

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER
EDUCATION
NORTH CAUCASUS FEDERAL UNIVERSITY

APPROVED BY:

Acting Dean of the Faculty of Mathematics
and Computer Sciences named after Professor N.I. Chervyakov
T.V. Grobova

WORK PROGRAM ON THE DISCIPLINE (MODULE)
DIGITAL IMAGE PROCESSING AND MACHINE VISION / ЦИФРОВАЯ
ОБРАБОТКА ИЗОБРАЖЕНИЙ И МАШИННОЕ ЗРЕНИЕ

Training program	02.04.01 Mathematics and Computer Sciences
Directivity (profile)	Training Program for Electronic Information
Year of admission	2026
Mode of study	intramural
Implemented in the semester	2

Developed by

PhD, Head of the Department of
Mathematical Modelling
Lyakhov P.A.

1. The purpose and objectives of mastering the discipline (module)

To provide Master's students with a rigorous mathematical and algorithmic foundation in classical digital image processing and computer vision, enabling the design, analysis, and implementation of robust visual information processing pipelines. The discipline establishes the necessary theoretical and practical groundwork for subsequent study of deep learning and modern AI architectures in electronic information engineering.

Objectives of the discipline:

1. Master spatial and frequency domain representations of images, including Fourier analysis, wavelet transforms, and discrete cosine transforms.
2. Acquire skills in image enhancement, restoration, and segmentation using classical methods (watershed, active contours, graph-based algorithms).
3. Study the geometric foundations of machine vision: epipolar geometry, camera calibration, and structure-from-motion (SfM).
4. Develop the ability to critically evaluate algorithmic assumptions, parameter sensitivity, and computational complexity in real-world noisy environments.
5. Implement core image processing and vision algorithms from scratch using Python, NumPy, scikit-image, and OpenCV, ensuring deep understanding beyond high-level library abstractions.
6. Learn to integrate classical geometric and variational methods with modern neural network approaches in hybrid processing architectures.

2. The place of the discipline (module) in the structure of the educational program

The discipline "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" (Index: Б1.О.09) belongs to Block 1, Mandatory Part of the Master's educational program 02.04.01 Mathematics and Computer Sciences. It is implemented in Semester 2 and carries 4 credit units. The course builds upon foundational knowledge acquired in Digital Signal Processing (Semester 1) and Machine Learning Programming Design Based on Python (Semester 1). It serves as a critical prerequisite for advanced modules in Semester 3–4, including Deep Learning, Reinforcement Learning, and Artificial Intelligence Models and Theories, by providing the classical algorithmic and mathematical framework required before transitioning to data-driven neural approaches.

3. List of planned learning outcomes in the discipline (module) correlated with the planned outcomes of the educational program

Competency code, description	Indicator code, description	Planned learning outcomes in the discipline (module) characterizing the stages of competencies and indicators formation
UC-1 Able to conduct critical analysis of information, apply a systematic approach to solve research and professional tasks	UC-1.1 Identifies and formulates professional problems, selects appropriate mathematical and algorithmic tools for their solution	Applies spatial and frequency domain representations, variational principles, and geometric vision models to design robust image processing pipelines for real-world visual data analysis.
OPC-1 Able to apply fundamental mathematical knowledge and modern programming technologies to solve professional problems in the field of	OPC-1.1 Develops and implements efficient algorithms for processing and analyzing visual data	Implements core image processing and computer vision algorithms from scratch using Python, NumPy, and OpenCV, demonstrating understanding of numerical stability, computational complexity, and algorithmic

mathematics and computer science		assumptions.
OPC-2 Able to conduct independent research using modern methods, tools, and information technologies in the chosen professional field	OPC-2.1 Designs and validates experimental pipelines for visual information processing, analyzing results against theoretical models	Designs complete processing workflows (filtering → segmentation → feature extraction → 3D reconstruction), critically evaluates parameter sensitivity and noise robustness, and justifies the integration of classical methods with hybrid AI architectures.
PC-1 Able to develop and apply mathematical models and software systems for processing electronic information and intelligent data analysis	PC-1.2 Applies geometric and algebraic methods for camera calibration, stereo vision, and structure-from-motion	Solves multi-view geometry problems, performs intrinsic/extrinsic camera calibration, and reconstructs 3D scenes from 2D image sequences using projective geometry, epipolar constraints, and optimization on manifolds.

