Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Fine-tuning Multimodal Transformer Models for Generating Actions in Virtual and Real Environments

ALEKSEI STAROVEROV[1,2], ANDREY S. GORODETSKY[2,3], ANDREI S. KRISHTOPIK[2], ULIANA A. IZMESTEVA[2], DMITRY A. YUDIN[1,2], ALEXEY K. KOVALEV[1,3], and ALEKSANDR I. PANOV[1,2]
[1]AIRI, Moscow, 121170, Russia
[2]Center of Cognitive Modeling, Moscow Institute of Physics and Technology, 9, Institutskij per., Dolgoprudny, 141701, Russia
[3]Federal Research Center "Computer Science and Control", 9, 60-letiya Oktyabrya pr., Moscow, 117312, Russia

Corresponding author: Aleksei Staroverov (e-mail: staroverov@airi.net).

This work was partially supported by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002; grant No. 70-2021-00138).

ABSTRACT In this work, we propose and investigate an original approach to using a pre-trained multimodal transformer of a specialized architecture for controlling a robotic agent in an object manipulation task based on language instruction, which we refer to as RozumFormer. Our model is based on a bimodal (text-image) transformer architecture originally trained for solving tasks that use one or both modalities, such as language modeling, visual question answering, image captioning, text recognition, text-to-image generation, etc. The discussed model was adapted for robotic manipulation tasks by organizing the input sequence of tokens in a particular way, consisting of tokens for text, images, and actions. We demonstrated that such a model adapts well to new tasks and shows better results with fine-tuning than complete training in simulation and real environments. To transfer the model from the simulator to a real robot, new datasets were collected and annotated. In addition, experiments controlling the agent in a visual environment using reinforcement learning have shown that fine-tuning the model with a mixed dataset that includes examples from the initial visual-linguistic tasks only slightly decreases performance on these tasks, simplifying the addition of tasks from another domain.

INDEX TERMS Action generation, bimodal transformer models, intelligent agent, robotic manipulator arm control.

I. INTRODUCTION

Transformers [1] are finding applications in increasingly diverse fields. Initially applied to natural language processing tasks [1], [2], this class of models quickly became a universal architecture used in computer vision [3], multimodal task solving [4], [5], reinforcement learning [6], and other domains [7]. Transformers have formed the basis of foundation models [8] and generalist agents [9].

In this work, we propose and explore the use of a pre-trained multimodal transformer of a specialized architecture to control an intelligent agent, which we call RozumFormer. In contrast to work that focuses on the use of large language models (LLMs) [10]–[12], we base our approach on a large bimodal (text-image) model, initially designed for solving tasks that require text and image modalities, such as question answering, text generation, visual question answering, image captioning, text-to-image generation, and others. However, the adaptation of this model for the control of an intelligent agent is complicated by the specific organization of the input token sequence, which needs to be maintained when addressing new tasks. Despite this, we have shown that such a model adapts well to new tasks. We illustrate this with the task of controlling a robotic manipulator arm from the VIMA-Bench benchmark [13] in both virtual and real environments. In addition, we conduct experiments on the Atari game environments [14] and show that fine-tuning the model with a mixed batch containing examples from the original and target tasks only slightly decreases performance on the original tasks, making it easier to add tasks from another domain.

An important part of our research is transferring the obtained model from the simulator to a real robot. It is important to note that in the simulator, a UR5[1] robotic manip-

[1]https://www.universal-robots.com/products/ur5-robot/

ulator arm was used, while in the experiments on the real robot, a ROZUM PULSE 75[2] robotic arm was used. Despite the kinematic similarities between these robotic manipulator arms, they have different dimensional characteristics and workspace sizes, indicating the multi-embodiment nature of the model. To facilitate the model transfer, the cameras on the robot stand were positioned to match their locations in the simulator. Due to a reality gap (lack of photorealism) between the simulator data and the actual robot data, additional datasets were collected and annotated to fine-tune the model.

The main contributions of the paper are:

1) We proposed the approach of adapting a pre-trained large multimodal model called RozumFormer to a new application domain – robotics tasks in virtual and real environments.
2) In addition to the classical Sim-to-real transfer, we proposed an original approach to transfer a model from a simulator environment to a real robotic stand, whose main feature is the ability to transfer the representation of images and actions of a real robotic system into a format suitable for working with models pre-trained on simulator data.
3) We collected and annotated datasets called Sweep-Seg and Sweep-Plan, which are used to train and evaluate the control model of a real robotic manipulator arm.

The model weights, source code, and collected datasets are available at the following link: https://github.com/cog-isa/rozumarm-vima.

