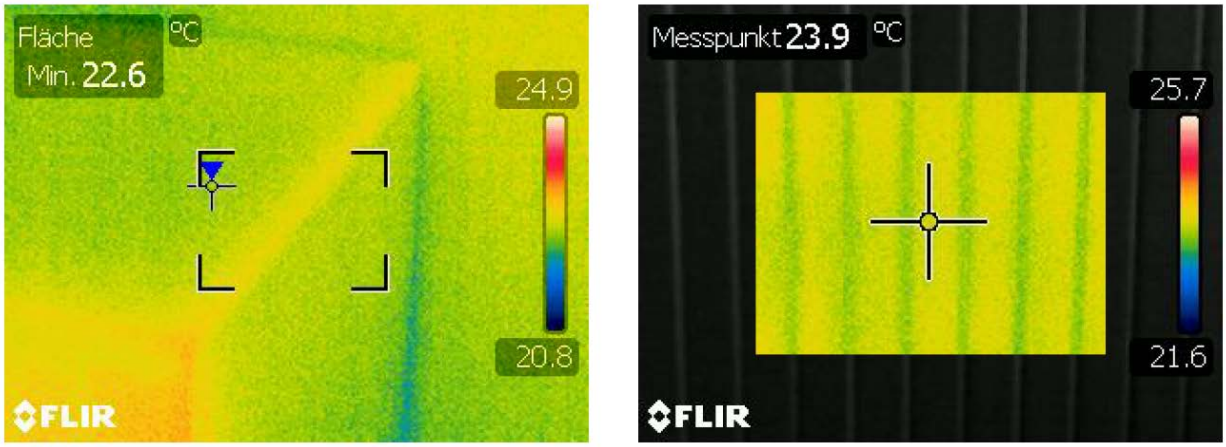


Figure 1. Examples of thermal effects due to the influence of a heat flux from the environment. (a) heating effect on the corner and edges of an office cupboard. (b) cooling effect on dents of a ripped surface.

In this paper, we address these limitations and present a novel approach to calculate thermal effects resulting from the direct and indirect occlusion of irradiance. Our idea is to define the thermal contributions on an idealized model and calculate an environment temperature using a radiation balance. With this environment temperature, we estimate an equilibrium temperature and adapt the surface temperature exponentially using our thermal model presented previously[1]. We combine this thermal simulation with real-time 3D computer graphics algorithms, in order to achieve a real-time frame rate in the simulation.

2. LITERATURE REVIEW

2.1 Shadows in the Infrared Spectrum

There are commercial products for 3D infrared image generation, which incorporate thermal shadows for non-real-time simulations. However, these products are often classified and no technical details are published. Therefore, we focus the literature review on research publications, which are relevant for this work.

Cathcart[2] realized thermal shadows in a 3D infrared image generation by modelling an apron around shadow casters and assigned optical properties to it.

Poglio[3] et al. adaptively triangulate a 3D scene into layered, homogenous patches. These patches are used to calculate a heat balance solution using a non-real-time radiosity approach. Shadows are realized by using shadow mapping.

Biesel and Rohlfing[4] present a real-time 3D infrared image generation system. Their idea is to assume thin materials and calculate a steady-state heat balance solution. However, shadows are neglected.

Schott et al.[5] combine a non-real-time unsteady-state thermal model with a ray tracer. A thermal shadow is realized by calculating a shadow history for each pixel. If a pixel is in shadow for a given time, the incident solar radiation is disabled and the surface temperature is adapted using the thermal model.

Maréchal et al.[6] realize winter scenarios by calculating a heat balance for a voxelized 3D scene using the finite volume method. A thermal shadow is produced by tracing a ray towards the sun for each voxel. If the ray intersects geometry, the solar irradiation is disabled.

Further approaches focus on the calculation of infrared signatures of ships. These infrared signatures include thermal shadows. Dumont et al.[7] calculate the infrared signature of a ship using the finite element method. Lapierre and Acheroy[8] hierarchically subdivide facets in order to accelerate the computation.

2.2 Shadows in the Visual Spectrum

The rendering of shadows for direct and indirect lighting has been studied extensively over the last years. Therefore, we focus our review on publications closely related to our work. For an exhaustive survey on other methods see Eisemann et al.[9] and Ritschel et al.[10].

Shadow mapping[11] is a popular algorithm for generating shadows in real-time rendering. The idea is to use a depth map (also known as a shadow map) for the occlusion test. A 3D scene is first rendered from the light's point of view and the depth values are stored in a depth map. In a second render pass, the scene is rendered from the observer's point of view and the depth values are transformed into the light's point of view. The occlusion test can then be realized by comparing the depth values. A pixel is classified to be in shadow if the depth in the depth map is smaller than the transformed depth value from the observer's point of view.

