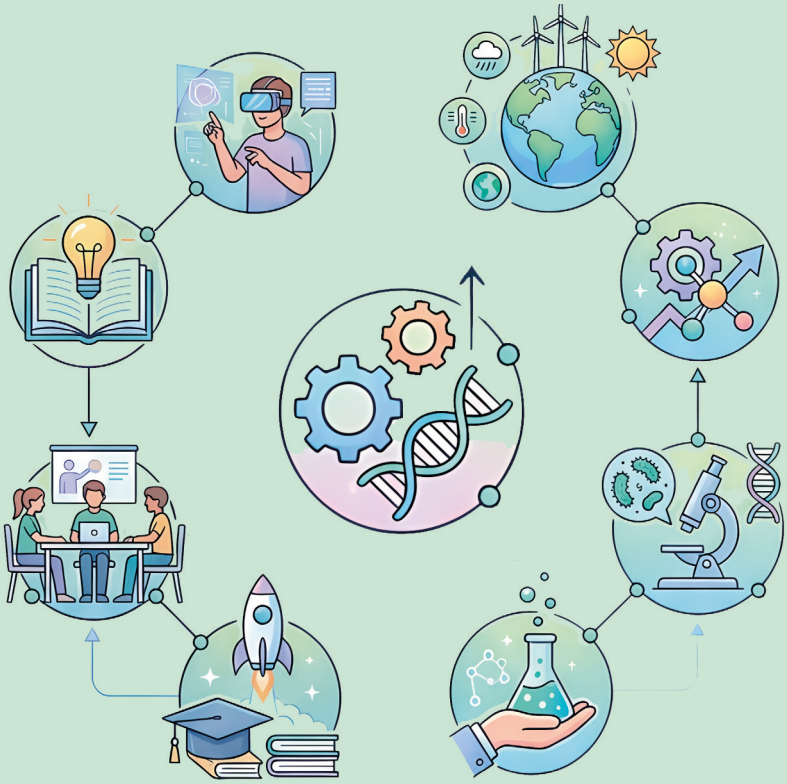
ISP
international science project

Sherman Oaks California (USA), 2026



INTELLECTUAL HORIZONS:

AN INTERDISCIPLINARY SYNTHESIS OF SCIENTIFIC KNOWLEDGE AND EDUCATIONAL INNOVATIONS

COLLECTIVE MONOGRAPH

INTELLECTUAL HORIZONS: AN INTERDISCIPLINARY SYNTHESIS OF SCIENTIFIC KNOWLEDGE AND EDUCATIONAL INNOVATIONS

Compiled by
VIKTOR SHPAK

Chairman of the Editorial Board
STANISLAV TABACHNIKOV

GS Publishing Services
Sherman Oaks
2026

The collective monograph is a scientific and practical publication that contains scientific articles by doctors and candidates of sciences, doctors of philosophy and art, graduate students, students, researchers and practitioners from European and other countries. The articles contain research that reflects current processes and trends in world science.

Text Copyright © 2026 by the Publisher «GS Publishing Services» and authors.
Illustrations © 2026 by the Publisher «GS Publishing Services» and authors.
Cover design © 2026 Publisher «GS Publishing Services».

Authors: Oksana Antolik, Olena Berezinska, Oleksii Bobko, Halyna Boikivska, Anastasia Boychuk, Alla Cherep, Oleksandr Cherep, Larysa Duda, Svitlana Faizulina, Dmytro Fedenko, Anatoliy Konokh, Andriy Konokh, Olena Konokh, Nina Korbozerova, Petro Krul, Oleksandr Lesko, Valentyna Moyseyenko, Anton Nosov, Oleksandr Novoseltsev, Andrii Nykula, Viktoriia Ogloblina, Lyudmila Oleynikova, Oleksandr Prytula, Natalia Punchenko, Oleksandr Romanyuk, Iryna Satsyuk, Anatolii Shamanskyi, Ivan Shcherbyna, Tetiana Skliaruk, Oleksii Stakhov, Dmytro Svetlyi, Ilona Tkachuk, Vira Tymoshenko, Valentyna Voronkova.

All rights reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means, or stored in a database or search engine without the prior written permission of the publisher. The authors are responsible for the content and reliability of their articles. Citation or other use of the monograph is possible only with reference to the publication.

Publisher «GS Publishing Services»
15137 Magnolia Blvd, # D,
Sherman Oaks, CA 91403, USA.

ISBN 979-8-9935428-7-4
DOI: 10.51587/9798-9935-42874-2026-029

Reviewers:
Hanna Kozhyna, Doctor of Medical Sciences, Professor, Head of the Department of Psychiatry, Narcology, Medical Psychology, and Social Work at Kharkiv National Medical University.
Volodymyr Verbytskyi, Doctor of Pedagogical Sciences, Professor, Director of the National Ecological and Naturalistic Center for Student Youth of the Ministry of Education and Science of Ukraine.

Scientific editors-reviewers:
S. Bobrovnyk, Yu. Bondar, A. Cherep, P. Glukhovskiy, P. Hovorov,
Yu. Kuznietsov, V. Lazurenko, V. Moyseyenko, L. Omelianchyk, Zh. Virna

The monograph is recommended for publication by the Academic Council of the Institute of Adult Education of the PO «National Academy of Sciences of Higher Education of Ukraine» (Protocol No. 6 dated September 23, 2026).

Intellectual Horizons: an Interdisciplinary Synthesis of Scientific Knowledge and Educational Innovations : collective monograph / Compiled by V. Shpak; Chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California : GS Publishing Services, 2026. 221 p.

Available at: DOI: 10.51587/9798-9935-42874-2026-029

CONTENT

COMPUTER SCIENCE

Oleksii BOBKO

METHODS OF CONTAINERIZING THE RENDERING OF THREE-DIMENSIONAL GRAPHIC IMAGES 6

Oleksandr ROMANYUK
Oleksii STAKHOV

METHODS FOR ACCELERATING THE RENDERING OF THREE-DIMENSIONAL SCENES 14

CULTUROLOGY

КРУЛЬ Петро Франкович,
ДУДА Лариса Ігорівна

ЦЕХОВА МУЗИКА МАГНАТСЬКИХ ТА КОЗАЦЬКО-СТАРШИНСЬКИХ МАЄТКІВ XVII – ПОЧАТКУ XIX СТОЛІТЬ 44

ECONOMIC SCIENCES

БОЙКІВСЬКА Галина Миколаївна

ТРАНСФОРМАЦІЯ ПІДХОДІВ ДО РОЗВИТКУ ТАЛАНТІВ В ОРГАНІЗАЦІЇ: ВІД УПРАВЛІННЯ ТАЛАНТАМИ ДО ФОРМУВАННЯ ЕКОСИСТЕМНОЇ МОДЕЛІ 54

ВОРОНКОВА Валентина Григорівна,
НІКІТЕНКО Віталіна Олександрівна,
ОГЛОБЛІНА Вікторія Олександрівна,
ПУНЧЕНКО Наталія Олегівна

ЦИФРОВА ТРАНСФОРМАЦІЯ МЕНЕДЖМЕНТУ В УМОВАХ НЕВИЗНАЧЕНОСТІ ТА ТУРБУЛЕНТНОСТІ: ВІД INDUSTRY 4.0 ДО MANAGEMENT 5.0 61

Oleksandr LESKO,
Anatolii SHAMANSKYI

INTELLIGENT CONTROL SYSTEMS FOR MANUFACTURING PROCESSES 76

СКЛЯРУК Тетяна Василівна

МІЖДИСЦИПЛІНАРНИЙ ПІДХІД ДО ФОРМУВАННЯ УПРАВЛІНСЬКИХ, ЦИФРОВИХ І СОЦІАЛЬНИХ КОМПЕТЕНТНОСТЕЙ МАЙБУТНІХ HR-ФАХІВЦІВ 86

ФЕДЕНКО Дмитро Олександрович

ПІДХОДИ ДО ВИКОРИСТАННЯ ВІЗУАЛЬНИХ КОНФІГУРАТОРІВ ПРИ ІМПЛЕМЕНТАЦІЇ ПОКАЗНИКІВ ІНТЕГРОВАНОГО ЗВІТУ ТА ЗВІТУ ЗІ СТАЛОСТІ..... 98

ЧЕРЕП Алла Василівна,
ОЛЕЙНІКОВА Людмила Григорівна,
НОСОВ Антон Дмитрович,
ЩЕРБИНА Іван Сергійович

СТРАТЕГІЯ СТАЛОГО РОЗВИТКУ ТА ЕКОЛОГІЧНА СТІЙКІСТЬ 121

Череп Олександр Григорович,
ФАЙЗУЛІНА Світлана Айгалівна,
СВІТЛИЙ Дмитро Олександрович,
БОЙЧУК Анастасія Сергіївна

