12th INTERNATIONAL SCIENTIFIC CONFERENCE ON DEFENSIVE TECHNOLOGIES
OTEH 2026
Serbia, Tara, 21 - 24 October 2026

REAL-TIME THERMAL IMAGE SYNTHESIS IN UNITY 3D VIA EMPIRICAL INFRARED ADAPTATION OF LIGHT SIMULATION MODELS

BOJAN ŠMIDLING
University of Belgrade, Institute Mihailo Pupin, Belgrade, bojan.smidling@pupin.rs

STEFAN STOJKOV
University of Belgrade, Institute Mihailo Pupin, Belgrade, stefan.stojkov@pupin.rs

Abstract: *This work introduces a method for synthesizing thermal imagery in real time within the Unity 3D game engine (Unity Engine/Unity) Editor. The approach builds on the engine's existing light simulation mechanisms, with adjustments that approximate behaviour in the infrared domain. While grounded in real life samples, the method is largely empirical, with calibration driven by the goal of achieving visually convincing results across complete scenes, including dynamic elements and motion. A central outcome is the ability to maintain real-time performance, on the order of 60 frames per second, while remaining sufficiently lightweight to integrate into pre-existing simulation systems developed in Unity. The method is evaluated through direct visual comparison with data captured from thermal cameras, alongside a user-based visual recognition task designed to probe perceived realism. Basic distinctions between material classes, such as metals, plastics, and fabrics, are represented, and the model accounts for ambient thermal contributions influenced by solar exposure and environmental temperature. The primary constraint lies in the preparatory effort required to define thermal characteristics for individual scene elements. Despite this, the approach proves to be a practical and adaptable solution for simulation contexts, particularly in defence and industrial domains, where real-time, visually credible thermal rendering is required without imposing significant computational burden.*

Keywords: *Synthetic infrared image generation, Real-time thermal rendering, Unity Engine, Infrared radiance modelling, Sensor-aware image synthesis.*

1. INTRODUCTION

Infrared (IR) and thermal imaging have moved well beyond their historical military niche into surveillance, pedestrian and vehicle detection, industrial inspection, search-and-rescue, and robotics operating under degraded visibility.

Despite its enormous potential, the development and deployment of IR systems face a significant bottleneck - the scarcity of suitable IR data. Training robust and reliable IR systems requires large, diverse, and high-quality datasets. However, obtaining such datasets in real world is challenging due to privacy concerns, logistical issues, and the difficulty in capturing a wide range of scenarios and conditions [1].

Synthetic IR image data generation is a natural response, following the precedent set by game-engine-based synthesis in the visible domain. This paper proposes a reproducible method for generating synthetic infrared images, and potentially image sequences, using Unity version 2022. The claim is a method proposal, not a fully validated thermal-sensing solution, and not a claim of complete radiometric accuracy. The primary contribution is synthetic infrared rendering in Unity. Although not primarily a dataset-generation tool, as used in [2], the method may later support such workflows. The paper is a starting point for a broader research direction, using simple, self-contained examples to demonstrate feasibility of the core rendering-repurposing strategy rather than delivering a large-scale benchmark.

Beyond dataset scarcity, a second motivation for engine-based infrared synthesis is controllability. Physical thermal data capture is constrained by weather, time of day, sensor availability, and scene access, whereas a synthetic pipeline allows systematic variation of object temperature, emissivity, geometry, and camera viewpoint within a single reproducible scene. This controllability is particularly relevant for generating edge cases, unusual object-background temperature contrasts, or repeated scenarios with only one parameter changed at a time, none of which are easily obtained from field data collection. The cost of measured data is high, and it is difficult to obtain infrared images in various scenarios [3]. The present paper does not exploit this controllability quantitatively, since it presents only two illustrative scenes, but the underlying method is structured so that later work could vary these parameters systematically once the core rendering approach has been established as viable.