II. RELATED WORKS

Recently, transformers have been actively applied in robotics [7], [10]–[12], [15]–[17], both for tasks related to Embodied Artificial Intelligence [7], [10], [11], [13] and for more specific tasks involving the control of robotic manipulator arms [12], [13], [15]–[17]. Typically, such models are highly specialized [7], [15], [17] and not designed to perform other tasks. Transformers have also gained wide popularity in reinforcement learning, both for controlling intelligent agents [6], [18]–[23] and for modeling the environment in which the agent operates [24].

The research results of large language models on tasks for which they were not originally trained [25]–[27], on the one hand, and recent work on adding a visual modality to LLMs for solving robotics tasks (e.g. PALM-E [10], RT-2 [12]), on the other hand, demonstrate the potential for leveraging and transferring knowledge from one domain to another, thereby expanding the list of tasks that the model can address.

There are many methods for solving the problem of transferring the skills of models trained in simulators to control real robots (the so-called sim-to-real problem) [28], [29]. This topic is still actively developing, since there is a huge variety of simulators, types of robots and their applications, teaching methods, and knowledge transfer. In our work, we focus on robotic manipulator arms. Research into the transfer of reinforcement learning methods to agents solving object grasping problems is very popular in this context [30]. There are approaches to real-sim-real transfer [31], which, despite their simplicity, are seen as convenient tools for the practical deployment of agents trained in simulators. The formulation of the zero-shot sim-to-real transfer problem based on human demonstrations [32], [33] is actively being investigated. The work by Brohan et al. [12] demonstrates that the use of a vast number of demonstrations available on the web enables a robotic agent to learn high-level actions that can then be applied to the control of a real robot.

In general, a universal model or approach to sim-to-real transfer has not yet been created. Therefore, in this work, we study it in relation to a specific group of manipulation tasks that a considered real robot must solve.

III. FORMAL PROBLEM STATEMENT

In this paper, we consider the case where the intelligent agent policy is learned by a large transformer model $\mathcal{LTM}$, which is capable of producing a solution out_i for task $i \in T_A$ from the set of available tasks T_A, taking various multimodal inputs $in_1, \dots in_j$, where j is the number of multimodal inputs:

$$out_i = \mathcal{LTM}(in_1, \dots in_j). \tag{1}$$

We explore the possibility of training such transformer models on new tasks without degrading the quality of solving the original tasks. We propose an approach to add new modalities to the model, both in the input and output data sequences.

To work with an intelligent agent control task, we include the modality of the agent's actions in the output. Additionally, we include the modality of the actions (i.e., the history of the agent's actions) and an optional reward in the input sequence, if it is being addressed using reinforcement learning. Thus, the input at timestep t is represented as $in^t = (in_1^t, \dots in_j^t) = (D, I^t, A^t, [r^t])$, where I^t is an image, D is a textual description of the task, A^t is the previous action (or the history of actions) of the intelligent agent, and r^t is an optional input – the immediate reward the agent received in the previous step during task execution. It is optional and is used for reinforcement learning.

The model should produce the next action:

$$A^{t+1} = out^{t+1} = \mathcal{LTM}(D, I^t, A^t, [r^t]). \tag{2}$$

One type of intelligent agent considered in this work is a robotic manipulator arm, which should be able to solve the intellectual task of object manipulation both in a simulator and in a real-world environment. Using a simulator allows the collection of many manipulator trajectories, varying the position and appearance of the manipulated objects. These data are then used to fine-tune the $\mathcal{LTM}$ model, which should be able to operate both in the simulator and on a real robotic stand.

In this work, a trajectory is considered as a sequence (I_1^t, I_2^t, P, A^t), where t is the index of the sequence element (timestep), I_1 and I_2 are images from the top and front robotic