ДОСВІД КРАЇН ЄС ЩОДО ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МАРКЕТИНГОВОЇ ДІЯЛЬНОСТІ ЦИФРОВИХ ТЕХНОЛОГІЙ 131

ЧЕРЕП Алла Василівна,
АНТОЛІК Оксана Миколаївна

ЦИФРОВІЗАЦІЯ ГРОМАДСЬКИХ ОРГАНІЗАЦІЙ ВЕЛИКОБРИТАНІЇ ТА УКРАЇНИ: ТЕНДЕНЦІЇ ТА ВИКЛИКИ 138

LINGUISTICS

БЕРЕЗІНСЬКА Олена Володимирівна

ІМЕРСИВНІ ТЕХНОЛОГІЇ ТА КЕЙС-МЕТОД У ФОРМУВАННІ ПРОФЕСІЙНО ОРІЄНТОВАНОЇ ІНШОМОВНОЇ КОМПЕТЕНТНОСТІ МАЙБУТНІХ ФАХІВЦІВ (НА ПРИКЛАДІ АНАЛІЗУ РИНКУ *BLEISURE-ТУРИЗМУ*) 145

КОРБОЗЕРОВА Ніна Миколаївна

МОВНА СИТУАЦІЯ ПІРЕНЕЇВ ДОРИМСЬКОГО ПЕРІОДУ 162

MEDICINE

МОЙСЕЄНКО Валентина Олексіївна,
НИКУЛА Андрій Тарасович

РОЛЬ ЛАКТОБАКТЕРІЙ У ПІДТРИМЦІ ГОМЕОСТАЗУ ОРГАНІЗМУ 167

PEDAGOGICAL SCIENCES

ТКАЧУК Ілона Львівна

КОГНІТИВНА МАТРИЦЯ СПРИЙНЯТТЯ: ЧОМУ КЛАСИЧНИЙ РИСУНОК І ЖИВОПИС ЗАЛИШАЮТЬСЯ НЕЗАМІННИМИ В ЕПОХУ DIGITAL ART 171

ПРИТУЛА Олександр Леонтійович,
КОНОХ Анатолій Петрович,
КОНОХ Олена Євгеніївна,
КОНОХ Андрій Анатолійович

ФІЛОСОФСЬКО-ПЕДАГОГІЧНИЙ ПОТЕНЦІАЛ КОЗАЦЬКОЇ СПАДЩИНИ У ФОРМУВАННІ ГОТОВНОСТІ МАЙБУТНІХ ФАХІВЦІВ ФІЗИЧНОЇ КУЛЬТУРИ І СПОРТУ ДО ЗАХИСТУ БАТЬКІВЩИНИ 176

POLITICAL AND LEGAL SCIENCES

ТИМОШЕНКО Віра Іванівна

ІСТОРИЧНИЙ ПОГЛЯД НА КРИМІНАЛІСТИЧНУ ІДЕНТИФІКАЦІЮ ОСОБИ ЗА БІОМЕТРИЧНИМИ ДАНИМИ 190

TECHNICAL SCIENCES

Oleksandr NOVOSELTSEV

RELIEF TEXTURING METHODS BASED ON REPRESENTING FREE SPACE WITH GEOMETRIC PRIMITIVES IN TWO-DIMENSIONAL TEXTURES 206

Oleksandr ROMANYUK,
Iryna SATSYUK

ANALYSIS OF DEVELOPMENT DIRECTIONS OF VIDEO CARD ARCHITECTURES 213

for ray tracing, distribution by samples or passes is effective; and for complex hybrid systems, a pipeline or service-based model can be used. The most universal scheme for high-performance rendering is based on long-running GPU containers with a dynamic queue of smaller graphics tasks.

A promising direction is the combination of containerization with predictive-adaptive balancing, in which the orchestrator uses historical data on the complexity of frame regions, GPU characteristics, cache state, and network locality. This makes it possible to regard container infrastructure not merely as a deployment mechanism.

DOI: 10.51587/9798-9935-42874-2026-029-6-14

Oleksandr ROMANYUK,
Doctor of Technical Sciences, Professor,
Vinnytsia National Technical University
ORCID iD: 0000-0002-2245-3364

Oleksii STAKHOV,
Senior Lecturer,
Vinnytsia National Technical University
ORCID iD: 0000-0002-4901-3211
Ukraine

METHODS FOR ACCELERATING THE RENDERING OF THREE-DIMENSIONAL SCENES

Rendering of three-dimensional scenes is the final and at the same time one of the most resource-intensive stages of computer graphics. At this stage, geometric representations of objects, material parameters, light sources, camera settings, textures, and auxiliary data structures are transformed into a two-dimensional image. Modern software systems impose contradictory requirements on rendering performance: realism must be increased, geometric complexity must grow, high image resolutions must be supported, and interactive frame rates must be maintained. Rendering acceleration should therefore be considered not as the optimization of one isolated algorithm, but as the coordinated design of an efficient graphics pipeline.

Performance improvement must begin with identifying the operations that form the critical computational path. In one scene the main limitation may be the processing of millions of polygons; in another it may be texture-memory traffic, complex illumination models, transparency, or repeated ray traversal. For real-time systems, the delay between a change in scene state and the appearance of the corresponding frame is also important. Consequently, optimization should rely on profiling, decomposition of frame time, and explicit control of image quality rather than only on reducing the number of arithmetic operations.

Research on the design of computer graphics systems emphasizes the need to coordinate architectural decisions with scene characteristics, illumination models, and the available hardware platform[1]. New approaches to high-performance rendering also require modularity, replaceable algorithmic blocks, and parallel execution[2]. This creates a methodological basis for combining rasterization, ray tracing, adaptive detail, spatial indexing, distributed computation, and machine-learning techniques within one software system.

The purpose of this monograph is to systematize methods for accelerating the rendering of three-dimensional scenes and to substantiate the principles of their combined application. Particular attention is paid to geometry optimization, culling, spatial search structures, rasterization and ray-tracing acceleration, illumination models, vector operations, levels of detail, load balancing, cloud and cluster rendering, image reconstruction, and adaptive scheduling. The expanded discussion also considers GPU-driven rendering, geometry clustering, memory and texture organization, temporal reuse, perceptual quality control, shader specialization, and the construction of a self-adjusting rendering architecture.

Frame formation time can be represented as a combination of costs associated with data preparation, geometry processing, construction or updating of spatial structures, rasterization or ray tracing, shading, post-processing, and output transfer. For a sequential abstraction one may write T_frame = T_CPU + T_geometry + T_spatial + T_visibility + T_lighting + T_post, whereas in a deeply pipelined system the actual frame time is largely determined by the longest

1 Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Novoseltsev O. O., Stakhov O. Y. Basic features of computer graphics system design. In: Teoretyko-metodolohichni zasady suchasnykh naukovykh doslidzhen [Theoretical and Methodological Foundations of Modern Scientific Research] : monograph. Kharkiv : SG NTM "Novyi Kurs", 2026. Section 4.5. Pp. 255-269. DOI: https://doi.org/10.61718/mon202601

2 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia : Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

parallel stage. This means that accelerating a noncritical stage may have almost no influence on frame rate. The first optimization method is therefore not code rewriting but accurate localization of the current bottleneck.

In geometry-rich scenes the main costs are caused by vertex, normal, and texture-coordinate transformations, clipping, tessellation, and primitive setup. In scenes with complex materials, pixel or fragment shading dominates. In ray tracing, a large share of time is spent on traversing acceleration structures, testing intersections, and generating secondary rays. In post-processing, the workload depends on resolution, the number of full-screen passes, and buffer formats. The same optimization can consequently have very different effects on different scenes.

Arithmetic, memory, and synchronization costs should be distinguished. A modern GPU can efficiently execute a very large number of similar operations, but its efficiency decreases under irregular memory access, branch divergence, cache misses, and frequent barriers. High theoretical throughput does not guarantee high application performance if the working set exceeds cache capacity or if neighboring threads follow different execution paths. Rendering optimization should therefore reduce the total amount of work, improve data locality, and increase useful parallelism simultaneously.

Practical profiling should be performed on a representative set of scenes: a geometrically complex scene, a material-intensive scene, a scene with many light sources, a transparency-heavy scene, and a ray-traced scene. Results should be recorded not only as frames per second but also in milliseconds per frame because time values correctly describe absolute improvement. A reduction from 20 ms to 16 ms per frame is a directly interpretable 20% reduction in frame time, while FPS differences are nonlinear with respect to the initial value.