4. The scope of the academic discipline (module) and control forms*

Scope of classes:	c.u.	Acad. hours
Total:	4	144
Lectures		16
Laboratory works		16
Practical lessons		-
Independent work		58
Control forms:		
Exam (Preparation & Control)		54

5. Content of the discipline (module), structured by topics (sections) with an indication of the number of hours and types of training sessions

No.	Section (topic) of the discipline	Implemented competencies, indicators	Contact work of students with a teacher, hours			Independent work, hours
			Lectures	Practical lessons	Laboratory works	
1 semester						
1	Image Fundamentals and Spatial Operations	UC-1.1, OPC-1.1	2.00	-	2.00	7.00
2	Frequency Domain Analysis: Fourier Transform & Filtering	UC-1.1, OPC-1.1, OPC-2.1	2.00	-	2.00	8.00
3	Wavelet Transforms and Multiscale Analysis	OPC-1.1, OPC-2.1	2.00	-	2.00	8.00
4	Morphological Processing and Image Segmentation	OPC-2.1, PC-1.2	2.00	-	2.00	8.00
5	Edge and Keypoint Detection	OPC-1.1, OPC-2.1	2.00	-	2.00	7.00
6	Descriptor Construction and Image Matching	OPC-2.1, PC-1.2	2.00	-	2.00	7.00
7	Single and Two-View Geometry: Calibration & Epipolar Geometry	PC-1.2, UC-1.1	2.00	-	2.00	7.00
8	End-to-End Processing Pipeline (Project Workshop)	OPC-2.1, PC-1.2	2.00	-	2.00	6.00
	TOTAL for the 2d semester		16.00	-	16.00	58.00
	TOTAL		16.00	-	16.00	58.00

6. Assessment Tools Fund for the continuous monitoring of progress and intermediate certification of students in the discipline (module)

The Assessment Tools Fund (ATF) for "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" is designed to objectively measure the formation of competencies **UC-1.1, OPC-1.1, OPC-2.1, and PC-1.2** through a multi-component evaluation system aligned with the 144-hour workload.

6.1. Structure of assessment components

Assessment type	Weight	Timing	Competencies assessed	Format
Laboratory reports (4 tasks)	30%	Weeks 2, 4, 6, 8	OPC-1.1, OPC-2.1	Code + written analysis (Python/NumPy/OpenCV), reproducibility checklist
Mid-term colloquium (2 sessions)	20%	Weeks 3, 7	UC-1.1, PC-1.2	Oral defense of mathematical derivations + short coding quiz
End-to-end project	30%	Week 8 submission	OPC-2.1, PC-1.2	3D reconstruction pipeline or variational restoration; report + demo + code review
Final exam	20%	Semester end	All	Written theory (40%) + practical algorithm design task (60%)

6.2. Competency-specific assessment criteria

For OPC-1.1 (Algorithm implementation):

Excellent (90-100%): Implements algorithm from scratch without high-level wrappers; demonstrates numerical stability analysis; provides complexity estimation; includes edge-case handling.

Satisfactory (70-89%): Core functionality works; minor reliance on library shortcuts; basic documentation present.

Unsatisfactory (<70%): Algorithm fails on test data; no understanding of underlying mathematics; code not reproducible.

For PC-1.2 (Geometric vision tasks):

Excellent: Correctly formulates epipolar constraints; performs bundle adjustment with convergence analysis; validates reconstruction error against ground truth.

Satisfactory: Pipeline produces plausible 3D output; calibration parameters within reasonable bounds; limited error analysis.

Unsatisfactory: Fundamental errors in projection models; no validation; results not interpretable.

For UC-1.1 (Critical analysis):

Assessed via comparative reports: student must justify method selection, analyze failure modes, and propose hybrid improvements (classical + deep learning).

6.3. Examination structure (54 hours preparation allocated)

Part A (Theory, 40 min): 4 questions covering frequency analysis, variational methods, epipolar geometry, optimization on manifolds.

Part B (Practical design, 80 min): Given a noisy stereo pair, design a complete processing pipeline: justify each stage mathematically, sketch pseudocode, estimate computational cost, discuss parameter sensitivity.

Grading rubric: Mathematical correctness (40%), algorithmic coherence (30%), critical evaluation (20%), clarity of presentation (10%).

6.4. Continuous monitoring procedures

Automated testing of lab submissions via university CI/CD pipeline (unit tests for core functions).

Peer code review sessions (Weeks 4, 6) to develop collaborative debugging skills.