[2]https://rozum.com/robotic-arm/#about

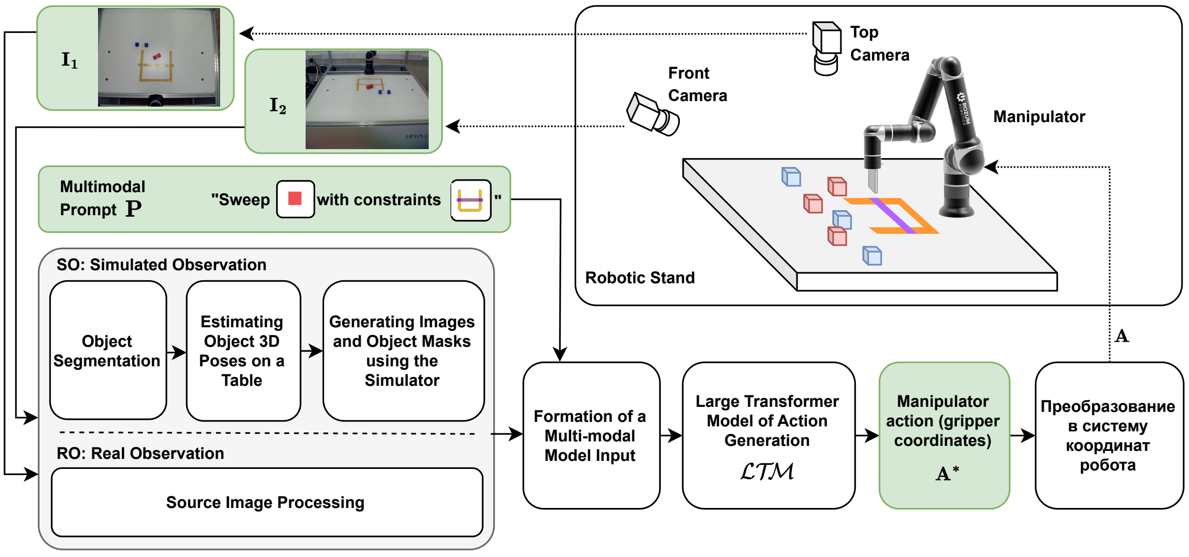

I1
I2
Top Camera
Front Camera
Manipulator
Robotic Stand
Multimodal Prompt P
"Sweep with constraints "
SO: Simulated Observation
Object Segmentation
Estimating Object 3D Poses on a Table
Generating Images and Object Masks using the Simulator
RO: Real Observation
Source Image Processing
Formation of a Multi-modal Model Input
Large Transformer Model of Action Generation
LTM
Manipulator action (gripper coordinates)
A*
Преобразование в систему координат робота
A

FIGURE 1. The overview of the proposed RozumFormer approach. We propose two approaches for transferring a multimodal transformer model $\mathcal{LTM}$ from a simulator to a real robot – Simulated Observation (SO) and Real Observation (RO). SO uses an intermediate step of generating synthetic images using a simulator and does not require fine-tuning of the $\mathcal{LTM}$ model on real data. RO uses direct preprocessing of images from the robotic stand camera but requires fine-tuning of the $\mathcal{LTM}$ model using previously collected data from the real robot.

stand cameras, P is a multimodal prompt for the entire sequence that does not depend on the timestep t. It includes both a textual task description and images of the target objects for manipulation, as well as any constraints that need to be taken into account. A represents the robotic arm's action, which consists of the (x, y) coordinates of the robotic arm's end effector that are sent to the robot's controller as the movement target.

To address the problem of transferring the behavior of a model trained in a simulator to control a real robot, we propose a new approach called RozumFormer, whose scheme is shown in Fig. 1. It enables the functionality of a large transformer model $\mathcal{LTM}$ trained on VIMA-Bench simulator data [13] to control the Rozum Pulse 75 robotic manipulator arm that is part of the real robotic stand.

As the $\mathcal{LTM}$ model is fine-tuned on the simulator data, it is worth noting the need to preprocess the images from the robotic stand cameras, I_1 and I_2, before inputting them into the neural network model. We consider two approaches, which we refer to as Simulated Observation (SO) and Real Observation (RO).

SO is based on the generation of synthetic images I_1^* and I_2^*, which allows for the use of pre-trained action generation models on simulator data. Using the initial images I_1 and I_2 from the robotic stand cameras, the target objects are segmented and their 3D poses in the workspace are determined. Then, according to the defined 3D poses, the corresponding target objects are placed in the simulator, and synthetic images I_1^* and I_2^* corresponding to the original images I_1 and I_2 are generated. The synthetic images I_1^* and I_2^* are inputted to the $\mathcal{LTM}$ model to generate the next action. This approach does not require fine-tuning of the $\mathcal{LTM}$ model on data collected from a real robot.

RO is based on the direct preprocessing of the original image from the top camera I_1 without a synthetic image generation step. This approach requires the fine-tuning of the $\mathcal{LTM}$ model on images from the real robotic stand camera, which leads to the necessity to collect data from the real robotic stand.

These approaches differ only in the way they process the images from the robotic stand cameras, the pipeline for generating robot actions remains the same.

Since the $\mathcal{LTM}$ model predicts actions A^* in simulator coordinates, to use it on a real robot, it is necessary to transform the predicted actions A^* into actions for a real robot's end effector A using the transformation $\mathcal{R}_{model \to robot}$:

$$A = \mathcal{R}_{model \to robot}(A^*). \tag{3}$$

The transformation $\mathcal{R}_{model \to robot}$ is determined through manual calibration.

IV. METHODOLOGY

A. ADAPTATION OF A LARGE BIMODAL TRANSFORMER MODEL TO CONTROL TASKS OF AN INTELLIGENT AGENT

To test the proposed RozumFormer approach, a pre-trained large bimodal transformer model with a specialized architecture was used. This bimodal transformer operates on both text and image modalities. It was initially pre-trained on tasks involving image captioning, image generation from text, and language modeling. It was then fine-tuned for six additional tasks: text question answering, mathematical problem solving, image generation from text, image captioning, visual question answering, and text recognition in unconstrained environments. As a result, the bimodal transformer with this specialized architecture initially solves a diverse range of text-visual tasks and has not been trained on the tasks of controlling an intelligent agent.