Table 1

Main groups of rendering-acceleration methods

Sourse: Authors

Method group	Target stage	Main effect	Typical limitations
1	2	3	4
Culling, LOD	Geometry	Reduction in the number of processed primitives	Detail error, visible LOD transitions
BVH, k-d tree	Visibility and rays	Reduction in intersection tests	Build and update cost

Continuation of table 2

1	2	3	4
BRDF optimization	Lighting	Less arithmetic and fewer samples	Risk of reduced realism
Parallelization	Entire pipeline	Higher throughput	Synchronization and imbalance
ML reconstruction	Post-processing	Fewer samples or rendered pixels	Artifacts and model dependence

The most direct way to accelerate a geometry-intensive scene is to prevent primitives that cannot influence the frame from entering later stages of the pipeline. This is achieved by combining model preprocessing, frustum culling, back-face culling, occlusion culling, and levels of detail. Every triangle rejected at an early stage saves work in vertex processing, primitive assembly, rasterization, fragment shading, and memory access. Early reduction of geometry therefore has a multiplicative effect on the entire pipeline.

Frustum culling evaluates an object relative to the six planes of the viewing volume. Instead of testing every triangle, it is more efficient to test a bounding sphere, axis-aligned bounding box, or oriented bounding box first. If the entire bound lies outside the frustum, the object is removed without internal geometry analysis. Static scenes benefit further from pre-grouping objects into spatial nodes. Excessively coarse grouping, however, reduces culling accuracy, whereas excessively fine grouping increases the number of tests and command-management overhead.

Occlusion culling removes objects that lie inside the view frustum but are completely hidden by other surfaces. In large interiors, urban environments, and industrial models, this technique can produce a larger gain than frustum culling alone. Efficient implementations should minimize the latency of visibility decisions and avoid synchronous readback from the GPU to the CPU. Hierarchical depth buffers and conservative visibility tests are effective because they enable early rejection without forcing the processor to wait for a completed graphics query.

Reducing the number of draw calls is a separate optimization problem. Even when the GPU has enough raw capacity to process the geometry, thousands of small submissions can produce a CPU bottleneck. Static batching, instancing, multi-draw commands, and indirect drawing move a larger part of command

generation toward the GPU. Aggressive batching must nevertheless be coordinated with culling: one very large batch is difficult to reject partially, so the optimal granularity depends on scene structure and camera movement.

Studies of high-performance rendering architectures support adaptive selection of geometric complexity rather than one fixed level for all conditions[3]. A practical criterion can combine projected object size, distance, surface curvature, silhouette importance, and the current frame-time budget. With this approach, LOD becomes not only an artistic asset-management technique but also a mechanism for active control of computational workload.

Spatial data structures are fundamental to reducing the number of tests in ray tracing, visibility determination, and collision-related queries. Without indexing, a ray potentially tests its intersection with every primitive in the scene, which becomes unacceptable for complex models. Bounding-volume hierarchies, k-d trees, octrees, and regular grids reduce the search space by rejecting large groups of primitives at once. The best structure depends on the proportion of static and dynamic objects, the spatial distribution of geometry, and the dominant type of query.

BVH is particularly suitable for ray tracing because it works well with nonuniform geometry and supports hierarchical rejection. Hierarchy quality is determined not only by depth but also by how effectively primitives are distributed among child nodes. Surface-area heuristics estimate expected traversal cost and often reduce the total number of node and primitive tests. Dynamic scenes introduce another dimension: a high-quality hierarchy may be too expensive to rebuild every frame, so fast refitting of an existing tree can outperform complete reconstruction in total frame cost.

Memory locality is critical during traversal. Nodes that are logically adjacent should be stored compactly to reduce random memory access. Linearized BVHs, dense index arrays, and spatial ordering of primitives can reduce cache misses and improve coalesced GPU access. Packet traversal benefits from rays with similar directions, although secondary rays gradually lose coherence. A modern implementation should therefore treat structure layout as part of the algorithm rather than as a secondary storage detail.

3 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia : Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

In hybrid systems the same spatial structure can support multiple functions: shadows, reflections, nearest-hit queries, ambient occlusion, and some visibility tests. Reuse reduces duplication of data but increases the need for a stable interface between the rendering modules and the acceleration-structure service. This separation is architecturally valuable because it allows the underlying structure to be replaced or tuned without rewriting high-level shading code.

Rasterization remains the basis of most interactive graphics because it converts geometric primitives into fragments with extremely high throughput. Its acceleration is determined less by the coverage test itself than by data organization and the amount of work performed for each fragment. If an expensive pixel shader is executed for a fragment that is later rejected by a depth test, the computation is wasted. Early depth testing, a depth pre-pass where justified, and front-to-back ordering of opaque geometry can significantly reduce such work.

The ratio between the number of vertices and the number of shaded pixels is important. Very small triangles increase setup and vertex-processing overhead while covering few pixels. Large triangles shift the workload toward fragment shading. Optimization should identify which mode dominates. For geometry-limited workloads, mesh simplification, meshlets, and fewer state changes are useful; for fragment-limited workloads, shader simplification, fewer texture samples, adaptive effect quality, and reconstruction from a lower internal resolution are more effective.

Pipeline-state changes and frequent transitions between resources also influence performance. Materials and objects are therefore commonly sorted by shared state to reduce expensive switches. Such sorting must preserve transparency correctness and should not undermine culling. A contemporary rendering architecture should treat the command stream as an optimized sequence of tasks rather than as a literal traversal of the scene graph[4].

For multipass rendering, the cost of writing and rereading intermediate buffers must be considered explicitly. Deferred shading is convenient with many lights but creates significant G-buffer bandwidth. Forward+ and clustered

4 Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Novoseltsev O. O., Stakhov O. Y. Basic features of computer graphics system design. In: Teoretyko-metodolohichni zasady suchasnykh naukovykh doslidzhen [Theoretical and Methodological Foundations of Modern Scientific Research] : monograph. Kharkiv : SG NTM "Novyi Kurs", 2026. Section 4.5. Pp. 255-269. 299 p. DOI: https://doi.org/10.61718/mon202601

shading can reduce intermediate storage and better support transparency. The preferred technique should be selected from measurements on target scenes, taking into account the number of lights, material variety, memory bandwidth, and the need for post-processing data.

Ray tracing provides a natural formulation for reflections, refractions, shadows, and global illumination, but it requires a very large number of intersection operations. In physically based rendering, each pixel may generate multiple primary and secondary rays and the variance decreases only as additional samples are taken[5]. Acceleration therefore depends on three coordinated strategies: reduce the number of rays, reduce the cost of traversing the scene, and reconstruct acceptable image quality from fewer samples.

The first optimization level is an efficient acceleration structure. The second is early termination when further contribution to the result becomes negligible or when any occluder is sufficient for a shadow query. The third is limiting maximum path depth and applying probabilistic path termination. In offline physically based rendering, these decisions should preserve estimator correctness; in real-time rendering, carefully controlled approximations are acceptable when the visual error remains below the perceptual threshold.

Packet execution and ray coherence can improve the efficiency of SIMD and SIMT hardware. Primary rays from neighboring pixels have similar directions and tend to traverse similar BVH nodes. After diffuse bounces, coherence decreases and thread divergence grows. Sorting or regrouping rays by direction, material, or spatial region may restore some coherence, but reordering itself has a cost. The method is useful only when the reduction in traversal divergence outweighs the overhead of queue management.

Hybrid rendering uses ray tracing where it has the largest visual advantage: accurate reflections, soft shadows, selected occlusion effects, and limited global-illumination components. Primary visibility can remain rasterized while a small number of rays is denoised or temporally accumulated. This shifts the design objective from exact simulation of every light path toward efficient acquisition and reconstruction of the image information that matters most.

The cost of illumination grows with the complexity of the reflectance function, the number of lights, the number of texture samples, and the

5 Pharr M., Jakob W., Humphreys G. Physically Based Rendering: From Theory to Implementation. 4th ed. Cambridge, MA: The MIT Press, 2023. 1280 p.

set of secondary effects. The choice of a BRDF must therefore account for computational cost in addition to physical plausibility. Romanyuk and co-authors have proposed and analyzed reflection models aimed at combining realism and efficiency[6],[7]. A practical rendering system benefits from several quality variants of a model that can be selected according to projected object size and available frame-time budget.

Lighting optimization begins by reducing the number of sources that actually affect a fragment. Screen-space or world-space clustering makes it possible to construct a compact list of relevant lights for each region. Instead of looping over every light in the scene, the shader processes only the local subset. The main engineering challenge is to build these lists efficiently without creating a CPU bottleneck or excessive memory fragmentation.

Another direction is to replace expensive transcendental operations with approximations, lookup tables, or pre-integrated functions. Simplification should be evaluated not only by numerical error of the BRDF itself but also by the effect on the final image. Small analytical errors can become conspicuous in specular highlights, grazing angles, silhouettes, or high-dynamic-range lighting. Performance decisions should therefore be based on image-space error as well as instruction count.