A further consideration is computational accessibility. Physically based infrared rendering, if pursued from first principles, typically requires dedicated radiometric renderers, detailed material spectral databases, and

Corresponding author: B Šmidling
E-mail: bojan.smidling@pupin.rs
DOI: 10.5937/OTEH26053S

substantial computational resources to reach acceptable frame rates. By contrast, the approach in this paper deliberately trades physical completeness for accessibility, relying on an engine that is freely available, widely taught, and already familiar to many engineers working on simulation tooling. The priority is establishing the basic approach, rather than delivering the most physically complete implementation achievable.

A final point worth stating explicitly concerns reproducibility. Since the method relies solely on standard Unity components, mesh objects, the Standard Shader's emission channel, a terrain object, and a script computing thermal radiation using Stephan-Boltzmann law per object, it can in principle be reimplemented in any recent Unity version without specialized plugins or proprietary rendering extensions. This stands in contrast to some physically based infrared renderers, which may depend on custom radiometric pipelines that are harder for other researchers to reproduce or extend. Ease of reimplementation is treated here as a practical advantage of the proposed approach, separate from its physical fidelity, and is offered as an additional argument for adopting it as a starting point for further research rather than as a claim about its ultimate accuracy.

Section 2 reviews Unity's visible-range rendering pipeline. Section 3 develops the analogy between visible and infrared rendering. Section 4 details the proposed method. Section 5 presents illustrative examples, and Section 6 concludes with limitations and future work.

2. VISIBLE RANGE RENDERING OVERVIEW

Game engine renders a scene composed of mesh-based objects, each assigned a material referencing a shader that determines how the surface responds to lighting and is rasterized into pixel values. The built-in Standard Shader used in this work exposes albedo, metallic, smoothness, and an emission channel that allows a surface to appear self-illuminated independently of external scene lighting [4].

A virtual camera defines viewpoint, field of view, and output resolution [5]. For each frame, the engine determines visible geometry from the camera viewpoint, applies lighting calculations per the active rendering pipeline, and writes resulting colour values into a two-dimensional image buffer [6]. This mature, real-time-efficient stack is precisely what motivates repurposing it for infrared rendering rather than building a separate renderer from scratch.

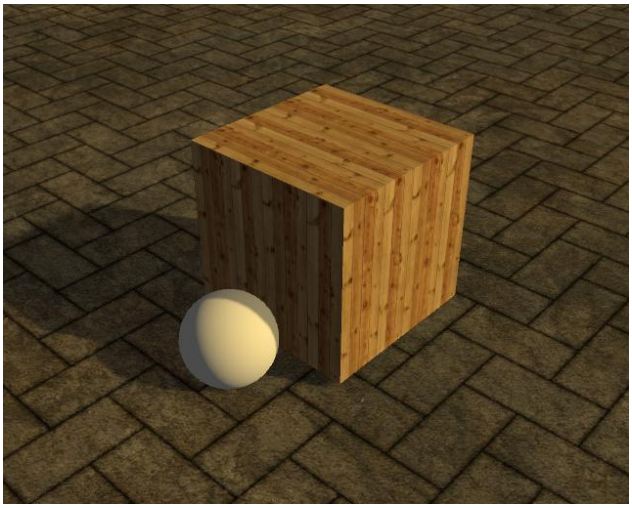


Figure 1. Example scene in visible range, concrete floor, wooden cube and brick sphere, illuminated by the Sun

3. CONNECTION AND ANALOGY OF VISIBLE AND IR RENDERING

The central idea is that the visible-range rendering stack need not be replaced, only repurposed, by changing what surface intensity represents. The same geometry, camera model, and largely the same material infrastructure are retained, but the value written to the output image is reinterpreted as an approximation of infrared emission rather than visible reflectance.

Unity does not natively render infrared radiation. Unity has no built-in concept of thermal emission or wavelength-dependent infrared material response. The method computes a simplified thermal value for each object outside Unity's native lighting model, then feeds that value into the emission property of the Standard Shader, reusing existing rasterization and display infrastructure. Emission functions purely as a rendering proxy, not as a physical model of infrared emission itself.