A feature of the architecture of the used bimodal transformer is the special organization of the input sequence of tokens used in training and inference. The input sequence is divided into three parts: left context, center context, and right context. In this case, the types of tokens used in each context are strictly fixed: only text tokens are transferred to the left context, only image tokens to the central context, and text tokens to the right context. They are filled differently depending on the current task. If the task is to answer a text question, then the question is placed in the left context, and the answer is generated in the right context. When generating an image based on a text description, the description is passed to the left context, and the central context is responsible for generating the image. When solving the image caption problem, the image is passed to the center context, and the right context is responsible for the caption. Thus, the combination of context usage depends on the type of problem to be solved.

In the process of adapting a bimodal transformer with a specialized architecture for tasks involving intelligent agent control, we encountered the challenge of preserving the original structure of the input sequence without altering the model's architecture. This was crucial for maintaining the ability to solve the initial tasks.

The feature of VIMA-Bench data [13] is that it uses a multimodal prompt to describe the problem to be solved. This prompt comprises a description of actions on target objects and images of the objects themselves. Additionally, in order to address the challenge of controlling an intelligent agent, it is necessary to provide the current state of the environment. Therefore, to adapt the bimodal transformer architecture with its specialized structure, we implemented the following steps. Initially, we partitioned the multimodal prompt into textual and visual components. The textual description of the task was inserted into the left context. For describing the state of the environment, we used an image from the top camera, combined it with images of the target objects from the prompt, and fed it into the central context. We allocated the right context for a new modality – agent actions.

The scheme of the proposed large transformer action generation model for the intelligent agent is shown in Fig. 2.

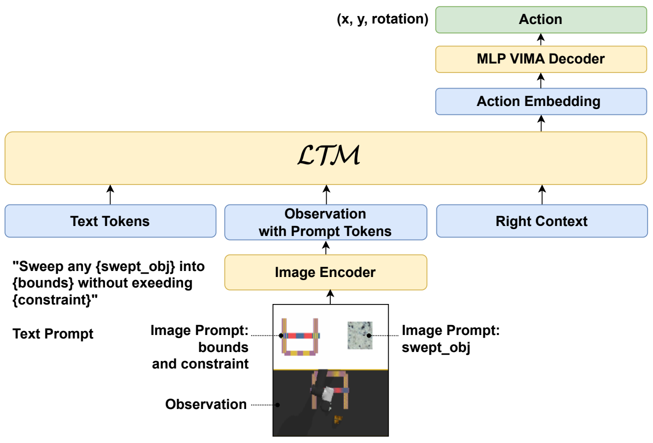


FIGURE 2. Scheme of an adapted architecture of a large bimodal transformer model RozumFormer for generating actions of an intelligent agent. We use the left context to provide a textual description of the task. The central context is provided with images of target objects from the multimodal prompt and the current state of the environment in the form of an image. The right context is used to predict the agent's action.

B. IMAGE SEGMENTATION FROM THE ROBOTIC STAND CAMERAS

In our work, we study two basic types of object segmentation in camera images: 1) object recognition by color features based on classical computer vision methods, and 2) deep neural network segmentation models that learn the object features automatically.

1) Image segmentation by color

First, we proposed an approach for color-based image segmentation using the HSV color space representation. It is illustrated in Fig. 3. HSV encodes color using three channels: Hue, Saturation, and Value. Hue corresponds to the color's shade, Saturation to its intensity, and Value to its brightness. To segment an object j of known color, it is necessary to set thresholds for each channel $(h^j_{min}, h^j_{max}, s^j_{min}, s^j_{max}, v^j_{min}, v^j_{max})$ in such a way that the color of the pixels in the object falls within the range defined by these thresholds.

To set the thresholds for the HSV channels, we first find the thresholds $(s^c_{min}, v^c_{min}), c \in [black, gray, white]$ for Saturation and Value that will remove black, gray, and white colors from the image. These colors have undefined Hue values. This means that Hue can take any value for these colors, so these colors need to be removed before setting the thresholds for Hue. The result of applying thresholds to the Saturation and Value channels is shown in Fig. 3. Note that when adjusting these thresholds, it is sufficient to adjust only the lower boundary and set the upper boundary to the maximum value (equal to one), since black, gray, and white colors have low Saturation or Value.

Before setting the Hue threshold, a region of interest is selected on the image, where all objects to be segmented will be located, as shown in Fig. 3. This simplifies setting the Hue threshold and avoids false positive detections during segmentation. Next, the Hue channel thresholds $(h^j_{min}, h^j_{max}), j \in O$ are adjusted. This adjustment must be done for each color j of the objects O that need to be segmented.