Percentage closer filtering[12] (PCF) is a technique for generating soft shadows by making multiple shadow comparisons within a user defined filter window in the depth map. The shadowing factor is then built by averaging the results. Fernando[13] extends this idea to generate contact-hardening soft shadows with PCF in an approach called Percentage Closer Soft Shadows (PCSS). He introduced an occluder search as a preprocessing step, where an average occluder depth for each screen space pixel is calculated and a penumbra width with a parallel planes approximation is approximated. Finally, the penumbra width is used to scale the PCF filter window. PCSS is up to now one of most used algorithms for computing realistic soft shadows in computer games as well as in simulations.

2.3 Ambient Occlusion

Ambient occlusion is a method to calculate shadows from indirect illumination. The idea is to calculate an occlusion factor by estimating the visible surface of a hemisphere above a surface point. The incident irradiation is then dynamically adapted with this occlusion factor.

In real-time computer graphics, ambient occlusion can be realized in a screen-space approximation using a depth map[14], which is rendered from the observer's point of view (in the contrary to shadow mapping where the depth map is rendered from the light's point of view) The idea is to create 3D samples randomly distributed within a sphere, access the depth map at the sample location and test for occlusion. The occlusion test is similar to shadow mapping. The depth values of the randomly distributed 3D samples are compared against the stored values in the depth map. The occlusion factor is the ratio of the occluded samples to the total number of samples, which exceeds 0.5, since half of 3D samples lies in the lower hemisphere

Volumetric obscurance[15] estimates ambient occlusion using line sampling. In order to achieve a continuous occlusion factor, the volume of the sphere is divided into cuboids. A fraction of the sphere volume is assigned to each line. During the visibility test, each line is tested for visibility and the assigned volume modified by the visible fraction of the line.

3. OUR ALGORITHM FOR THERMAL SHADOWS

Our idea to realize real-time simulations of thermal shadows is to calculate an equilibrium temperature and adapt the surface temperature with two exponential functions. One exponential function represents the short-time solution and the second one the long-time solution. These exponential functions are weighted and scaled by fitting them against a reference solution (see Klein et al.[1] for details).

In order to calculate the equilibrium temperature, we define an environment temperature based on the heat contributions of a black body. In particular, we use the following heat contributions:

- The direct heat flux S_d of a heat source.
- The indirect heat flux S_i of a heat source.
- The heat flux R_e of the environment.
- The low wavelength heat flux R_{LW} of the atmosphere.

Further heat contributions, such as convection, evaporation or conduction are neglected for our approach. Using these contributions, the environment temperature T_a can be defined with a radiation balance:

$$\varepsilon\sigma T_a^4 = S_d\, f_d (1-\alpha)(1-\beta) + S_i\, f_i (1-\alpha) + (1-f_i) R_e + \varepsilon f_i R_{LW} \tag{1}$$

where f_d is the shadow factor for the direct heat flux, f_i is the occlusion factor for the indirect heat flux, ε is the emissivity of the surface, α is the albedo of the surface, β is a factor for the cloudiness and the Stefan-Bolzmann constant σ. We use shadow mapping to calculate the shadow factor for the direct heat flux and ambient occlusion to estimate the occlusion factor.

We calculate the environment temperature by solving Equation 1 for T_a:

$$T_a = \sqrt[4]{\tfrac{1}{\varepsilon\sigma}\left(S_d\, f_d (1-\alpha)(1-\beta) + S_i\, f_i (1-\alpha) + (1-f_i) R_e + \varepsilon f_i R_{LW}\right)} \tag{2}$$

3.1 Direct Heat Flux

The direct heat flux is the heat flux resulting from a direct irradiance of a heat source. As it is dependent on the surface orientation, we define it as following:

$$S_d = S_{d0} \cos(N \cdot L) \tag{3}$$

where S_{d0} is the direct irradiance $\left\lfloor \frac{W}{m^2} \right\rfloor$, N the normal of the surface and L the direction vector from the surface to the heat source and "·" is the dot product between two vectors.

3.2 Indirect Heat Fluxes

The indirect heat flux consists of the heat flux of a heat source that is reflected from the environment:

$$S_i = \int_\Omega S_{i0} \cos(\gamma) d\Omega \tag{4}$$

where S_{i0} is the indirect irradiance $\left\lfloor \frac{W}{m^2 sr} \right\rfloor$, Ω the sphere around a surface point and γ the incident irradiation angle. We use a full sphere instead of a hemisphere in order to realize heating and cooling effects on geometry with sharp edges, where a solid angle of more than 2π sr is visible. We assume a homogenous heat density which simplifies the integral over the sphere to $S_i = S_{i0} 4\pi$.