This distinction, between the physical thermal computation and the rendering mechanism used to display it, connects Section 2 to Section 4: once a thermal value is computed, it is injected into the existing visible-range pipeline through the emission channel, and the remainder of the pipeline proceeds exactly as for an ordinary scene. The method therefore inherits, essentially for free, Unity's existing support for real-time performance, dynamic elements, and animated characters, none of which need reimplementation.

4. METHOD

The proposed method is deliberately minimal in physical scope, covering surface temperature, surface emissivity, and a single spectral-band assumption framed as a long-wave infrared (LWIR) style approximation. Atmospheric transmission, reflected background radiation, detector calibration, sensor noise, optical blur, detector response, and angular or wavelength-dependent emissivity variation beyond the chosen band are intentionally omitted.

4.1. Thermal value computation

For a simplified broadband approximation, emitted radiant exitance is modelled using the Stefan-Boltzmann law:

$$M = \varepsilon\sigma T^4 \quad (1)$$

where M is emitted radiant exitance, ε is surface emissivity, σ is the Stefan-Boltzmann constant, and T is absolute surface temperature in Kelvin. Celsius inputs are converted to Kelvin before substitution.

Stefan-Boltzmann's law approximates total emitted thermal power integrated over all wavelengths, whereas real sensors measure band-limited radiance shaped by reflection, atmosphere, optics, and calibration. The present model is therefore a first-order thermal-appearance approximation, not a complete sensing model. Emissivity values are treated as approximate, band-specific constants per object in LWIR range, not universal material constants, since real emissivity depends on wavelength, angle, texture, and material state.

4.2. Object-level implementation

Each object is assumed to have a single, uniform temperature and emissivity across its visible surface, or each object is approximated to a diffuse reflector, which assumes the radiation reflection intensity of the target is equal in all directions, as stated in [7].. Shadows, ambient lighting, and post-processing are disabled or excluded where they would interfere with the simplified thermal logic, since they belong conceptually to the visible-light domain. In the implementation, every renderable object is parametrised by two scalar values: its surface temperature TTT (in °C, converted internally to K) and its surface emissivity ε, the latter taken from a long-wave infrared emissivity table [8]. In the example scene (**Figure 2**) the wooden cube is assigned $\varepsilon = 0.76$ at $T = 25\ °C$ (ambient), the concrete floor $\varepsilon = 0.90$ at $T = 25\ °C$ (ambient), and the brick sphere $\varepsilon = 0.94$ at $T = 40\ °C$ (self-heated), consistent with emissivity values reported for rough concrete, wood, and fine brick in LWIR .

The emitted radiant exitance M for each object is computed once from Equation 1, using its assigned temperature and emissivity, and serves as a purely numerical proxy for radiative power. This per-object thermal value is then mapped into a colour emission term through Equation 2:

$$EmissionColor = Color.white \cdot \frac{M(T,\varepsilon)}{M(Tmax,1)} \quad (2)$$

In the example configuration $Tmax = 100°C$, so all object emissions are normalised to the radiative output of an ideal blackbody at 100 °C. The resulting $EmissionColor$ is finally written into the *_EmissionColor* property of the Unity Standard Shader material for that object, ensuring that the visible emission intensity in the rendered frame is a monotonic function of the underlying thermal state while remaining independent of conventional lighting and shading passes.



Figure 2. Raw emission values image of wooden cube, concrete floor and brick sphere in LWIR mode of Unity

4.3. Terrain-level implementation

Outdoor scenes require a terrain surface, handled differently from discrete mesh objects since Unity's Diffuse Terrain shader does not directly expose an emissive parameter in the same way as the Unity Standard shader. The terrain shader takes a single RGB texture and tiles it repeatedly across the terrain. Our implementation takes a texture of grass, soil, and occasional stones, which is then converted into a thermal proxy by editing the artwork so that each material category is recoloured to a normalised grey level proportional to its emissivity, using the mapping:

$$Color_{part} = Color.white \cdot M_{part} \quad (3)$$

In effect, stone regions appear as the brightest tones, soils intermediate, and grass as the darkest, reflecting emissivity-like behaviour in the 0.90–0.98 range for typical soils and vegetation in LWIR, as stated in [8] and [9]. The resulting emissivity texture is then assigned as the albedo input to the Diffuse Terrain shader and treated as a spatially varying emissivity proxy over the terrain surface.