A Hue histogram is constructed to find the thresholds for the Hue channel, which shows the number of pixels in the image for each Hue value. The Hue thresholds for each color of objects should be set so that the corresponding peak falls within the range defined by the thresholds.

After segmentation, there is often a lot of noise – small, unnecessary segments that are false positives. This noise can be filtered out by applying a threshold to the contour length of the segment, denoted as $l^{contour}_{min}$. The contour length $l^{contour}$ of noise segments is much smaller than that of the target objects. Additionally, the contour length threshold can help to separate

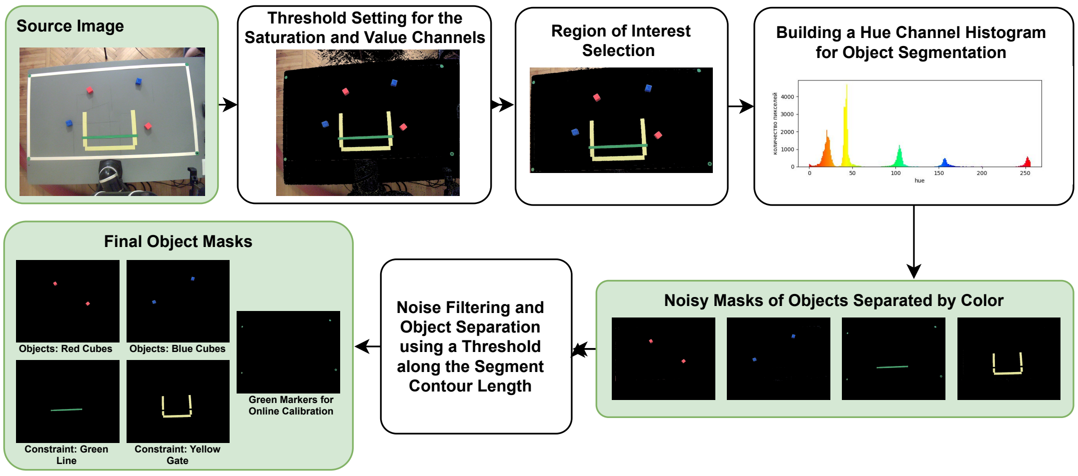

Source Image
Threshold Setting for the Saturation and Value Channels
Region of Interest Selection
Building a Hue Channel Histogram for Object Segmentation
hue
Noisy Masks of Objects Separated by Color
Noise Filtering and Object Separation using a Threshold along the Segment Contour Length
Final Object Masks
Objects: Red Cubes
Objects: Blue Cubes
Green Markers for Online Calibration
Constraint: Green Line
Constraint: Yellow Gate

FIGURE 3. **Scheme of the approach to segment objects in an image by color using the HSV (Hue, Saturation, Value) representation.**

different objects of the same color if they have different sizes, as is the case with the green boundary line and the green calibration markers (see Fig. 3).

This approach has limitations. For example, it may segment unwanted objects with colors similar to the target objects. To address this issue, one can remove all unnecessary objects from the scene, set a region on the image where segmentation will be performed, and filter the detected objects based on their area or contour length. In addition, if objects of the same color are placed closely together, they may be segmented as a single object.

2) Neural network-based object segmentation

To overcome the limitations of color-based object segmentation, we also explore segmentation methods based on deep neural networks. The most useful for solving the given task are convolutional methods capable of real-time operation. Examples of such neural networks are the classic Mask R-CNN model [34], the fast YOLACT-EDGE model [35], which takes into account a sequence of images, and the most recent YOLOv8 model [36]. Existing transformer models for object segmentation with a fixed number of objects, such as OneFormer [37], or models with an open vocabulary, such as OpenSeed [38], are not considered due to their limited computational efficiency. Semantic segmentation methods are also not considered, as they do not separate closely located objects of the same class.

C. ESTIMATING 3D COORDINATES OF OBJECTS ON THE TABLE

The task of determining the 3D coordinates of objects (colored cubes) on the table involves first determining their 2D coordinates observed in the image I_1 from the top camera and then adding a third coordinate that considers the table's fixed height. The obtained coordinates are used to place the objects in the simulator further so that the scene in the simulator replicates the real scene. This is necessary to generate observations from the simulator (see Fig. 1).

The idea of the approach to determine the coordinates of the cubes on the table is to find a transformation $\mathcal{T}_{I\to sim}$ that maps the pixel coordinates (u_I, v_I) in the camera image to the coordinates on the table (x_{sim}, y_{sim}) in the simulator. To find this transformation, a set of four points with known coordinates is chosen in both the camera image $p_I^1...p_I^4$ and in the simulator $p_{sim}^1...p_{sim}^4$. This transformation is represented by a 3×3 perspective transformation matrix and is computed in such a way that it maps a point from one set to the corresponding point in the other set:

$$\mathcal{T}_{I\to sim} = \mathrm{PerspTr}(\mathrm{Undist}(p_I^1...p_I^4, d_{par}), p_{sim}^1...p_{sim}^4). \quad (4)$$

The computation and application of such a perspective transformation is performed using the OpenCV library [39].