Additionally, we account for the indirect heat fluxes of the environment and the long wavelength heat flux of the atmosphere. The indirect heat fluxes of the environment are calculated with a reference solution, such as RadTherm[16]. The latter contribution is calculated by using an atmospherical model, such as MODTRAN[17].

3.2 Equilibrium Temperature

We assume that a surface consists of a single, infinitely extended layer. We define the equilibrium temperature T_s as the equilibrium between the radiation and the conductive heat flow in the surface:

$$\varepsilon\sigma(T_a^4 - T_s^4) = \lambda \frac{T_s - T_c}{d} \tag{5}$$

where ε is the emissivity of the surface, T_a the environment temperature, T_c the core temperature, λ the thermal conductivity, d the thickness of the surface and σ the Stefan-Bolzmann constant. The core temperature is the temperature of an underlying layer which will be assumed to be constant during simulation. In order to calculate the equilibrium temperature, we solve Equation 5 for T_s:

$$T_s = \frac{T_b}{2}\left(\sqrt{\frac{2\sqrt{3K}}{\sqrt{3K^2 - 4\theta_c - 4\theta_a^4}} - K + \frac{4\theta_c + 4\theta_a^4}{3K}} - \frac{\sqrt{3K^2 - 4\theta_c - 4\theta_a^4}}{\sqrt{3K}}\right) \tag{6}$$

With the terms:

$$T_b \equiv \frac{1}{\sqrt[3]{b}} \quad b \equiv \frac{d\sigma\varepsilon}{\lambda} \quad K \equiv \left(\frac{N}{2\cdot 3^{3/2}} + \frac{1}{2}\right)^{1/3} \quad \theta_C \equiv \frac{T_C}{T_b} \quad \theta_a \equiv \frac{T_a}{T_b}$$

$$N = \sqrt{256\frac{T_c^3}{T_b^3} + 768\frac{T_a^4 T_c^2}{T_b^6} + 768\frac{T_a^8 T_c}{T_b^9} + 256\frac{T_a^{12}}{T_b^{12}} + 27}$$

4. IMPLEMENTATION

We implemented the proposed algorithm using real-time computer graphics algorithms. For the implementation, we define a time range for the simulation and divided it into a number of iteration steps. We perform the thermal simulation for each iteration step. Figure 2 shows an overview of the implementation.

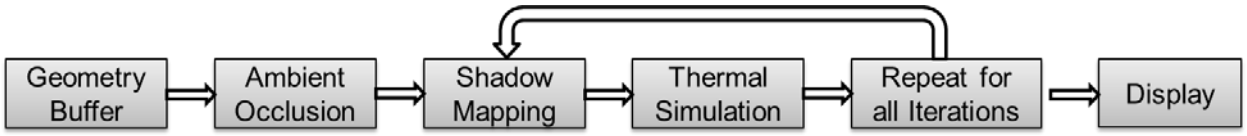


Figure 2. Overview of the implementation.

Our implementation supports 3D scenes with dynamic geometry and can be divided into the five parts: geometry buffer rendering, ambient occlusion, shadow mapping, thermal simulation and display. The shadow mapping and thermal simulation is executed for each iteration step. The geometry buffer rendering and ambient occlusion steps are only executed when camera or the 3D geometry is changed.

4.1 Geometry Buffer Rendering

We implemented our algorithm with a deferred rendering approach. A deferred renderer caches the visible 3D geometry of a scene in each frame into a geometry buffer (g-buffer) texture. If the 3D scene is rendered multiple times from the same viewport, this will reduce the workload of the vertex and geometry stage in the graphics pipeline of a graphics card. In the g-buffer texture, we store the following attributes: The linearized z-value of the geometry and an index into a material lookup table.

4.2 Ambient Occlusion

In our implementation, we use volumetric obscurance to calculate an occlusion factor. The implementation follows the approach of Ownby et al.[18]. We modified the scaling of the occlusion factors in such a way that unoccluded planar surface points will have an occlusion factor of 0.5, which represents a fully visible hemisphere. Occluded points will have an occlusion factor of < 0.5 and exposed points, e.g. in edges of a geometry, an occlusion factor of > 0.5. Figure 3 shows an example of the resulting occlusion factors.