Since Diffuse Terrain provides no emission colour property, a dedicated directional light source is introduced to act as the terrain's thermal emission proxy. This light is restricted to the terrain via Unity's layer system, so that it illuminates only the terrain layer and no other scene objects, thereby isolating the terrain contribution from the rest of the lighting pipeline. For the terrain we set a uniform surface temperature $Tmax = 30°C$ and reuse Equation 2, to compute an emission colour $EmissionColor$. Instead of writing this value into Standard shader property *_EmissionColor* as in section 4.2., the $EmissionColor$ is applied directly to the C*olor* property of the terrain's dedicated directional light. The light is configured to mimic pure emission rather than conventional illumination by disabling shadows and reflections and aligning its direction to be perpendicular to the terrain surface, yielding a terrain contribution that is visually consistent with the rest of the thermal model while remaining conceptually separate from the visible-light domain.

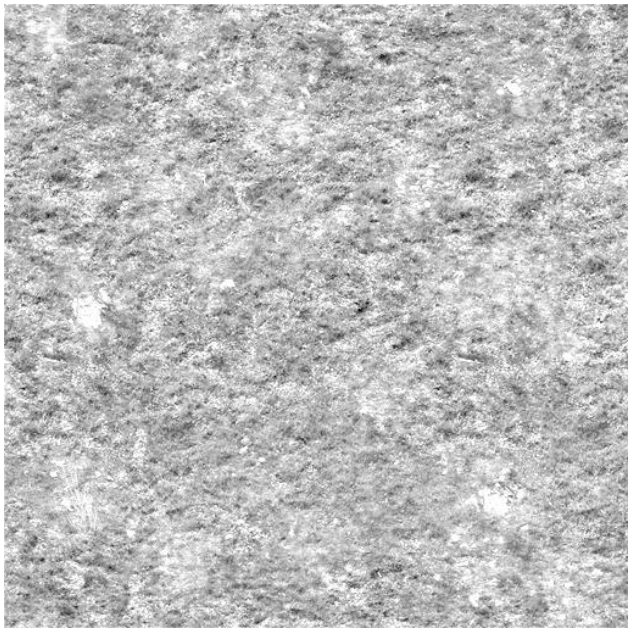


Figure 3. Emissivity texture of terrain used in example scene

4.4. Display mapping and thermal palettes

Once thermal values are computed for all objects and terrain, the scalar field is converted into a displayable image through a mapping stage. Pixel intensity is normalised using the minimum and maximum thermal values present in the current view frustum, so the coldest visible value maps to the darkest output and the hottest to the brightest. Because normalisation is per-frame, absolute brightness is not directly comparable across frames unless the underlying value range is also reported.

In the prototype implementation, the thermal field is rendered into a single render texture by Unity's camera, and a custom post-processing shader, *ReplaceColor*, is applied to this texture to perform the final colour mapping. For every pixel, *ReplaceColor* computes a normalised offset in the palette domain using its current blue-channel value (treated as the scalar thermal intensity) and the global minimum and maximum over the entire render texture,

$$offSetX = \frac{(inputTex2D.b * 255 - _minB)}{(_maxB - _minB)} \quad (4)$$

where *_minB* and *_maxB* are the minimum and maximum blue-channel values observed in the frame. *offSetX* acts as a lookup coordinate into a one-dimensional false-colour palette texture *_PalleteTex*. The final pixel colour is then obtained as

$$PixelColorValue = tex2Dlod(_PalleteTex, float4(offSetX, offSetY, 0, 0)) \quad (5)$$

where *offSetY* is a hard-coded offset selecting the active palette row (e.g. white-hot, red-hot, yellow-hot) within the palette texture. In this scheme the internal thermal output remains a single scalar intensity field, while the choice of palette texture and *offSetY* purely controls visualisation, following the behaviour of commercial thermal imaging systems where switching palettes changes only appearance and not temperature data.