Mandatory revision cycle: students resubmit lab reports with corrections within 7 days to receive full credit.

7. Methodological guidelines for students on mastering the discipline

7.1. Course organization and expectations

This discipline follows an intensive 8-week format (Semester 2) with synchronous lectures (2 h/week) and laboratory sessions (2 h/week). The remaining 112 hours are allocated to independent work (58 h) and exam preparation (54 h). Success requires consistent weekly engagement; falling behind is difficult to recover due to cumulative mathematical content.

7.2. Preparation strategy

Before each lecture:

Review prerequisite mathematics (e.g., complex exponentials for Fourier week, PDE basics for variational methods).

Skim the corresponding chapter in Gonzalez & Woods or Forsyth & Ponce.

Prepare 1-2 questions on conceptual difficulties.

During laboratory sessions:

Implement algorithms *from scratch* first (e.g., manual convolution, SVD-based fundamental matrix estimation) before verifying with OpenCV/scikit-image.

Document all parameter choices and their impact on output quality.

Save intermediate results for comparative analysis in reports.

Independent work (58 hours) guidance:

Weeks 1–4: Focus on mathematical derivations; reproduce key formulas by hand; use Jupyter notebooks to visualize filter responses, wavelet decompositions, morphological operations.

Weeks 5–8: Shift to project development; allocate 15–20 hours to the end-to-end pipeline; conduct ablation studies (e.g., "What happens if I remove the RANSAC step?").

Reading assignments: Prioritize original papers (Lowe SIFT, ROF 1992) after understanding the textbook treatment; write 1-page critical summaries.

7.3. Exam preparation (54 hours)

Weeks 9–12: Systematic review using the competency map (Section 3); solve past exam problems; practice explaining algorithms verbally (for oral defense components).

Focus on *connections*: e.g., how wavelet thresholding relates to variational regularization; how epipolar geometry constraints reduce search space in stereo matching.

Form study groups for collaborative problem-solving, but ensure individual mastery of derivations and code.

7.4. Academic integrity and collaboration policy

Code sharing is permitted for debugging assistance only; all submitted work must be individually authored.

Use of generative AI tools is allowed for code documentation and explanation refinement, but *not* for core algorithm implementation or mathematical derivations.

All external sources (papers, tutorials, Stack Overflow snippets) must be cited in lab reports and project documentation.

8. Teaching and learning materials on the discipline

8.1. List of essential and supplementary literature

8.1.1. Essential literature (core textbooks):

1. Gonzalez, R. C., & Woods, R. E. *Digital Image Processing*. 4th Edition. Pearson, 2018. (*Covers spatial/frequency domains, morphology, segmentation*)
2. Forsyth, D., & Ponce, J. *Computer Vision: A Modern Approach*. 3rd Edition. Pearson, 2019. (*Bridges classical methods to learning-based approaches*)
3. Hartley, R., & Zisserman, A. *Multiple View Geometry in Computer Vision*. 2nd Edition. Cambridge University Press, 2004. (*Definitive reference for calibration, epipolar geometry, SfM*)

8.1.2. Supplementary literature (advanced/reference):

1. Szeliski, R. *Computer Vision: Algorithms and Applications*. 2nd Edition. Springer, 2022. (*Open-access; excellent for project inspiration and comparative analysis*)
2. Mallat, S. *A Wavelet Tour of Signal Processing*. 3rd Edition. Academic Press, 2008. (*Rigorous treatment of multiscale analysis*)
3. Sonka, M., Hlavac, V., & Boyle, R. *Image Processing, Analysis, and Machine Vision*. 4th Edition. Cengage, 2014. (*Practical algorithms with implementation notes*)
4. Davies, E. R. *Computer and Machine Vision: Theory, Algorithms, Practicalities*. 4th Edition. Academic Press, 2012. (*Emphasis on real-world constraints and optimization*)

8.2. Materials for independent work and project development

Code templates: GitHub repository with Jupyter notebooks containing algorithm skeletons
(e.g., `convolution_from_scratch.ipynb`, `fundamental_matrix_ransac.py`).

Datasets: Curated subsets of Middlebury (stereo), KITTI (visual odometry), and COCO (keypoint matching) for controlled experiments.

Validation tools: Scripts for PSNR/SSIM computation, epipolar error measurement, and runtime profiling.