The four-point set is defined by green calibration markers placed at the corners of the working area, which can be seen in Fig. 3. The coordinates of the markers on the image are determined by the centers of the marker segments, which can be obtained by color-based segmentation. To order the markers, the marker with black color at its center (lower right marker) is identified first. This is achieved by exploiting the noise-filtering property of color-based segmentation, which results in filled spaces within the segments. By comparing the areas of the marker segments before and after noise filtering, the marker labeled with black color at its center can be determined, as its area after filtering is significantly larger than before filtering. This labeled marker is placed first in the set. The remaining markers are ordered in a clockwise manner. This ensures that the order of the markers remains consistent and independent of the camera orientation.

Before computing the perspective transformation based on two ordered sets of marker coordinates in the image and on the table in the simulator, correcting the distortion of the marker coordinates in the image is necessary. This is achieved by calibrating the camera parameters using the Kalibr tool[3] to

[3]https://github.com/ethz-asl/kalibr

determine the distortion parameters d_{par}, and then correcting for it.

The coordinates of the markers on the table of the robotic stand $(p^1_{robot}...p^4_{robot})$ are determined using the robotic arm. The robotic arm is brought to each marker on the table in the order they are arranged in the image. The coordinates of the robotic arm's end-effector (x_{robot}, y_{robot}) determine the coordinates of the marker it is positioned over. The coordinates of the markers on the table in the simulator are taken to be equal to the coordinates of the markers on the real table. In cases where the robotic arm cannot reach the markers, temporary markers can be used. These temporary markers should be placed closer to the robotic arm's workspace, and their positions can be used to find the positions of the outermost markers in the same way as the object positions will be determined.

To determine the position of objects on the table in the simulator, it is necessary to segment the object (e.g., a colored cube) in the image I_1, find the segment's center $(x^j_I, y^j_I), j \in O$, and transform the coordinates of the center using the found perspective transformation after distortion correction:

$$(x^j_{sim}, y^j_{sim}) = \mathcal{T}_{I\to sim}(\text{Undist}(x^j_I, y^j_I)). \quad (5)$$

D. GENERATING IMAGES AND MASKS OF OBJECTS USING THE SIMULATOR

Using the obtained coordinates $(x^j_{sim}, y^j_{sim}), j \in O$ of N objects O, they are added to the environment of the VIMA-Bench simulator. Subsequently, synthetic images I^*_1 of the top camera and masks of the detected objects $M^{1*}_1...M^{1*}_N$ are generated by the simulator. Similarly, images I^*_2 and object masks $M^{2*}_1...M^{2*}_N$ can be generated for the front camera. This approach makes a transition to the domain of simulator images on which the large transformer model was trained. It ensures consistent performance regardless of the camera placement, textures, and colors in the robot stand scene.

E. TRANSFORMATION OF THE ORIGINAL IMAGES

This paper also explores an alternative approach to preparing input images for the large bimodal transformer model RozumFormer directly from the color images of the top camera (option RO in the scheme in Fig. 1).

This approach does not require calibration and segmentation procedures, which significantly simplifies the data preparation process for the model and does not necessitate using a simulator. However, it does require collecting a natural dataset from a real robot stand, for which it is challenging to collect many training trajectories compared to a simulator.

The transformation of the original top camera image I_1 involves applying an affine transformation to it with manually chosen parameters p_{affine} to obtain an image I^*_1 that is close in the domain (geometric characteristics) to the top camera images from the VIMA dataset:

$$I^*_1 = \text{Affine}(I_1, p_{affine}). \quad (6)$$

F. COLLECTION OF DATASETS FOR TRANSFERRING THE MODEL TO A REAL ROBOT

1) Data collection on the robot

Since the RO approach requires data from a real robot to fine-tune the model, it is necessary to collect trajectories on a robot stand. Therefore, the Sweep-Plan dataset consisted of 100 trajectories where the agent successfully solved the ‘‘sweep_without_exceeding’’ task from VIMA-Bench [13] was collected. Each trajectory was a sequence of alternating observations and actions. The length of each trajectory was limited to 5 steps, consistent with the simulator.

The oracle agent provided by VIMA-Bench was used for this purpose. The oracle agent selected actions based on the positions of the cubes in the simulator, which were synchronized with their real positions on the table at each step. The positions of the cubes on the table were determined by the detector from images captured by the installed cameras. At the beginning of each episode, the cubes were arbitrarily placed on the table by a human expert.