The thermal output is represented internally as a single scalar intensity field. For display, it can be written as equal RGB values for grayscale output, or remapped through a false-colour palette such as white-hot, red-hot, or yellow-hot (Figure 4), following conventions common in commercial thermal software [10]. Palette choice affects only visualisation, not the underlying physical or computational content.

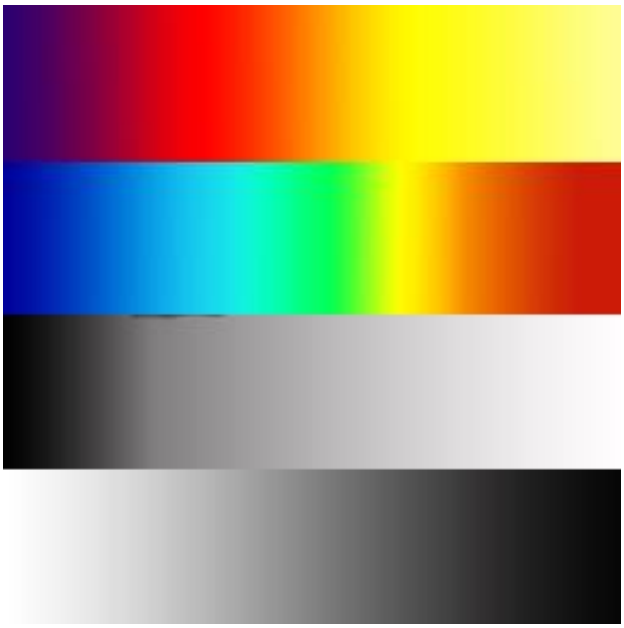


Figure 4. False colour thermal palettes, yellow hot - up, red hot - down

5. RESULTS

The examples below are illustrative demonstrations of feasibility, consistent with the paper's goal of establishing a starting point rather than a validated system. No quantitative sensor-level or task-based evaluation is performed here.

5.1. Uniform-material object scene

The first scene consists of simple geometric objects, each with a distinct material and a distinct temperature-emissivity pair, following Section 4.2. Concrete, wood, and brick surfaces were included, with object temperatures chosen to differ noticeably from one another. Under the normalisation procedure of Section 4.4., the rendered image shows visually distinguishable contrast between objects, driven entirely by Equation 1, without additional post-processing (Figure 5).