Methodological guides:

"How to derive a filter from a variational principle" (PDF)

"Checklist for reproducible computer vision experiments" (Markdown)

"Parameter tuning strategy for noisy real-world data" (interactive notebook)

8.3. Internet resources and software environment

Essential online resources:

1. OpenCV Documentation: <https://docs.opencv.org> (*Reference for verification, not primary implementation*)
2. scikit-image Gallery: https://scikit-image.org/docs/stable/auto_examples (*Algorithm comparisons and parameter guides*)
3. arXiv cs.CV section: <https://arxiv.org/list/cs.CV/recent> (*For staying current with hybrid classical/deep methods*)
4. PyImageSearch tutorials: <https://pyimagesearch.com> (*Practical implementation insights*)

Software requirements:

Python 3.9+ with NumPy, SciPy, scikit-image, OpenCV-Python, Matplotlib

JupyterLab environment (provided via university cloud workspace)

Optional: CUDA-enabled PyTorch for project extensions (hybrid architectures)

University infrastructure:

Access to NCFU high-performance computing cluster for computationally intensive project tasks (e.g., bundle adjustment on large image sets).

LMS-integrated submission portal with automated testing for laboratory assignments.

Virtual consultation hours via university-approved videoconferencing platform.

9. List of information technologies used in the course of educational process in the discipline (module), including a list of software and information reference systems

The educational process for the discipline "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" utilizes a modern software stack and digital infrastructure to support both theoretical instruction and practical algorithm development. The primary programming language is Python version 3.9 or higher, selected for its extensive ecosystem of scientific computing libraries and its suitability for rapid prototyping of image processing algorithms. Core numerical operations are performed using NumPy and SciPy, which provide efficient array handling, linear algebra routines, and optimization primitives essential for implementing filters, transforms, and geometric computations from first principles.

Image processing functionality is supported by two complementary libraries: scikit-image, which offers pedagogically transparent implementations of classical algorithms suitable for educational purposes, and OpenCV-Python, which provides high-performance reference implementations for verification and comparison. Visualization of results, including filter responses, frequency spectra, feature matches, and three-dimensional reconstructions, is accomplished using Matplotlib and Plotly. All laboratory assignments and project work are developed in JupyterLab, enabling reproducible, interactive workflows that combine code, mathematical notation, and visual output in a single document. Version control and collaborative development are managed through Git with university-hosted GitHub or GitLab instances, ensuring proper tracking of iterative improvements and facilitating code review.

For students extending their projects to hybrid architectures that combine classical geometric methods with neural feature extractors, optional access to PyTorch with CUDA support is provided. The development environment is standardized through conda environment specification files, guaranteeing consistent package versions across all student workstations and cloud resources.

Information reference systems essential to the discipline include IEEE Xplore and SpringerLink for access to peer-reviewed literature on classical and modern computer vision methods, and the arXiv cs.CV section for preprints on emerging hybrid approaches. Official documentation for OpenCV, scikit-image, NumPy, and SciPy serves as the primary technical reference for function signatures, parameter details, and performance characteristics. Curated datasets for controlled experimentation include subsets of the Middlebury Stereo Dataset for epipolar geometry validation, the KITTI Vision Benchmark for structure-from-motion tasks, and the COCO dataset for keypoint matching evaluation; these are distributed via the university learning management system with integrity checksums.

University digital infrastructure supports the educational process through the NCFU Electronic Library, providing access to subscribed textbooks such as Gonzalez and Woods, Forsyth and Ponce, and Hartley and Zisserman. The Moodle-based LMS hosts submission portals for laboratory reports, automated testing feedback, and course announcements. Computationally intensive project tasks, such as bundle adjustment or dense stereo reconstruction, may be executed on the university high-performance computing cluster managed via Slurm, with allocated GPU time for enrolled students. A cloud-based JupyterHub instance with pre-installed course dependencies serves as a backup development environment, accessible via university single sign-on to ensure continuity for students with limited local hardware resources.

Academic integrity in the use of digital tools is maintained through clear policies: generative AI assistants may be used for code documentation and explanation refinement but not for core algorithm implementation or mathematical derivations; all external code snippets or tutorial references must be explicitly cited in submitted work; and all submissions must include dependency specifications and execution instructions to ensure reproducibility. Automated continuous integration pipelines validate laboratory submissions against reference outputs, providing immediate feedback on correctness and computational efficiency while reinforcing best practices in reproducible research.