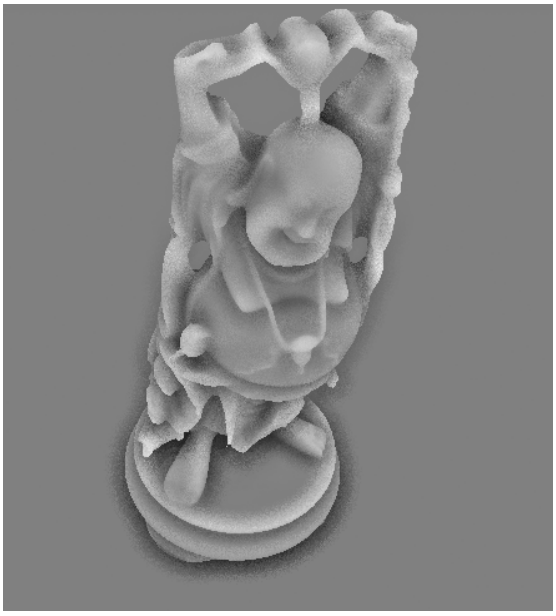


Figure 3. Example of the occlusion factors.

4.3 Shadow Mapping

In the shadow mapping part, we render a shadow map for each simulation step. A camera is setup for the current position of the heat source that produces the direct heat flux. The 3D scene is then rendered from this position and the depth values are stored in a 2D texture. These depth values will then be compared against the depth values stored in the g-buffer. The shadow factors are computed during the thermal simulation step.

4.4 Thermal Simulation

For the thermal simulation, we first calculate the shadow factor for the direct heat flux using the percentage closer soft shadow (PCSS) [13] algorithm. In order to realize the shadow test, the depth values of the g-buffer must be transformed by the view and projection matrix of the camera positioned at the location of the heat source. As this transformation requires the full 3D coordinates of the vertex, we first reconstruct the 3D coordinate with a view ray and transform it with the view and projection matrix. In the next step, we read the occlusion factor from the occlusion texture and the material parameters from the material lookup table. With these data, we calculate the environment temperature using Equation 2 and the equilibrium temperature using Equation 6. Finally, we load the time constants and scaling factors for the thermal model from the material lookup table and adapt the surface temperature (see Klein et al.[1] for details).

4.5 Display

For this work, we display the surface temperatures using a false color display. The temperatures are read in and interpolated between a user defined minimum and maximum temperature. In order to simulate a sensor, the display step can be extended to incorporate atmospheric effects, such as atmospheric absorption or path radiance.

5. RESULTS

5.1 Comparing the Absolute Temperatures

We compare the result of our approximation against a RadTherm solution, which performs a 4D heat transfer simulation in non-real-time. For our approximation, we used a simulation time between 8:00 and 11:30 am. Both simulations use the same atmospheric data. The parameters for our thermal model are estimated by fitting the two exponential functions against the RadTherm solution.

Our 3D scene consists of a concrete block, which casts a shadow onto an asphalt road. Table 1 shows the used material parameters for our approximation. Figure 4 shows the result of our approximation in comparison to the RadTherm solution. It can be observed, that our solution produces a minor temperature difference in the core shadow area and on top of the block. However, the penumbra of the thermal shadow is larger in the reference solution. Therefore, the temperature difference in this area is increased. As we neglect the lateral heat conduction, our solution fails to reproduce the temperature in the top right and bottom left corner of the cement block.

Table 1. Used Parameters of our approximation

Surface	**Thickness**	**Thermal Conductivity**	**Albedo**	**Emissivity**
Asphalt	15cm	0.7	0.75	0.93
Concrete	10cm	1.28	0.6	0.88

Figure 4 shows a plot of two temperature curves of the image. It can be observed that our approximation reproduces the reference solution with a temperature difference up to 3.5 K. However, the temperature in the core shadow is cooler than in the RadTherm solution. This is a limitation of the single layer approach, where the heat exchange with lower layers is neglected.