2) Semi-automatic data labeling for scene recognition

Automated annotation was used to create the dataset for object segmentation in images from the top and front cameras of the robot stand. The first step involved the color-based segmentation procedure described earlier in the corresponding section, followed by the conversion of the obtained annotations into the CVAT 1.1 format [4]. In the second step, manual corrections were made to the object masks using the CVAT tool[5]. The following annotation errors were addressed: uneven or incomplete masks, merging of masks of adjacent objects (cubes) of the same color, absence of annotated objects, and incorrectly marked objects. The final annotations of the dataset were exported in the COCO format [40], which is suitable for further training neural network models.

The collected Sweep-Seg dataset for scene recognition includes 232 images with dimensions of 1280×1024 pixels. The original annotations consist of 232 grayscale uint16 images, where the higher byte encodes the object class, the lower byte encodes its number, and a 0-byte indicates the background. The data contains four categories of objects (red cube, blue cube, yellow Π-shaped boundary line, and straight boundary line). The dataset is divided into two parts: a training set of 187 images and a test set of 45 images. The test images were chosen equally for the top and front cameras and for different colors of the boundary line.

V. TOOLS AND DATA USED

A. BASELINE

As a baseline, we use the VIMA model [13] proposed together with the VIMA-Bench benchmark [13]. VIMA is a transformer that takes as input the sequence of cropped object images presented on the scene and the agent's actions. Cross-attention layers are used to consider the multimodal prompt,

[4]https://opencv.github.io/cvat/docs/manual/advanced/xml_format/#annotation

[5]https://opencv.github.io/cvat/docs/

which alternates with causal self-attention layers. The multimodal prompt processes the frozen T5 model [41], after which the resulting embedding is fed into the input of the cross-attention layers. VIMA is trained on all tasks from the VIMA-Bench benchmark. The model options differ in the parameters: 2 m, 20 m, and 200 m, designated VIMA 2M, VIMA 20M, and VIMA 200M, respectively.

B. DATASETS

1) VIMA-Bench simulation data

The main experiments on fine-tuning the bimodal transformer were conducted using data from the VIMA-Bench simulator[6]. VIMA-Bench includes 17 types of manipulation tasks of varying complexity, ranging from simple object transfer (*visual_manipulation*) to restoring the original arrangement of objects relative to each other (*rearrange_then_restore*). A distinctive feature of VIMA data is the presence of a multimodal prompt (text and image) that specifies the type of task, the objects that need to be manipulated, and the short length of the trajectories. An action is defined by the initial and final position of the robotic arm's gripper. This paper conducted experiments for object movement without crossing a boundary (''sweep_ without_exceeding''), with training performed on 50,000 trajectories.

2) RozumCube dataset

As an enhancement to the VIMA-Bench simulation data, we have compiled a unique dataset, the episodes requiring more precise actions from the model to fulfill the task. The VIMA-Bench dataset primarily consists of episodes where distractor cubes are all positioned on one side and swept objects on the opposite side of the table. In contrast, the RozumCube dataset episodes invariably contain four cubes that can be dispersed across the table in any arrangement. We generated 30,000 episodes using a script that uniformly distributes cubes across the table surface.

3) Real robot data

To transfer the model from the simulator to the real robot, fine-tuning was performed using a dataset called Sweep-Plan collected by the Oracle agent on the robot stand. The dataset consists of 100 trajectories. To address the object segmentation task in the images from the cameras on the robot stand, the Sweep-Seg dataset was developed, consisting of 232 annotated images. Its detailed description is given in the previous section.

4) The Atari environments for controlling intelligent agents

The proposed approach was validated on the game Breakout from the Atari 2600 environment [14]. The task was formulated as an Offline Reinforcement Learning (RL) problem, where the agent's goal is to achieve the most efficient behavior based on fixed, limited experience. The training data consisted of trajectories (replay dataset) generated by the DQN model [42], trained on 200 million frames in each of the 60 Atari environments. In this paper, we conduct experiments on the Atari Breakout environment since this is the most popular Atari environment used. We use rewards to measure the quality of the learned policy.

[6]https://github.com/vimalabs/VimaBench

5) Textual and visual-linguistic data

The bimodal transformer was trained to solve textual and visual-linguistic (VL) tasks. To test the hypothesis that training with a mixed batch has only a slight impact on the quality of solving the original tasks, we mixed in data from the original tasks: Text Question Answering (TextQA) – 45,328 examples from SQuAD [43], Mathematic QA (MathQA) – 100,000 examples from [44], image generation and image captioning (ImageGen, ImageCap) – 82,783 examples from the COCO dataset [45], visual question answering – 85,759 examples from the Visual Genome dataset [46], text recognition (TextRec) – 40,312 examples from COCO [45].

For TextQA and MatQA tasks, the F1 score is used. The METEOR [47] score measures the quality of the generated answers and captions for VQA and image-captioning tasks. The Fréchet inception distance (FID) [48] is used to evaluate the generated images. For text recognition in the wild, the Normalized edit distance (NED) is used.