Image-based lighting demonstrates a general acceleration principle: repeated integrations can often be moved from frame time to preprocessing. Environment maps are prefiltered for different roughness levels and some parts of the lighting equation are represented by lookup textures. Complex integration is then replaced with a small number of texture reads. The same idea can be applied to static visibility, shadow data, material terms, and other slowly changing quantities.

Vector operations are basic to geometric transformations, lighting, direction calculations, and normal processing. Vector normalization is especially frequent and requires a square root and division in its direct form. Repetition of this operation millions of times makes it a meaningful optimization target. Work on

6 Romanyuk O., Zavalniuk Y., Romanyuk O., Kotlyk S., Bobko O., Melnyk O. New Illumination Model for Rendering. In: 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 17-19 September 2025. IEEE, 2025. Pp. 723-727. DOI: https://doi.org/10.1109/ACIT65614.2025.11185843

7 Romanyuk O., Lytvynenko V., Zavalniuk Y. Modified cosine-quadratic reflectance model. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska. 2025. Vol. 15, No. 1. Pp. 45-48. DOI: https://doi.org/10.35784/iapgos.7384

accelerated vector normalization for rendering tasks proposes techniques for reducing computational cost while maintaining acceptable accuracy[8].

A software system should distinguish operations that require high accuracy from those that tolerate approximation. Normals used in sharp specular lighting are more sensitive to error than auxiliary directions used for coarse visibility estimates. This makes it possible to provide several implementations of the same operation: an accurate variant, a fast approximate variant, and a batch SIMD/SIMT variant. The selected path can depend on material, effect, and quality mode.

Attribute interpolation also has a significant accumulated cost. Perspective-correct texture mapping requires reciprocal depth, and a large number of variables transferred from the vertex stage to the fragment stage increases interpolator traffic. Optimization consists of transmitting only the attributes that are actually needed, selecting compact formats, and reconstructing some quantities in the shader when arithmetic is cheaper than bandwidth. The correct balance depends on the microarchitecture and memory hierarchy of the target GPU.

Micro-optimizations that complicate code without measurable effect should be avoided. Shader compilers already perform many local transformations. Larger gains usually come from changing the algorithm: reducing the number of normalizations, moving invariant values out of loops, evaluating slowly varying terms per vertex instead of per pixel, or reusing previously calculated results. Optimization must remain measurement-driven.

Levels of detail are based on the principle that geometric and texture accuracy should correspond to an object's actual contribution to the image. If an object occupies only a few dozen pixels, millions of polygons cannot be visually resolved and merely consume resources. Classical LOD uses prebuilt versions of a model, while more flexible systems consider screen-space error, distance, camera motion, and object importance. This converts scene representation into a controllable performance variable.

LOD switching should avoid visible discontinuities. Geometric morphing, cross-fading, and threshold hysteresis can reduce popping. Shadows and

8 Romanyuk O., Zavalniuk Y., Mykhaylov P., Chekhmestruk R., Titova N., Romanyuk S. Accelerated Vector Normalization for Rendering Tasks. In: 2024 14th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 19-21 September 2024. IEEE, 2024. Pp. 640-644. DOI: https://doi.org/10.1109/ACIT62333.2024.10712602

reflections must also be considered: an object may be small in the main camera yet occupy a large region in a reflection or shadow map. Different passes can therefore use different detail policies. The objective is not maximum simplification but rational allocation of the geometric budget according to visible significance.

Adaptive tessellation is the opposite but related operation: geometry is refined only where silhouettes, displacement, or close-up inspection require additional detail. Tessellation factors must be bounded because uncontrolled generation of micro-polygons can create a severe geometry bottleneck. LOD, tessellation, and resource streaming should operate together so that data not used at the current quality level is neither processed nor unnecessarily loaded into memory.

For distant objects, impostors, billboards, and pre-rendered representations can replace full geometry. Their limitations include changing illumination, viewpoint dependence, and parallax, but multi-view or layered representations can reduce these problems. The important methodological point is that acceleration can come from changing the representation of the scene itself, not only from making an existing algorithm faster.

Parallelism is inherent in rendering because many vertices, fragments, or rays can be processed independently. A GPU exploits this property with thousands of concurrent threads. On the CPU side, scene preparation, animation, culling, command construction, resource streaming, and parts of acceleration-structure building can also be distributed across cores. The essential requirement is to remove unnecessary synchronization. A single globally locked scene structure can eliminate the practical benefit of otherwise parallel stages.

Amdahl's law shows that speedup is limited by the sequential portion of work: $S = 1 / ((1 - p) + p/N)$, where p is the parallel fraction and N is the number of execution units. If ten percent of the workload remains strictly sequential, unlimited scaling of the rest cannot produce a speedup larger than ten. For rendering systems this means that optimizing submission, scheduling, synchronization, and serial resource management can be as important as adding more computational units.

Distributed rendering can be decomposed by frames, image tiles, objects, rays, or lighting tasks. Frame-level distribution is simple but does not reduce latency of a single frame. Tile-based decomposition processes one image

in parallel but introduces load imbalance because a tile with sky may be far cheaper than a tile containing transparency, dense geometry, and reflections. Dynamic work allocation is therefore more effective than a fixed equal partition in heterogeneous scenes.

Networked systems add the cost of transferring scene data, textures, updates, and results. Effective systems use local caches, resource versioning, and delta transfer rather than repeatedly sending complete data sets. Static scenes benefit from pre-replication of resources. Dynamic scenes require compact update formats and, where possible, prediction of resources that will be required by subsequent frames.

A static division of the image into equal regions is simple but does not guarantee equal processing time. Real complexity depends on visible primitives, depth complexity, materials, transparency, shadows, and the number of rays. A method for balanced loading of rendering units for three-dimensional scene formation emphasizes the need to account for heterogeneous computational complexity across different frame regions[9]. In practice, task weight should be based on expected work rather than only on the number of pixels.

A tile weight can be modeled as W = aN_tri + bN_light + cN_trans + dN_ray + eT_hist, where the coefficients are determined through profiling and T_hist is previously observed time for a similar region. The model does not need to be exact; it only needs to rank relative complexity better than an equal-area assumption. A dynamic scheduler can then assign the most expensive available work to idle workers and reduce the tail latency of the frame.

Work stealing is another useful strategy. Each worker maintains a local queue but can take tasks from an overloaded neighbor when it becomes idle. This approach scales well for many independent tasks, yet excessively fine granularity increases scheduling and synchronization overhead. Task size should be large enough to use the GPU or CPU efficiently but small enough to prevent one exceptionally complex task from determining completion time of the entire frame.

Heterogeneous systems must account for different processing rates of individual nodes. Giving the same number of tiles to two GPUs from

9 Bobko O. L., Romanyuk O. N. Method of balanced loading of renderers in the formation of three-dimensional graphic scenes. Naukovi pratsi Donetskoho natsionalnoho tekhnichnoho universytetu. Seriia: Informatyka, kibernetyka ta obchysliuvalna tekhnika [Scientific Works of Donetsk National Technical University. Series: Informatics, Cybernetics and Computer Science]. 2026. No. 1 (42). Pp. 119-126. DOI: https://doi.org/10.31474/1996-1588-2026-1-42-119-126 (In Ukrainian).

different generations is not balanced. A scheduler should estimate observed throughput for every executor and adjust workload share dynamically. For long jobs, the performance model can be updated during execution. Load balancing thus becomes a feedback process: predict, execute, measure, and correct.

Cloud rendering provides elastic access to computation and makes it possible to scale a rendering task according to current demand. In offline visualization, this can mean parallel computation of animation frames; in interactive applications, it may mean remote rendering with video delivery to a client. The benefit depends on network latency, data-transfer cost, resource startup time, and the scalability of the rendering workload. Elasticity alone does not eliminate algorithmic bottlenecks.

Offline animation is particularly suitable for frame-level distribution because network overhead is amortized over long computations and the output of each worker is comparatively small. A single complex frame requires decomposition by image region or samples, which in turn requires shared scene resources and deterministic aggregation. Large texture sets should be cached near compute nodes to avoid repeated transfer.

Interactive remote rendering is constrained by end-to-end latency. Even a very fast server cannot compensate for an unstable network when immediate response is required. The relevant quantity is the sum T_input + T_network1 + T_render + T_encode + T_network2 + T_decode. Hardware video encoding, adaptive resolution, motion prediction, and selective local execution can reduce the total. A cloud system should be optimized for latency variance as well as average delay.

Cluster rendering also requires fault tolerance. Long-running tasks should be divided into recoverable units so that lost work can be reassigned. Determinism is valuable because the same tile or sample range should produce a reproducible result from identical input data. Deterministic task boundaries simplify caching, error detection, and partial re-execution.