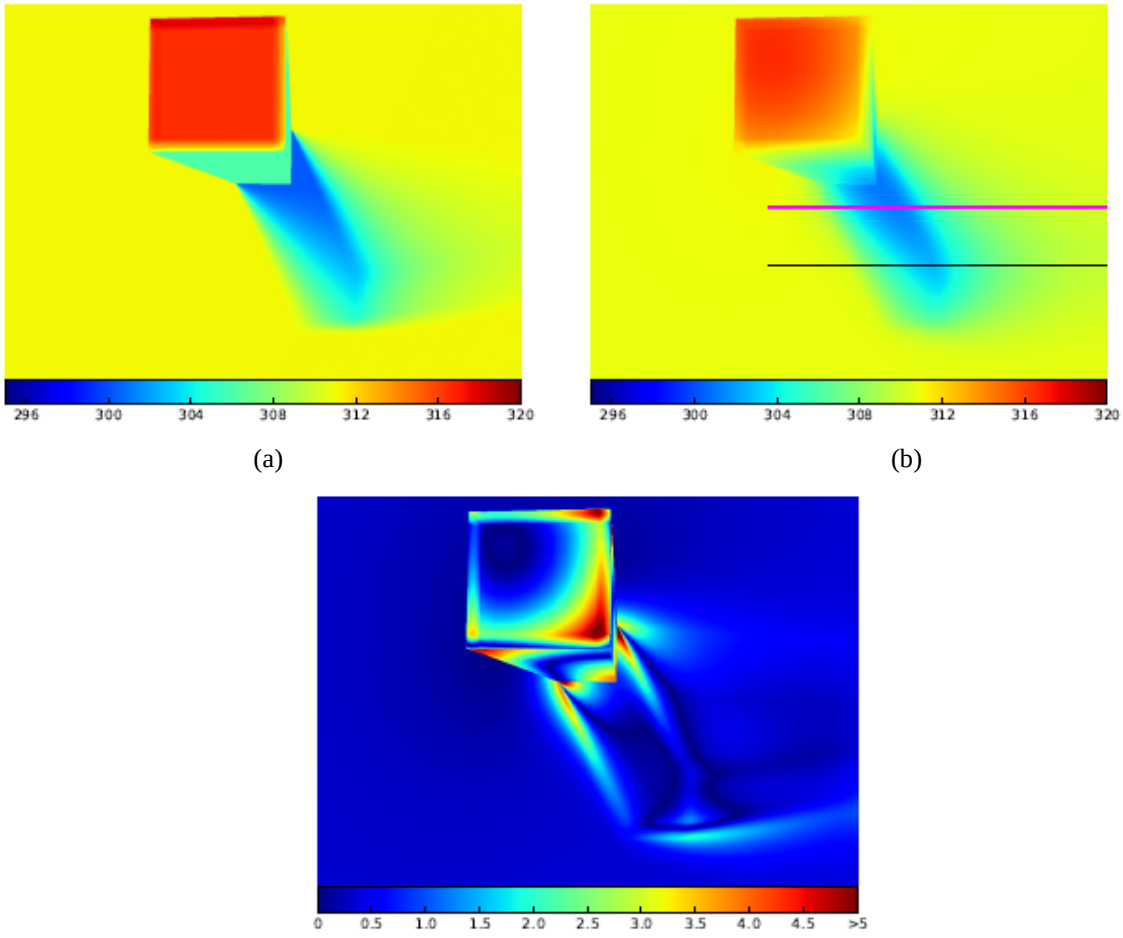
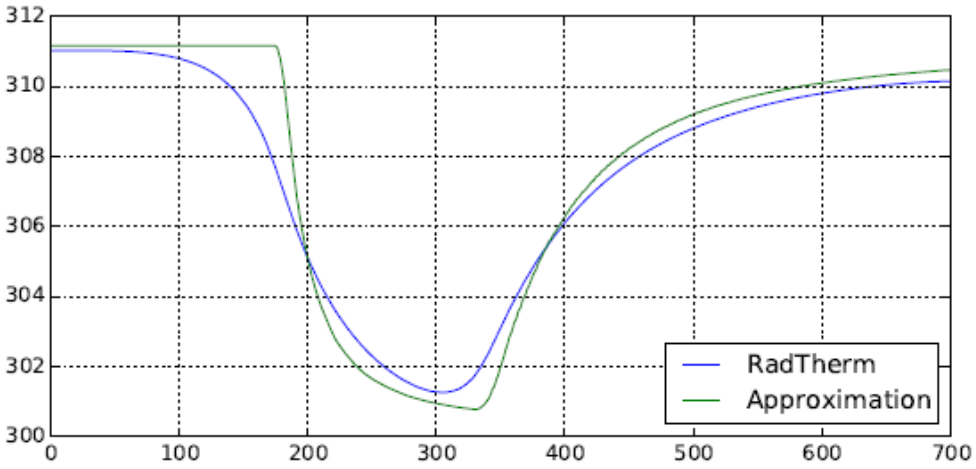
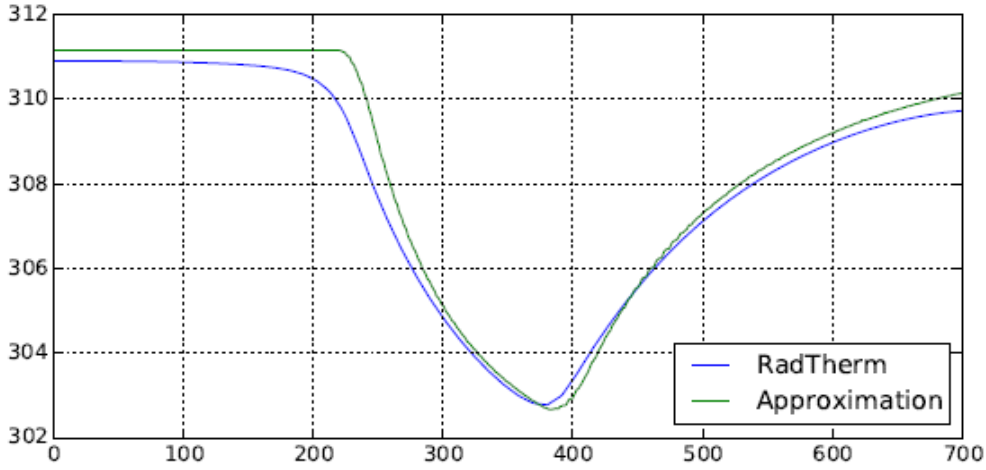


Figure 4. Result of our approximation (a), the RadTherm solution (b) and the difference image (c). The lines in (b) display the location of the temperature curves. Temperatures in Kelvin.