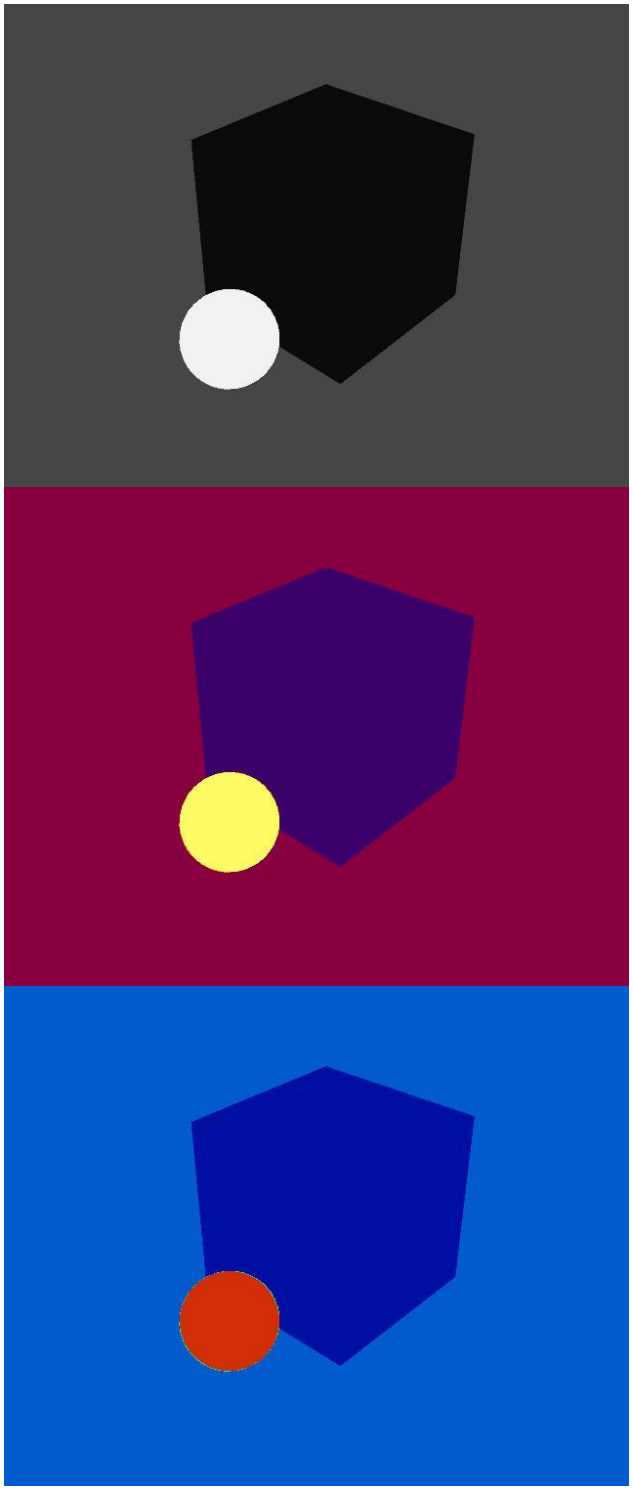


Figure 5. Synthetized LWIR image of example scene in White hot (top), Yellow hot (middle), and Red hot (bottom) colour modes, with normalization procedure

This scene isolates the core thermal computation and normalisation steps in a setting with no terrain or dynamic elements, so the basic mechanism's behaviour is not obscured by scene complexity.

5.2. Outdoor terrain scene with human figures

The second scene combines the object-level and terrain-level procedures of Sections 4.2 and 4.3. A real infrared photograph of soldiers in an open field served as a qualitative visual reference, not a ground-truth target for quantitative comparison. Soldier meshes were placed in a Unity terrain scene, assigned a common body and clothing temperature of approximately 36 degrees Celsius, converted to kelvin before use in Equation 1, together with approximate band-specific emissivity values for clothing and exposed skin of 0.88 (Fabric: Hessian, green) [8]. The terrain was treated with the thermal proxy texture and lighting workaround of Section 4.3., with grass, dirt, and stone assigned emissivity values in the 0.9 to 0.98 range and surface temperature of 30 degrees Celsius.

The rendered output shows human figures as distinctly brighter thermal forms against a cooler terrain background, qualitatively consistent with the reference photograph, where warm-bodied subjects stand out against cooler surroundings (Figure 6). This correspondence is offered strictly as evidence of plausibility, demonstrating that the method reproduces the basic figure-against-background contrast pattern typical of real thermal imagery of people outdoors, not that it reproduces exact radiometric values. Optional blur and synthetic sensor noise were identified as promising additions for improving perceptual resemblance to real thermal camera imagery, and are left for future work.

5.3. Interpretation and evaluation framing

Both scenes support plausibility only and are not sufficient to prove physical radiometric correctness. Appropriate evaluation depends on intended downstream use, and the present paper avoids committing to a single universal protocol that was not actually executed. For realism aimed at human observers, in simulation or training contexts, evaluation should rely on human judgement or structured realism review.

The image provided in [11] was used as a visual reference point (Figure 6). Top panel shows an LWIR image of several people on rough terrain, while the bottom panel shows a Unity-generated synthetic infrared image produced by the proposed method. Blur and noise were added in Photoshop to facilitate visual comparison with the reference image.

For algorithm testing or benchmarking, evaluation should rely on downstream task performance against an established benchmark protocol. Both strategies are left as future work.