Machine-learning methods alter the formulation of rendering acceleration. Instead of computing every high-quality pixel directly, a system can generate a cheaper intermediate result and reconstruct missing information with a learned model. Research on machine learning for accelerated rendering identifies neural reconstruction, adaptive sampling, and denoising as promising ways to reduce

cost[10]. In most practical architectures, the learning module optimizes a specific expensive stage rather than replacing the complete graphics pipeline.

Neural upscaling allows the scene to be rendered at a lower internal resolution and then reconstructed to the target resolution. The gain arises because a large part of fragment shading, ray tracing, and post-processing scales with pixel count. Reconstruction quality improves when the network receives motion vectors, depth, normals, and information from previous frames. The main engineering risks are temporal trails, flicker, errors on thin geometry, and instability in newly revealed regions.

Neural denoising is particularly important for ray tracing. It reduces the required samples per pixel and converts a noisy intermediate image into a stable approximation. The denoiser must not invent structurally incorrect detail; auxiliary buffers, temporal history, and carefully selected training data are therefore important. In scientific visualization, where image structures can have quantitative meaning, learning-based reconstruction must be validated against domain-specific error criteria as well as visual appearance.

Adaptive sampling uses a prediction of uncertainty or local variance. More rays or samples are directed toward complex and unstable regions, while fewer are assigned to smooth areas. The same complexity map can assist load balancing because it predicts which tiles will be expensive. Machine learning can therefore become part of the scheduling loop, not merely a final image filter[11].

No single rendering method is optimal for every component of an image. Rasterization is extremely efficient for primary visibility, ray tracing naturally represents many optical effects, and reconstruction can compensate for sparse sampling. Precomputed structures can accelerate static illumination components. A practical high-performance renderer therefore combines methods and assigns each visual component to the algorithm with the best quality-to-cost ratio.

A typical hybrid frame can use rasterization to generate primary surface information, clustered lighting for local sources, a limited number of rays

10 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

11 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California : GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

for reflections and shadows, spatio-temporal denoising for stabilization, and upscaling to restore the target output resolution. The challenge is to keep the data produced by these stages consistent. Depth, normals, geometric boundaries, motion vectors, and material identifiers must refer to the same scene state, otherwise reconstruction produces ghosting and edge artifacts.

Hybridization also applies to quality. Important objects, the central region of the frame, or a still-image capture mode may use more expensive illumination while peripheral or fast-moving regions use a simplified path. In virtual reality this principle is related to foveated rendering. Spatially nonuniform quality is effective only when transitions remain visually unobtrusive and the quality policy does not create temporal instability.

The software architecture must support such combinations without duplicating the complete renderer. Modularity and clear interfaces between subsystems are central to architectures for high-realism and high-performance rendering[12]. Each module should return not only its image contribution but also information about execution cost, confidence, or residual error so that the scheduler can adapt the next frame.

A suitable architecture divides the renderer into scene management, visibility, geometry preparation, illumination, tracing, post-processing, and profiling subsystems. Scene management maintains resources and versions. The visibility module performs culling and LOD. Geometry preparation forms batches or meshlets. The illumination module constructs light lists and evaluates BRDF parameters. Tracing uses a shared acceleration-structure service. Post-processing performs reconstruction, while the profiler returns timing statistics to the scheduler[13].

The core principle is a closed optimization loop. During frame k the system measures stage times and resource use. Before frame k+1, the scheduler

12 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia: Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

13 Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Novoseltsev O. O., Stakhov O. Y. Basic features of computer graphics system design. In: Teoretyko-metodolohichni zasady suchasnykh naukovykh doslidzhen [Theoretical and Methodological Foundations of Modern Scientific Research]: monograph. Kharkiv: SG NTM "Novyi Kurs", 2026. Section 4.5. Pp. 255-269. 299 p. DOI: https://doi.org/10.61718/mon202601; Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia: Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

compares these measurements with the target budget. If the budget is exceeded, the cost of the dominant stage is reduced first: LOD may be decreased, ray count lowered, effect resolution reduced, or BRDF complexity simplified. If there is stable headroom, quality can be increased. This targeted response is more efficient than reducing every quality parameter equally.

Stable and rapidly changing control variables should be distinguished. LOD and internal resolution can change relatively often, while rebuilding large data structures or changing shader sets may have higher cost and should occur less frequently. Hysteresis and filtered metrics are useful for preventing oscillation. For example, a quality reduction may be triggered only after several consecutive frames exceed the budget, while quality is restored only after stable spare capacity is observed.

The architecture should support asynchronous operations. Texture loading, shader compilation, partial structure building, and data copying should not block the main thread when dependencies permit parallel execution. Task queues, worker pools, and dedicated GPU queues make this possible. Asynchronous execution is not free, however: dependency tracking and resource-state management must be explicit to avoid hidden waits that merely move synchronization to a less visible location.

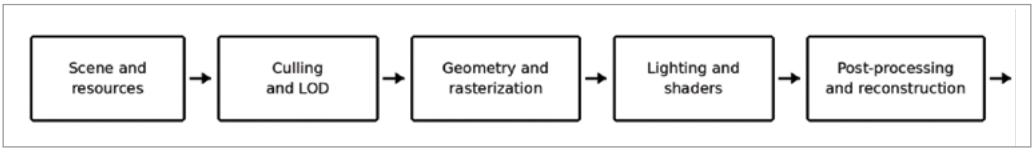


Figure 1. Generalized structure of an accelerated rendering pipeline

A correct evaluation must separate genuine acceleration from random variation. Before measurement, caches should be warmed, shader compilation completed, and the clock-frequency regime stabilized. For every scene, the distribution of frame times should be recorded rather than only the average. The 95th and 99th percentiles reveal rare but user-visible stalls. In interactive systems, stable latency is often more important than the maximum average frame rate.

Methods should be compared at equal quality or with an explicitly stated quality difference. If one variant uses a lower resolution or fewer rays, the result is not pure algorithmic acceleration but a quality-for-speed trade-off. Such trade-offs are legitimate, but they require two metric groups: performance and

quality. PSNR, SSIM, perceptual metrics, temporal stability measures, and expert assessment may be combined; geometric LOD can additionally be evaluated by screen-space silhouette or normal error.

The contribution of individual optimizations should be measured using ablation experiments. LOD, culling, accelerated normalization, denoising, temporal reuse, or load balancing can be disabled one at a time. This reveals which component produces the observed gain. When several techniques interact, their combined speedup may be smaller than the sum of separate gains because the dominant bottleneck changes after each successful optimization.

For reproducibility, the driver version, graphics API, GPU and CPU models, memory capacity and frequency, output resolution, quality parameters, ray count, denoiser version, scene revision, and camera trajectory should be fixed. Recorded camera paths allow the same workload to be replayed. This is particularly important for adaptive systems because scene complexity varies during movement and a single static view may not expose the difficult frames.

If a renderer is CPU-limited, priorities include reducing draw-call overhead, using instancing, preparing commands in parallel, simplifying scene-management logic, and minimizing synchronization. If the geometry stage on the GPU is the bottleneck, LOD, meshlets, culling, and compact vertex formats are appropriate. If fragment shading dominates, overdraw, material complexity, light counts, texture sampling, and internal resolution should be examined. This classification keeps optimization focused on the active constraint.

For ray tracing, the first checks are the quality and update strategy of the BVH, ray count, path depth, divergence, and denoiser efficiency. For distributed rendering, attention shifts to task imbalance, network traffic, and worker idle time. For cloud-interactive rendering, the complete latency path should be decomposed into network, rendering, encoding, and decoding components. The largest contribution should be optimized first, not simply the component that is easiest to modify.

Methods proposed by Romanyuk and co-authors demonstrate that performance reserves exist at multiple levels: pipeline organization[14], software

14 Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Novoseltsev O. O., Stakhov O. Y. Basic features of computer graphics system design. In: Teoretyko-metodolohichni zasady suchasnykh naukovykh doslidzhen [Theoretical and Methodological Foundations of Modern Scientific Research]: monograph. Kharkiv : SG NTM "Novyi Kurs", 2026. Section 4.5. Pp. 255-269. 299 p. DOI: https://doi.org/10.61718/mon202601

architecture[15], load balancing[16], machine learning[17], reflectance models[18], and basic vector operations[19]. This supports a multilevel optimization strategy. The strongest result should be expected from successive removal of bottlenecks with repeated measurement, not from one isolated "universal" technique.

The final criterion is not the minimum number of operations but fulfillment of functional requirements within a defined time budget. In an interactive system this means stable frame time; in offline rendering, minimum time to the required quality; in cloud services, a balance of speed, monetary cost, and throughput; and in scientific visualization, preservation of visual and numerical fidelity. Acceleration must therefore be evaluated in the context of the target scenario.