(a)



(b)

Figure 5. Temperature curves of the magenta (a) and black line (b). Temperatures in Kelvin.

5.2 Influence of the Atmospheric Occlusion

Figure 6 shows an example of a ribbed surface, where the influence of the atmospheric occlusion is visible. In this example, the environment temperature is warmer than the surface temperature. Therefore, the surface temperature on the edges is warmer due to an increased influence of the atmospheric radiation. On the other hand, the surface temperature in corners is cooler due to the reduced influence of the atmosphere.

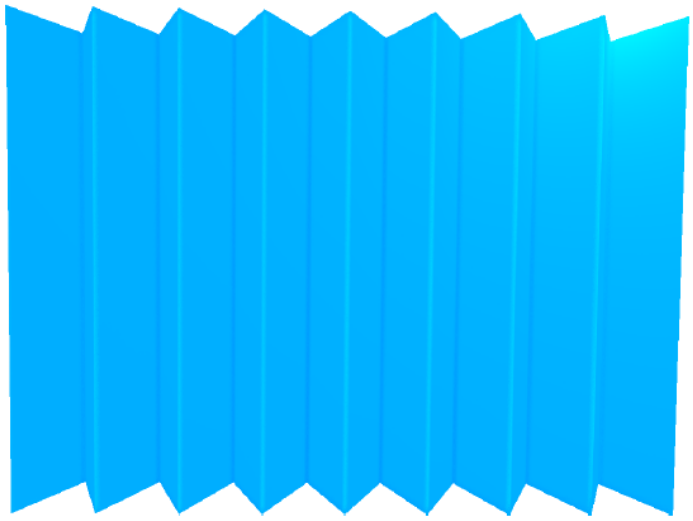
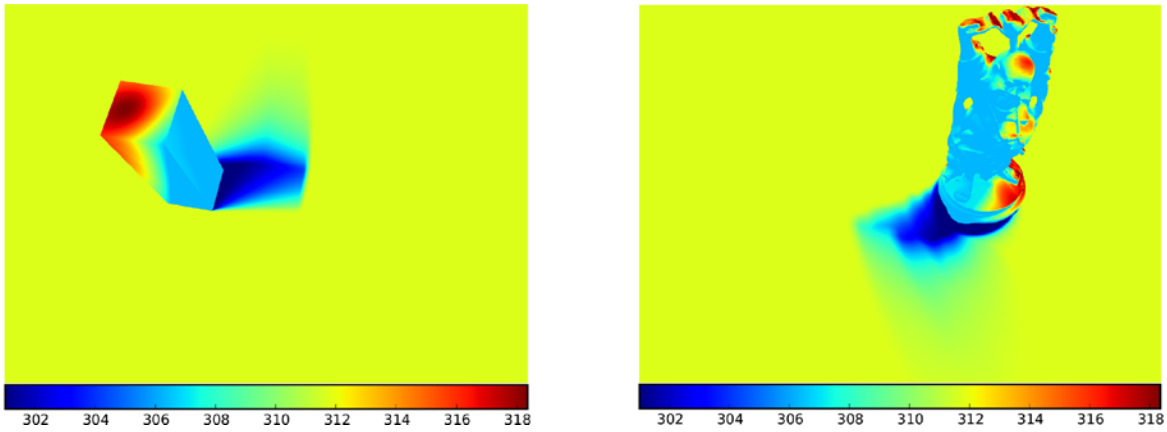


Figure 6. The temperature on exposed points is higher than the temperature in the corners of the surface.

5.3 Performance Results

In this section we measure the performance of our approach. The performance was measured on an Intel i7-3820 with 3.6 GHz, 16 GB RAM and a NVIDIA GeForce GTX Titan graphics card with 6144 MB memory. In our simulation, we used a screen resolution of 1024x768, a shadow map size of 1024x1024, 8 samples for the occluder search of PCSS and 32 samples for the PCF kernel. For volumetric obscurance, we used 13 paired samples. The times are the mean value of 1000 frames.