10. Description of the material and technical support necessary for the educational process in the discipline (module)

The educational process for the discipline "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" requires specialized material and technical infrastructure to support both theoretical instruction and hands-on algorithmic development within the approved 144-hour workload. Lecture-type sessions are conducted in a classroom equipped with projection capabilities (minimum 3000 lumens, 1920×1080 resolution), a motorized screen, and audio reinforcement to ensure clear delivery of mathematical derivations and algorithm visualizations. The instructor workstation includes a desktop computer with a dedicated GPU (NVIDIA RTX 3060 or higher), 32 GB RAM, and dual-monitor configuration to facilitate live coding demonstrations and real-time visualization of filter responses, epipolar geometry overlays, and 3D reconstruction previews.

Laboratory sessions require student workstations meeting minimum specifications of a 4-core CPU, 8 GB RAM, and 256 GB SSD storage, with recommended configurations including 16 GB RAM and a dedicated GPU (NVIDIA GTX 1660 or higher) for computationally intensive tasks such as wavelet decomposition and bundle adjustment. All machines run either Ubuntu 20.04 LTS or Windows 10/11 with WSL2 to ensure compatibility with the Python scientific stack. The mandatory software environment includes Python 3.9 or later, NumPy and SciPy for numerical computing, scikit-image and OpenCV-Python for image processing reference implementations, Matplotlib and Plotly for visualization, and JupyterLab for interactive development and assignment submission. Version control is managed through Git with university-hosted GitHub or GitLab instances.

Data resources for the course consist of curated subsets of the Middlebury stereo dataset, KITTI visual odometry sequences, and COCO keypoint annotations, pre-filtered for course tasks and totaling approximately 5 GB. These datasets are distributed via the university learning management system with checksums to ensure integrity. Validation scripts for PSNR/SSIM computation, epipolar error measurement, and runtime profiling are provided through a dedicated GitHub repository. Students requiring additional computational power for project work may request allocated GPU time on the university high-performance computing cluster, which is managed via Slurm and provides access to NVIDIA A100 or V100 accelerators with 100 GB of shared storage per student.

Technical support is provided through pre-semester deployment of standardized conda environment images to all laboratory machines, accompanied by installation guides for personal devices. During the semester, a teaching assistant is available during laboratory hours for environment troubleshooting, and a ticketing system handles software-related issues. All submitted code must include dependency specifications and execution instructions to ensure reproducibility, and an automated continuous integration pipeline validates submissions against reference outputs.

All software used in the discipline is licensed under open-source agreements (MIT, BSD, Apache 2.0) or university-wide academic licenses. Student code and project data are stored on university-managed infrastructure compliant with Russian Federation data protection regulations (152-FZ). GPU resource allocation follows fair-share scheduling to ensure equitable access for all enrolled students, and usage is monitored to prevent resource exhaustion. The cloud-based

JupyterHub instance serving as a backup development environment is configured to meet accessibility standards, and documentation for all course software includes accessibility notes to support inclusive participation.

11. Features of mastering the discipline (module) by persons with disabilities

The educational process for the discipline "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" is organized in accordance with the principles of inclusive higher education and the requirements of accessible learning environments. Students with disabilities are provided with individualized accommodations that preserve the academic rigor of the course while ensuring equitable access to all learning activities, including lectures, laboratory sessions, independent work, and assessment procedures.

For students with visual impairments, the course materials are adapted to support non-visual or low-vision interaction with visual content. All algorithmic explanations and mathematical derivations are provided in text-based formats compatible with screen readers, including properly structured LaTeX source files and accessible Jupyter notebooks. Image processing results and visualizations are accompanied by detailed textual descriptions of key features, such as edge locations, segmentation boundaries, or reconstruction quality metrics. Where visual inspection is pedagogically essential, alternative tactile or sonified representations of image data may be employed, and students are permitted to work with high-contrast display settings, screen magnification tools, or assistive software. Laboratory tasks can be completed using keyboard-only navigation in development environments, and code templates are provided with comprehensive inline documentation to reduce reliance on visual debugging.

For students with motor or mobility impairments affecting upper-limb function, the course accommodates alternative input methods for programming and laboratory work. Code submission is accepted via voice-to-text systems, adaptive keyboards, or through an assistant who transcribes dictated code under the student's direct supervision. Time allowances for laboratory assignments and examinations are extended proportionally to the documented needs of the student. The use of integrated development environments with customizable shortcuts, code completion, and macro support is encouraged to minimize repetitive keystrokes.