Traditional rendering pipelines rely on the CPU to traverse the visible scene and submit a large number of commands. This model becomes inefficient when scenes contain hundreds of thousands of potentially visible objects. GPU-driven rendering transfers visibility filtering, command compaction, and often LOD selection to compute shaders or mesh-processing stages. The CPU submits a small number of persistent or indirect command buffers, while the GPU determines which objects and clusters should actually be drawn. The advantage is not only lower submission overhead but also improved scalability when the visible set changes rapidly from frame to frame.

15 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia: Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

16 Bobko O. L., Romanyuk O. N. Method of balanced loading of renderers in the formation of three-dimensional graphic scenes. Naukovi pratsi Donetskoho natsionalnoho tekhnichnoho universytetu. Seriia: Informatyka, kibernetyka ta obchysliuvalna tekhnika [Scientific Works of Donetsk National Technical University. Series: Informatics, Cybernetics and Computer Science]. 2026. No. 1 (42). Pp. 119-126. DOI: https://doi.org/10.31474/1996-1588-2026-1-42-119-126 (In Ukrainian).

17 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

18 Romanyuk O., Zavalniuk Y., Romanyuk O., Kotlyk S., Bobko O., Melnyk O. New Illumination Model for Rendering. In: 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 17-19 September 2025. IEEE, 2025. Pp. 723-727. DOI: https://doi.org/10.1109/ACIT65614.2025.11185843; Romanyuk O., Lytvynenko V., Zavalniuk Y. Modified cosine-quadratic reflectance model. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska. 2025. Vol. 15, No. 1. Pp. 45-48. DOI: https://doi.org/10.35784/iapgos.7384

19 Romanyuk O., Zavalniuk Y., Mykhaylov P., Chekhmestruk R., Titova N., Romanyuk S. Accelerated Vector Normalization for Rendering Tasks. In: 2024 14th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 19-21 September 2024. IEEE, 2024. Pp. 640-644. DOI: https://doi.org/10.1109/ACIT62333.2024.10712602



An efficient GPU-driven pipeline usually stores object descriptors in compact arrays. A compute pass tests bounding volumes against the frustum and hierarchical depth information, selects LOD, and writes identifiers of visible items into a compacted list. A second stage converts this list into indirect draw or dispatch commands. Because the entire operation remains on the GPU, there is no synchronous readback of the visible set. The design must, however, reserve enough command-buffer capacity and handle the case where conservative estimates overallocate memory.

Data organization is critical. Object descriptors should separate frequently accessed fields from rarely used metadata so that culling does not fetch complete material or animation records. Dense identifiers and structure-of-arrays layouts improve access when thousands of neighboring threads read the same field. Stable indices are also useful because they allow visibility history and occlusion confidence to be preserved across frames. The GPU-driven approach therefore depends as much on data-oriented architecture as on the use of indirect commands.

The main limitation is reduced simplicity of debugging and synchronization. Resource barriers between culling, command generation, and drawing must be correct, and the system must prevent one frame from overwriting indirect data still used by another. Double or triple buffering of command lists is often appropriate. A well-designed GPU-driven renderer can be integrated with the modular architecture described above by treating command generation as a visibility product rather than as a CPU-side scene traversal step.

Large scenes benefit from partitioning geometry into small clusters, commonly called meshlets, that contain a limited number of vertices and triangles. Such clusters provide a convenient unit for culling, LOD selection, streaming, and parallel dispatch. Instead of processing an entire mesh because one part is visible, the renderer can reject or select only the relevant clusters. This increases culling precision while keeping the number of scheduling units manageable.

A cluster can store a compact local vertex index set, a bounding sphere or box, a normal cone, material identifiers, and links to neighboring detail levels. The normal cone allows back-facing clusters to be rejected before triangle processing, while the bound supports frustum and occlusion tests. If cluster metadata is small enough to remain cache-resident, visibility decisions become

inexpensive. The quality of clustering affects both locality and culling efficiency; clusters that are spatially incoherent reduce the benefit even if they satisfy a fixed triangle count.

Geometry streaming should operate at the same granularity. When the camera approaches a region, the system requests the clusters needed by the selected LOD rather than loading a complete high-resolution object. A residency manager tracks which blocks are present in GPU memory and which can be evicted. Prefetching based on camera velocity, scene topology, or visibility history can hide transfer latency. Conversely, overly aggressive prefetching wastes bandwidth and may evict useful texture or geometry data.

The strongest architecture combines cluster-based geometry with asynchronous storage and GPU-driven visibility. The visibility pass identifies missing high-detail clusters, the streaming system schedules them for future frames, and a coarser resident representation is used until loading completes. This avoids hard stalls and supports very large scenes whose full geometry does not fit in graphics memory. LOD is thus connected directly with memory management and is no longer a purely geometric decision.

Consecutive frames in an interactive renderer are highly correlated. Most surfaces remain visible, many illumination terms change slowly, and camera motion is limited from one frame to the next. Temporal reuse exploits this coherence by reprojecting results from previous frames instead of recomputing them completely. The method is applicable to antialiasing, denoising, ambient occlusion, indirect illumination, reflections, shadow filtering, and many post-processing effects.

Reprojection uses motion vectors and depth to map previous samples into the current frame. A history sample should be accepted only when geometric and material consistency tests indicate that it still represents the same surface. Depth discontinuities, disocclusion, animation, and rapidly changing lighting reduce reliability. Confidence values can therefore be stored with the history and used to determine the blend factor between reused and newly computed information.

Temporal accumulation effectively increases sample count without increasing the number of rays or shading evaluations per frame. The benefit is largest in stable areas but it introduces a new optimization objective: avoid spending excessive work where history is already reliable while directing new

samples toward unstable regions. This is closely related to adaptive sampling and machine-learning uncertainty estimation[20]. A common scheduler can use temporal confidence as one of the inputs to the quality controller.

Caching is broader than temporal accumulation. Static shadow terms, prefiltered environment data, material intermediates, transformed constants, and spatial query results can all be reused when their dependencies have not changed. The essential requirement is correct invalidation. A cache that returns obsolete information may produce subtle visual errors that are difficult to diagnose. Resource versioning and explicit dependency tracking therefore provide both correctness and performance.

Dynamic internal resolution is an effective method when fragment shading or ray tracing scales approximately with pixel count. If frame time exceeds the target, the internal render size can be reduced and the output reconstructed to the display resolution. The control algorithm should respond smoothly: abrupt changes in scale can cause visible fluctuations in sharpness and can destabilize temporal filters. Hysteresis, filtered frame-time estimates, and bounded rate of change are appropriate.

Perceptual control extends the idea by recognizing that not all regions of an image contribute equally to perceived quality. Areas with high contrast, readable text, thin geometry, or direct user attention should receive more resources. Regions with strong motion or low visual sensitivity can often tolerate reduced shading rate or fewer rays. In virtual-reality systems, eye tracking enables foveated rendering, in which the foveal area is rendered at high quality and peripheral regions at lower quality.

The main difficulty is to prevent the optimization from becoming visible. Spatial transitions in resolution or shading rate should be smooth, and temporal stability must be preserved as the gaze or camera moves. Quality decisions should also respect content semantics. A small interface element may occupy few pixels yet be visually important, while a large blurred background can tolerate aggressive simplification. A practical quality map can combine screen-space derivatives, motion, depth discontinuities, saliency, and application-specific importance.

20 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

Perceptual methods are especially effective when combined with reconstruction. Lower shading rates or sparse ray samples create missing information, while temporal and neural filters reconstruct the full-resolution image. This forms a general pattern: computational work is concentrated where it most influences perceived result, and reconstruction supplies the remaining detail. The scheduler must nevertheless maintain conservative limits to avoid systematic degradation in scenes that differ from those used to tune the quality policy.

In many modern renderers, performance is limited not by arithmetic throughput but by memory traffic. Large texture sets, high-resolution G-buffers, shadow maps, geometry attributes, and acceleration structures compete for bandwidth and cache capacity. Reducing transferred bytes can therefore be more valuable than reducing a small number of shader instructions. The first step is to measure bandwidth, cache hit rates, and the size of intermediate surfaces rather than assuming that a shader is compute-bound.

Texture compression reduces both memory capacity and bandwidth requirements, provided that formats are chosen according to content. Normal maps, color textures, high-dynamic-range data, and masks may require different compression strategies. Mipmapping is equally important because it prevents distant surfaces from sampling unnecessarily high-resolution data. Streaming should load only the mip levels required by the current projected size and maintain a residency margin to prevent visible late loads during camera motion.