(a) (b)

Figure 7. Visual results in the box (a) and Buddha (b) test scene. Temperatures are in Kelvin.

Table 2 shows the performance results and Figure 7 the visual results in false color plots. We measured the performance in two test scenes with two different numbers of iteration steps as well. The first scene consists of a box, which casts a shadow on a plane. This scene has 45 polygons. In the second scene, we placed a Buddha statue, which consists of 1.08 million polygons. Our implementation can render the box scene within 8.4 milliseconds for 27 iteration steps and 25.2 milliseconds for 90 iteration steps. In the Buddha scene, the performance is reduced due to the number of polygons. The scene can be rendered in 25.2 milliseconds for 27 iteration steps and 56.8 milliseconds for 90 iteration steps. However, when comparing the number of polygons, the implementation scales

well with the increased number of polygons. This is the result of the deferred renderer, which caches the geometry once for each frame.

Table 2. Performance results in milliseconds and frames per second (FPS).

Scene	Iterations	Time
Box	27	8.4 ms (119 FPS)
Box	90	25.2 ms (39 FPS)
Buddha	27	18.2 ms (54 FPS)
Buddha	90	56.8 ms (17 FPS)

Table 3 shows the duration of the single steps in the algorithm. It can be observed, that the bottleneck of the algorithm is the rendering of the shadow maps. For each iteration step, a shadow map is rendered for the current sun position. However, the performance can be further improved by reusing shadow maps for multiple iteration steps, as shown in our previous publication[1].

Table 3. Duration of the single steps in the algorithm. Performance measured in the Buddha scene with 90 iteration steps.

Step	Time
G-Buffer	0.7 ms
Ambient Occlusion	1.1 ms
Shadow Maps	36.2 ms
Thermal Simulation	18.1 ms
Display	0.7 ms
Total	56.8 ms

6. USAGE IN HARDWARE IN THE LOOP SIMULATION

A missile system usually has several components, such as the seeker or the flight control and navigation computers. While most of these components are missile specific they still share some independent tasks. These tasks include the control of the flight motion simulator and the infrared image generation. The key challenge in this approach is the synchronization between the missile simulation and the simulation environment, such as the image generation and the flight motion simulation.

In order to accelerate the development of missile HWIL simulations, MBDA Germany developed a generic HWIL architecture, which is shown in an overview in Figure 8. The idea is to generalize the common concepts in each missile HWIL simulation and realize them on a rendering and control computer. The rendering and control computer has two main tasks. First, it generates infrared images from synthetic 3-D scenarios using the EMIT software. These images can be either directly injected into the missile's image processing computer or projected onto the seeker using an infrared projector. The second task is the control of the flight motion simulator, where the missile position and the position of the target are simulated. Currently we are extending the features of our generic HWIL approach by integrating a space segment simulator. This allows the use of the GPS receiver of a navigation computer to run in a closed loop simulation.