For students with hearing impairments, all oral instruction during lectures and laboratory consultations is supplemented with real-time captioning or sign-language interpretation upon request. Video materials and recorded demonstrations include accurate subtitles and, where applicable, transcripts. Written instructions for laboratory tasks are provided in advance, and assessment questions are presented in written form to ensure clarity. Group project work is structured to allow written or text-based communication channels as a primary mode of collaboration.

Assessment procedures are adapted to maintain validity while accommodating individual needs. Examination tasks may be presented in extended time windows, in alternative formats (e.g., written response instead of oral defense), or with the use of assistive technology. The competency-based evaluation criteria remain unchanged; only the mode of demonstration is adjusted. All adaptations are documented in an individual learning plan developed in coordination with the university's accessibility support services and are reviewed periodically to ensure their effectiveness.

The technical infrastructure supporting the discipline—including the university's JupyterHub instance, version control systems, and high-performance computing resources—is configured to comply with accessibility standards. Documentation for all software tools used in the course includes accessibility notes, and teaching staff receive guidance on inclusive pedagogical practices for technical subjects. Students are encouraged to communicate their specific requirements at the start of the semester to enable timely implementation of appropriate accommodations.

12. Features of the implementation of the discipline with the use of distance learning technologies and e-learning

The implementation of the discipline "Digital Image Processing and Machine Vision / Цифровая обработка изображений и машинное зрение" with the use of distance learning technologies and e-learning is organized in accordance with Federal Law No. 273-FZ "On Education in the Russian Federation" and the regulatory framework of North Caucasus Federal University. The course is designed to maintain full academic rigor and practical competency development regardless of the delivery mode, ensuring that remote students achieve the same learning outcomes as those attending on-campus sessions.

The discipline leverages a hybrid synchronous-asynchronous model. Synchronous components include live lecture broadcasts and laboratory consultations conducted via university-approved videoconferencing platforms with screen-sharing capabilities for real-time code demonstration and mathematical derivation. Asynchronous components comprise recorded lecture segments, interactive Jupyter notebooks with embedded exercises, and self-paced laboratory assignments with automated testing feedback. All instructional materials are hosted on the NCFU electronic information and educational environment, providing centralized access to course content, submission portals, and communication channels.

Laboratory sessions, which form the core practical component of the discipline, are adapted for remote execution through cloud-based computational resources. Students access a pre-configured JupyterHub environment containing all required software packages (Python, NumPy, scikit-image, OpenCV) and curated datasets (Middlebury, KITTI subsets). This eliminates local installation barriers and ensures reproducibility across diverse student hardware configurations. Laboratory assignments are submitted via version-controlled repositories, enabling automated validation of code correctness, computational efficiency, and output quality against reference implementations.

Assessment procedures in the distance format preserve competency-based evaluation criteria while adapting delivery mechanisms. Written theoretical examinations are administered through secure online testing systems with randomized question pools and time constraints. Practical examinations require students to design and implement image processing pipelines within a controlled time window, with code execution and results verified through containerized evaluation environments. Oral defenses of project work are conducted via videoconference with screen-sharing for live demonstration of algorithm performance.

Technical requirements for remote participation include a stable internet connection capable of supporting video streaming and large dataset transfers, a workstation meeting minimum specifications for image processing tasks (8 GB RAM, 4-core CPU, SSD storage), and access to a webcam and microphone for interactive sessions. Students with limited local computational resources may request allocated GPU time on the university high-performance computing cluster for project tasks requiring accelerated computation.

Academic integrity in the distance format is maintained through a combination of technical and procedural measures. Code submissions are checked for originality using plagiarism detection tools adapted for programming assignments. Examination sessions may employ remote proctoring solutions where institutionally approved. All external resources used in assignments must be explicitly cited, and the use of generative AI tools is permitted only for documentation and explanation refinement, not for core algorithm implementation or mathematical derivations.

The course is taught using e-learning and distance learning technologies, using the North Caucasus Federal University's electronic information and educational environment, accessible to students via the internet, or through the resources of other organizations, including platforms providing video conferencing, online meetings, and distance learning services (MTS Link), as well as social media for student-teacher communication.

The teaching and methodological support for the course, taught using e-learning and distance learning technologies, includes a work program, teaching aids or a lecture course,

methodological guidelines for various types of student learning activities required by the course, and other teaching and methodological materials available in the North Caucasus Federal University's electronic information and educational environment.