Intermediate buffers should use the narrowest precision that preserves the required quality. A G-buffer that stores every attribute in a 32-bit floating-point format can consume several times more bandwidth than necessary. Normals can often be encoded compactly, material identifiers stored as integers, and some reconstructed quantities omitted entirely. Such changes should be evaluated globally because reducing memory traffic may increase arithmetic cost. On many GPUs, exchanging bandwidth for moderate arithmetic is advantageous.

Memory allocation strategy also matters. Frequent allocation and destruction of short-lived render targets can introduce synchronization and fragmentation. Transient resource pools, aliasing of non-overlapping buffers, and frame graphs can reuse the same physical memory for different logical resources. The frame graph additionally makes resource lifetimes explicit and helps the renderer

schedule barriers and asynchronous work. Memory management becomes an active component of performance architecture.

A renderer with many materials, quality modes, light types, and geometric options can generate a combinatorial number of shader variants. A fully generic shader reduces the number of binaries but may execute unnecessary branches and access unused resources. Excessive specialization, on the other hand, increases compilation time, cache size, and pipeline-state management overhead. High-performance systems therefore seek a controlled middle ground.

Specialization should be applied to parameters that remove substantial work, such as the presence of normal mapping, skinning, alpha testing, or a major illumination feature. Minor numerical options can often remain uniform parameters. Variant keys should be compact and stable so that shader and pipeline caches can be indexed efficiently. Frequently used combinations may be precompiled, while rare variants are compiled asynchronously and temporarily fall back to a compatible simpler path.

Pipeline-state creation can cause visible stalls if performed on demand during rendering. A preparation phase should identify likely states from scene materials and compile them before they are needed. Persisting a pipeline cache between runs can reduce startup and traversal stalls. Telemetry is useful because it identifies which variants are actually used and which merely increase complexity. Eliminating unused combinations improves both memory consumption and development reliability.

Shader optimization remains subject to the same bottleneck principle as the rest of the renderer. Instruction count alone is not sufficient. Register pressure can reduce occupancy, divergent branches can waste lanes, and texture access may dominate execution. A shorter shader may even run slower if it increases memory stalls. Profilers and hardware counters should therefore guide specialization decisions rather than source-code appearance.

A frame graph describes rendering as a dependency graph of passes and resources. Instead of manually sequencing every stage, the system records which resources each pass reads or writes and derives a valid execution order. This makes lifetimes explicit, reveals opportunities for transient-memory reuse, and simplifies synchronization. More importantly, it makes it possible to identify independent work that can overlap on modern hardware.

Graphics, compute, and copy queues can execute concurrently when resource dependencies allow. For example, a compute queue may build light clusters for a later pass while the graphics queue rasterizes an unrelated shadow map, and a copy engine can upload future resources in parallel. Asynchronous overlap is beneficial only if it uses otherwise idle units. Launching compute work that competes for the same bottlenecked memory bandwidth can increase total frame time even when execution appears parallel.

The scheduler should therefore estimate not only dependencies but also resource contention. Passes that are both bandwidth-heavy may be better serialized, whereas a compute-heavy pass can overlap with a transfer-heavy stage. Hardware counters and historical timings can provide simple heuristics. The closed-loop architecture described earlier can extend from quality adaptation to queue scheduling by choosing overlap policies that minimize critical-path duration.

Frame graphs also improve correctness. Automatic barrier generation reduces the risk of reading data before writes are complete, and aliasing can be restricted to resources with non-overlapping lifetimes. For a large renderer with dozens of passes, this structured representation is easier to maintain than a growing sequence of manual transitions. Architectural clarity thus supports performance because it makes more aggressive scheduling possible without sacrificing reliability.

The techniques considered above can be integrated through a scheduler that treats frame time as a controlled resource. Its inputs include measured times of pipeline stages, GPU occupancy, memory traffic, visible-object counts, ray counts, temporal-confidence maps, streaming pressure, and worker utilization. Its outputs are quality parameters such as LOD thresholds, internal resolution, ray budget, shadow quality, BRDF variant, streaming priority, and task distribution. The scheduler should change only a limited set of parameters per frame to preserve stability.

A practical policy can be hierarchical. The first level preserves hard constraints such as memory limits and required latency. The second level ensures that the dominant bottleneck remains within the frame budget. The third level distributes remaining capacity to visible-quality improvements. This hierarchy prevents the controller from increasing geometric detail while the application is already bandwidth-limited or from adding rays when the denoiser is not the limiting stage. Optimization becomes a resource-allocation problem rather than a collection of independent switches.

Control should include prediction because measurements describe the previous frame while the next frame may be more complex. Camera velocity, scene regions, animation state, and historical complexity can provide short-term forecasts. Conservative prediction is valuable during fast camera movement because newly revealed geometry and textures can create sudden spikes. Prefetching, temporary LOD reduction, and a reserved time margin reduce these spikes without permanently lowering image quality.

The scheduler can also learn long-term characteristics of the application. It can estimate the cost of each quality option on the current hardware and build a device-specific performance model. Such calibration allows one software system to adapt to a wide range of GPUs and CPUs without manually tuned presets. Machine learning may assist this prediction[21], but a transparent rule-based controller remains useful because its decisions are easier to validate and constrain.

A single optimization recipe cannot be applied to all applications. In architectural visualization, geometry streaming, occlusion culling, high-resolution textures, and stable reflections are central. In games, latency, animation, material variety, and dynamic lighting dominate. In engineering visualization, accuracy and very large assemblies may be more important than cinematic effects. Offline production rendering prioritizes time to converged quality and efficient use of clusters. The acceleration strategy should therefore be assembled from methods that match the scene and interaction model.

For a large static interior, a strong combination is hierarchical visibility, compact clusters, GPU-driven submission, cached static lighting terms, and dynamic resolution for post-processing. For an outdoor open world, resource streaming, LOD, impostors, virtualized texture access, and predictive prefetching become more important. For reflective product visualization, high-quality BRDF evaluation[22], selective ray tracing, temporal denoising, and stable high-resolution reconstruction should be prioritized.

21 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

22 Romanyuk O., Zavalniuk Y., Romanyuk O., Kotlyk S., Bobko O., Melnyk O. New Illumination Model for Rendering. In: 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 17-19 September 2025. IEEE, 2025. Pp. 723-727. DOI: https://doi.org/10.1109/ACIT65614.2025.11185843; Romanyuk O., Lytvynenko V., Zavalniuk Y. Modified cosine-quadratic reflectance model. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska. 2025. Vol. 15, No. 1. Pp. 45-48. DOI: https://doi.org/10.35784/iapgos.7384

For scientific or medical visualization, perceptual shortcuts require additional caution because image structures may carry quantitative meaning. Approximation is acceptable only when it does not alter diagnostically or scientifically relevant features. In such systems, deterministic rendering paths, error bounds, and reproducible quality settings are more valuable than aggressive adaptive changes. Performance optimization can still use spatial indexing, data locality, parallelism, and careful memory management without modifying the information content of the visualization.

The scenario-oriented approach does not contradict a common architecture. On the contrary, a modular renderer with unified profiling and scheduling can expose different policy configurations while sharing the same underlying execution framework. This is one of the strongest arguments for separating scene representation, visibility, lighting, tracing, reconstruction, and scheduling into explicit subsystems[23],[24].

Further development of high-performance rendering is expected to increase the degree of adaptivity. Geometry, shading, rays, and reconstruction will increasingly be controlled by measured contribution to image quality rather than by fixed global settings. Hardware support for mesh processing, ray traversal, variable shading, matrix operations, and specialized memory access encourages architectures that dynamically select the most efficient computational path for each part of a frame.

The boundary between traditional graphics and machine learning will continue to become more flexible. Learned modules can estimate visibility, sample importance, reconstruction confidence, and performance cost, while deterministic modules preserve geometric and physical constraints. The most reliable systems are likely to use learning as a controlled accelerator inside a structured pipeline rather than as an opaque replacement for the entire rendering process[25].

23 Romanyuk O. N., Zavalniuk Y. K., Bobko O. L., Novoseltsev O. O., Stakhov O. Y. Basic features of computer graphics system design. In: Teoretyko-metodolohichni zasady suchasnykh naukovykh doslidzhen [Theoretical and Methodological Foundations of Modern Scientific Research]: monograph. Kharkiv: SG NTM "Novyi Kurs", 2026. Section 4.5. Pp. 255-269. 299 p. DOI: https://doi.org/10.61718/mon202601

24 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia: Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

25 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

Another important direction is joint optimization of rendering and resource transport. Very large models already require out-of-core data management, and cloud rendering introduces another memory hierarchy that spans client, network, server memory, and accelerator memory. Future schedulers must decide not only what to compute but also where the required data should reside and when it should be moved. Computation and data movement should be planned as one problem.