6. CONCLUSION

This paper has proposed a deliberately simplified, first-order method for synthesizing infrared imagery in Unity 3D, and demonstrated through two simple examples that visible-range rendering infrastructure can be repurposed to produce structurally plausible thermal-like output. The method retains Unity's existing geometry, materials, and camera model unchanged, introducing a thermal computation stage based on a single-band (LWIR) Stefan-Boltzmann approximation, per-object temperature, and approximate emissivity, injected into the pipeline through the Standard Shader's emission property used strictly as a

rendering proxy.

The implementation is explicitly limited by object-wise uniform emissivity and temperature, a single spectral band (LWIR), and the omission of atmospheric transmission, reflected background radiation, detector calibration, sensor noise, optical blur, and angular or wavelength-dependent emissivity variation. Each of these omissions is identified as an addressable direction for future research.

Concrete extensions include material-class differentiation between metals, plastics, and fabrics through class-specific emissivity behaviour, incorporation of ambient thermal contributions from solar exposure and environmental temperature via Equation 1, and a move from qualitative plausibility toward quantitative evaluation through human-judgement studies or downstream task-based benchmarking. This work suggests that general-purpose real-time game engines are a viable and accessible foundation for synthetic infrared research, provided the physical scope and limitations of any such method are stated as clearly as they have been here.



Figure 6. LWIR image captured by IR camera (up) and LWIR generated image in Unity (down) with blur and noise added in post-process

Acknowledgment

Bojan Šmidling and Stefan Stojkov were supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, under Contract no. 451-03-33/2026-03/200034

References

[1] UPADHYAY, AVINASH & SHARMA, MANOJ & MUKHERJEE, PRERANA & SINGHAL, AMIT & LALL, BREJESH. (2025). A comprehensive survey on synthetic infrared image synthesis. Infrared Physics & Technology. 147. 105745. 10.1016/j.infrared.2025.105745.

[2] CARBONE, CARLOS & POTENA, CIRO & NARDI, DANIELE. (2020). Simulation of near Infrared Sensor in Unity for Plant-weed Segmentation Classification. 81-90. 10.5220/0009827900810090.

[3] ZHIJIAN H, BINGWEI H AND SHUJIN S (2022) An Infrared Sequence Image Generating Method for Target Detection and Tracking. Front. Comput. Neurosci. 16:930827. doi: 10.3389/fncom.2022.930827

[4] Unity Technologies, "Unity User Manual 2022.3 (LTS)", Accessed: Jul. 30, 2026. [Online]. Available: https://docs.unity3d.com/2022.3/Documentation/Manual/render-pipelines.html

[5] Unity Technologies, "Unity User Manual 2022.3 (LTS)", Accessed: Jul. 30, 2026. [Online]. Available: https://docs.unity3d.com/2022.3/Documentation/Manual/CamerasOverview.html

[6] ANDREW S. GLASSNER. 1994. Principles of Digital Image Synthesis. Morgan Kaufmann Publishers Inc.

[7] LI, BO & ZHAO, HUAI-CI & DU, MEI & LV, JINFENG. (2013). The Infrared Multispectral Image Simulation Research Based on 3D Scene. 10.2991/eeic-13.2013.32.

[8] ThermoWorks, "Emissivity Table", Accessed: Jul. 30, 2026. [Online]. Available: https://industrial-lab.thermoworks.com/blogs/learning-center/emissivity-table

[9] AN, NI & HEMMATI, SAHAR & CUI, YU-JUN. (2017). Assessment of the methods for determining net radiation on different time-scales of meteorological variables. Journal of Rock Mechanics and Geotechnical Engineering. 9. 10.1016/j.jrmge.2016.10.004.

[10] TOPDON Incorporated 2017, "How to Choose the Right Color Palette for Clearer Thermal Imaging", Accessed: Jul. 30, 2026. [Online]. Available: https://community.topdon.com/detail?id=892&type=1

[11] D. JEYABHARATHI, DR & DHARMA, DEJEY. (2018). Efficient background subtraction for thermal images using reflectional symmetry pattern (RSP). Multimedia Tools and Applications. 77. 10.1007/s11042-018-6220-1.