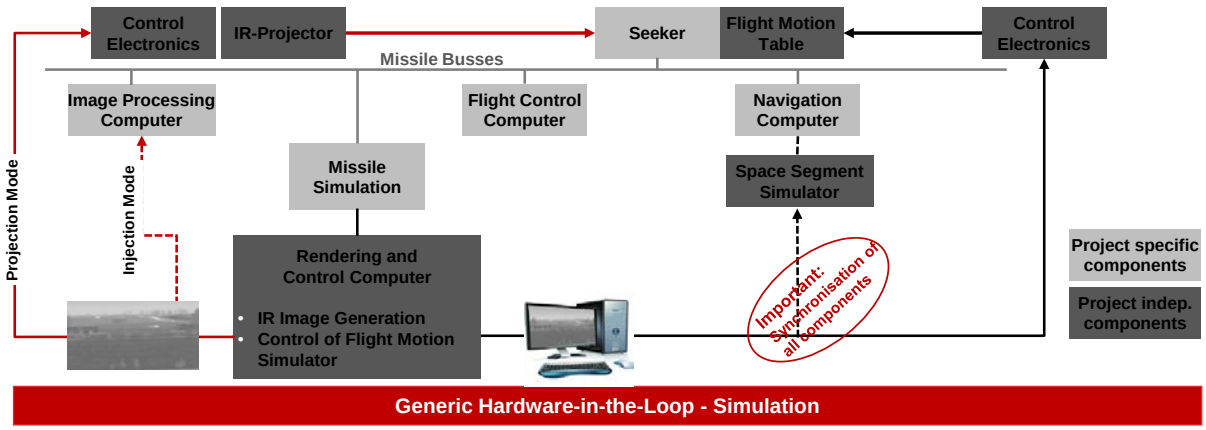


Figure 8. Overview of the generic HWIL architecture.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an algorithm to calculate thermal effects resulting from the occlusion of direct and indirect irradiance. We used a black body to model the incoming and outgoing heat contributions and extracted an environment temperature using a radiation balance. This environment temperature is used to calculate the equilibrium temperature based on an equilibrium between radiation and soil heat flow. The surface temperatures of a geometry in a 3D scene are then adapted exponentially using the thermal model presented in our previous publication[1]. The main progress of our work is that we can reach an IR image quality close to RadTherm but in conjunction with real-time frame rates. In order to maintain real-time frame rates, we implemented the algorithm using ambient occlusion and shadow mapping algorithms of real-time 3D computer graphics. The results show that our approximation can handle geometries with up to 1.08 million polygons with interactive frame rates. Furthermore, we demonstrated that our approximation can represent the results of a numerical simulation with RadTherm with an acceptable accuracy for multiple materials in this use case.

In future works, we intend to extend our algorithm to include further thermal contributions, such as convection and conduction. Furthermore, we plan to improve the rendering of larger penumbras areas, in order to reduce the temperature differences in these areas.

REFERENCES

[1]A. Klein, A. Nischwitz, P. Schätz and P. Obermeier, "Incorporation of thermal shadows into real-time infrared three-dimensional image generation", Opt. Eng. **53**(5), 053113 (2014) [doi:10.1117/1.OE.53.5.053113].
[2]M. J. Cathcart, “Thermal shadow modelling within a physically-based scene simulation,” Proc. SPIE **1967**, 256–267 (1993).
[3]T. Poglio, E. Savaria, and L. Wald, “Outdoor scene synthesis in the infrared range for remote sensing applications,” *in Proc. International Conference on Imaging Science, Systems and Technology (CISST)*, CSREA Press, Las Vegas (2002).
[4]H. Biesel and T. Rohlfing, “Real-time simulated forward looking infrared(FLIR) imagery for training,” Proc. SPIE **0781**, 71–80 (1987)
[5]J. R. Schott, R. V. Raqueno, and C. Salvaggio, “Incorporation of a time-dependent thermodynamic model and a radiation propagation model into IR 3D synthetic image generation,” Opt. Eng. **31**(7), 1505–1516 (1992)
[6]N. Maréchal, E. Guérin, E. Galin, S. Mérillou, and N. Mérillou, “Heat transfer simulation for modeling realistic winter sceneries”. Computer Graphics Forum **29**(2), 449–458, (2010) [doi:10.1111/j.1467-8659.2009.01614.x]

[7]R. Dumont, T. Robert, Y. Verdavaine, F. Lapierre, M. Acheroy, and J.-P. Marcel. "Safir, a testbed for maritime simulation", *in 3rd International IR Target, Background Modelling & Simulation (ITBMS) Workshop*, ONERA, Toulouse, France, (2007).
[8]F.D. Lapierre, and M. Acheroy. "Performance enhancement and validation of the open-source software for modeling of ship infrared signatures (osmosis)". Journal of Computational and Applied Mathematics **234**(7), 2342–2349, (2010) [doi:10.1016/j.cam.2009.08.091].
[9]E. Eisemann, M. Schwarz, U. Assarsson, and M. Wimmer, Real-Time Shadows, A K Peters/CRC Press (2011).
[10]T. Ritschel, C. Dachsbacher, T. Grosch, J. Kautz. „The State of the Art in Interactive Global Illumination". Computer Graph. Forum **31**(1): 160-188 (2012)
[11]L. Williams, "Casting curved shadows on curved surfaces", in SIGGRAPH '78, pp. 270–274, ACM, New York (1978).
[12]W. T. Reeves, D. H. Salesin, and R. L. Cook, "Rendering antialiased shadows with depth maps," in SIGGRAPH '87, pp. 283–291, ACM, New York (1987).
[13]R. Fernando, "Percentage-closer soft shadows", in ACM SIGGRAPH 2005 Sketches, ACM, New York, NY (2005).
[14]M. Mittring. "Finding next gen: Cryengine 2" in ACM SIGGRAPH 2007 Courses, SIGGRAPH 07, 97–121. ACM, New York, NY, USA, (2007)
[15]B. F. Loos, and P.-P. Sloan. "Volumetric obscurance", in I3D'10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, 151–156. ACM, New York, NY, USA,(2010).
[16]Thermo Analytics Inc., "RadTherm," http://www.thermoanalytics.com (5. October 2014).
[17]G. P. Anderson et al., "MODTRAN4: radiative transfer modeling for remote sensing," Proc. SPIE **3866**, 2–10 (1999).
[18]J.-P. Ownby, R. Hall und C. Hall. "Rendering techniques in toy story 3", in Advances in Real-Time Rendering in Games, SIGGRAPH 2010 courses. ACM, New York, NY, USA, (2010)