Finally, performance evaluation should evolve from isolated FPS measurements toward reproducible quality-performance curves. A renderer should be characterized by how much image quality it can maintain at 8, 16, or 33 ms budgets and how stable that quality remains during difficult scene transitions. Such a representation makes different methods comparable and directly supports adaptive scheduling. It also reflects the central principle of this monograph: performance is a controlled resource that must be allocated according to visual importance and system constraints.

Rendering algorithms should be adapted to the execution characteristics of the target hardware rather than optimized in an abstract machine model. The same workload can behave differently on a desktop GPU, an integrated graphics processor, a mobile accelerator, or a multi-GPU workstation. Differences in cache hierarchy, memory bandwidth, register capacity, wave or warp width, ray-traversal hardware, and asynchronous execution support change the relative cost of algorithms. A portable high-performance renderer should therefore separate the semantic description of a task from the hardware-specific strategy used to execute it.

Occupancy is an important but not absolute objective. Increasing the number of active warps can hide latency, but excessive register or shared-memory consumption may reduce the number of simultaneously resident thread groups. Conversely, a shader with lower occupancy can still be faster if it has better locality and fewer memory transactions. Practical tuning should therefore examine execution counters together: active-wave occupancy, cache behavior, instruction issue, memory stalls, and branch divergence. No single counter should be used as a universal perfor= mance score.

Hardware ray-tracing units change the cost balance between traversal, shading, and reconstruction. When intersection tests are accelerated, material

evaluation, memory traffic, and secondary-ray management can become the new bottlenecks. The renderer should respond by reducing shader divergence, improving material locality, and batching rays with compatible work. Similarly, matrix or tensor accelerators can make learned reconstruction less expensive, but the benefit depends on data-transfer overhead and the precision format supported by the device.

A useful architectural solution is a capability layer that describes hardware limits and exposes them to the adaptive scheduler. The same quality goal can then be achieved differently on different devices: one GPU may favor more rays and a lightweight denoiser, while another may favor fewer rays and a stronger reconstruction stage. Hardware-aware optimization therefore complements, rather than replaces, the bottleneck-oriented approach developed throughout this monograph.

Compact representations reduce both storage and transfer costs. Vertex positions, normals, texture coordinates, transforms, and material parameters do not always require 32-bit floating-point precision. Quantization to smaller integer or floating formats can reduce bandwidth and cache pressure, provided that the resulting error is bounded. The correct precision depends on the coordinate range and the visual role of the attribute. For example, local object-space positions can often be quantized more aggressively than world-space camera transforms.

Normal vectors are especially suitable for compact encoding because a unit vector contains only two independent degrees of freedom. Octahedral or related encodings store a normal in two components and reconstruct the third during shading. The arithmetic cost of reconstruction is typically small relative to the bandwidth saved when millions of normal values are fetched. Similar reasoning applies to tangent frames, material parameters, and packed visibility flags. The objective is to transfer information rather than redundant numerical precision.

Compression is also relevant to distributed and cloud rendering. Geometry blocks, textures, and intermediate results may traverse PCI Express, local networks, or Internet links. A format that is slightly more expensive to decode can still reduce total latency if it substantially lowers transfer size. The scheduler should therefore consider end-to-end cost: encode time, transfer time, decode time, and subsequent cache behavior. This is particularly important when

scene updates are frequent and the communication channel is the dominant constraint.

Quantization must be validated at image level. Numerical error that is harmless in a diffuse region may become visible in silhouettes, high-frequency normal maps, or reflective surfaces. A robust system can select precision per resource class and, in advanced cases, per LOD. Coarser LODs may use more aggressive quantization because their screen-space error is already bounded. Compact representation thus becomes another adaptive dimension of the rendering pipeline.

A practical optimization process should be iterative and evidence-based. The first stage is to define the target scenario: output resolution, desired frame time, quality requirements, hardware class, scene range, and acceptable latency. The second stage is to establish a reproducible benchmark with fixed camera paths and representative content. The third stage is profiling, in which CPU time, GPU time, memory traffic, synchronization, streaming, and network costs are separated. Optimization begins only after the dominant constraint is known.

The next stage is to choose the least invasive method capable of reducing the bottleneck. Geometry-limited frames suggest culling, LOD, clustering, or streaming changes; fragment-limited frames suggest overdraw reduction, simpler materials, or dynamic resolution; traversal-limited frames suggest acceleration-structure and ray-coherence improvements; memory-limited frames suggest compression, compact buffers, and reduced intermediate traffic. After each change, the complete benchmark is repeated because the bottleneck may move to another stage.

Regression control is essential. A performance improvement that introduces unstable frame times, visible artifacts, excessive memory use, or a large increase in code complexity may not be an engineering improvement. Each optimization should therefore be recorded with its performance effect, quality effect, memory effect, hardware dependence, and maintenance cost. This creates a technical history that prevents previously rejected approaches from being reintroduced without evidence and makes future tuning more systematic.

The final stage is adaptive integration. Once several optimizations are validated independently, they can be connected to the runtime scheduler and activated according to measured conditions. The system should use

conservative boundaries, hysteresis, and fallbacks so that adaptation does not oscillate or produce unexpected visual changes. The result is a renderer in which performance engineering is not a one-time optimization phase but a permanent feedback mechanism integrated into the software architecture.

Acceleration of three-dimensional scene rendering is a multilevel problem that includes scene representation, visibility, rasterization, ray tracing, illumination, vector operations, memory organization, parallel execution, workload distribution, and image reconstruction. The most important principle is early elimination of unnecessary work: invisible objects should not proceed through later stages, excessive geometry should be replaced by an appropriate LOD, and expensive calculations should be performed only where they influence the final image.

The second principle is bottleneck orientation. A profiling system should identify the dominant stage, after which the parameters of that stage are adjusted first. This prevents significant effort from being spent on components that do not limit performance. In multicore, GPU, and cluster systems the same principle extends to load balancing, memory traffic, command generation, and synchronization overhead.

The third principle is hybridization. Rasterization, ray tracing, adaptive detail, approximate reflectance models, cloud resources, temporal reuse, and machine learning are not mutually exclusive. They can operate as coordinated parts of one pipeline. Machine learning is particularly promising for reconstruction, denoising, uncertainty estimation, and performance prediction because it allows expensive samples to be concentrated in visually significant regions[26].

The expanded analysis shows that modern acceleration increasingly depends on data-oriented and adaptive architectures. GPU-driven submission, meshlets, asynchronous streaming, frame graphs, memory aliasing, temporal caches, variable internal resolution, shader specialization, and predictive scheduling can all reduce critical-path time when applied to the correct bottleneck. Their effectiveness depends on clear module boundaries, reliable telemetry, and conservative synchronization.

26 Romaniuk O., Zavalniuk Ye., Bobko O., Stakhov O., Romaniuk Ok. Machine learning for accelerated rendering. In: Science and education: innovations and prospects: collective monograph / compiled by V. Shpak; chairman of the Editorial Board S. Tabachnikov. Sherman Oaks, California: GS Publishing Services, 2026. Pp. 7-17. 191 p. DOI: https://doi.org/10.51587/9798-9917-51995-2026-027-6-16

A promising direction is the construction of self-adjusting rendering systems that use per-frame telemetry to select LOD, internal resolution, ray count, illumination complexity, streaming priority, and workload distribution. Combined with modular architectures[27] and efficient reflectance models[28],[29], this makes it possible to treat performance not as a fixed property of an implementation but as a manageable resource dynamically distributed to achieve the required quality of three-dimensional visualization.

DOI: 10.51587/9798-9935-42874-2026-029-14-43

27 Zavalniuk Y. K., Romanyuk O. N., Pavlov V. S., Bobko O. L., Novoseltsev O. O. New software system architecture for highly realistic and high-performance rendering of 3D images. Optiko-elektronni informatsiino-enerhetychni tekhnolohii [Optical-Electronic Information-Energy Technologies]. Vinnytsia: Vinnytsia National Technical University, 2026. No. 1 (51). Pp. 97-107. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-97-107 (In Ukrainian).

28 Romanyuk O., Zavalniuk Y., Romanyuk O., Kotlyk S., Bobko O., Melnyk O. New Illumination Model for Rendering. In: 2025 15th International Conference on Advanced Computer Information Technologies (ACIT), Sibenik, Croatia, 17-19 September 2025. IEEE, 2025. Pp. 723-727. DOI: https://doi.org/10.1109/ACIT65614.2025.11185843

29 Romanyuk O., Lytvynenko V., Zavalniuk Y. Modified cosine-quadratic reflectance model. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska. 2025. Vol. 15, No. 1. Pp. 45-48. DOI: https://doi.org/10.35784/iapgos.7384