C. ROBOTIC STAND

The experimental robotic stand comprises a table, a Rozum PULSE 75 robotic manipulator arm, and two SVEN IC-545 cameras. The robotic arm has six degrees of freedom and a working radius of 750 mm. It can move the gripper to a specified point with a specified orientation, with a repeatability of 0.1 mm. A spatula is attached to the gripper of the robotic arm and is used to move objects on the table surface. The cameras have a focal length of 4.3 mm and capture images of the table from above and from the side with a 1280×1024 pixels resolution. The manipulation objects are red and blue cubes in a rectangular area of 100×50 cm (Fig. 4). The dimensions and structure of the testbed replicate the scene in the VIMA-Bench simulator (the placement of the cameras and the dimensions of the working area coincide with an accuracy of 0.5 cm). Still, the textures and colors of the surfaces differ significantly.

The ''sweep_without_exceeding'' task was run in the experimental setup in two configurations: with real observations (RO) and with observations obtained by a simulator (SO). In both configurations, the model selected an action as a pair of points on the table based on the observation $p_1, p_2 \in \mathbb{R}^2$. Then the robotic arm moved the end of the spatula from p_1 to p_2 at a fixed height.

In the SO configuration, the observation consisted of images generated by rendering the scene in the simulator. The state of the scene in the simulator, defined by the positions of the cubes, was mapped to the state of the real scene on the standing table through cube detection and determination of their positions relative to the camera. In the RO configuration,

TABLE 1. Quality of object segmentation methods on the Sweep-Seg test sample.

Method	mAP50-95-B	mAP50-B	mAP50-9-M	mAP50-M	InfTime, s
ColorSeg	0.7890	0.8970	**0.7780**	0.8770	<u>0.0271</u>
Mask R-CNN	0.7610	0.9150	0.6670	0.9180	0.0745
YOLOv8n	0.8543	0.9536	0.6606	0.9391	**0.0260**
YOLOv8s	<u>0.8852</u>	**0.9632**	0.6829	**0.9576**	0.0280
YOLOv8m	**0.8890**	<u>0.9626</u>	<u>0.6833</u>	<u>0.9539</u>	0.0383

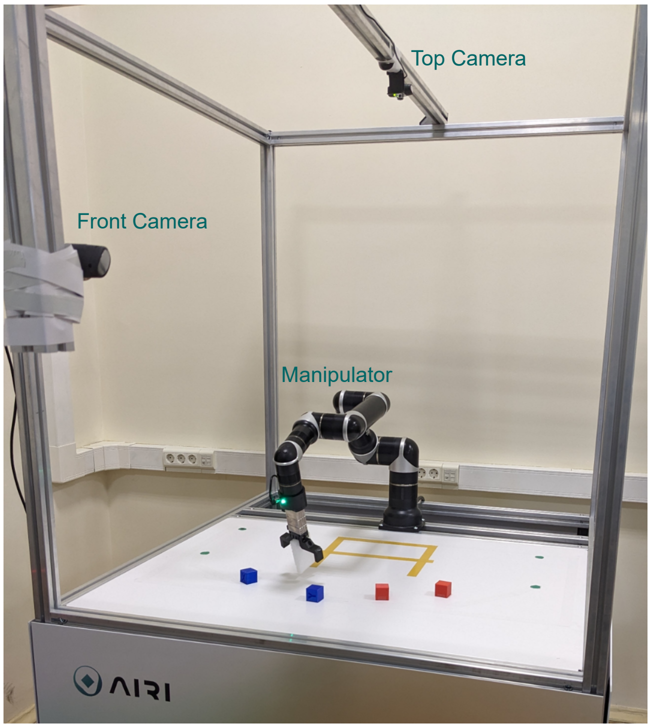


FIGURE 4. The experimental robot stands with the Rozum PULSE 75 robotic manipulator arm and two SVEN IC-545 cameras for the task "sweep_without_exceeding" from VIMA-Bench.

images from the cameras were used directly to make the observation.

VI. EXPERIMENTAL RESULTS

A. RESULTS OF SEGMENTATION AND DETECTION OF SCENE OBJECTS

The training of object segmentation models on the images from the robotic stand cameras was performed on a workstation with a CPU Intel i5 6 core 2.90 GHz and an Nvidia GPU GeForce RTX3060 with 12GB of memory.

The following high-performance segmentation models were selected for training: Mask R-CNN [34] with ResNet50_FPN backbone, YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium) [49]. All versions of YOLOv8 were trained for 200 epochs with a learning rate of 0.01. The Mask R-CNN model was also trained for 200 epochs with a learning rate 0.02. Table 1 shows the metrics for the best weights of each model, the results of the color-based segmentation (ColorSeg), and the inference time (InfTime). After training, the YOLOv8s model showed the best metrics on the developed dataset, Sweep-Seg test set. In addition, the average time to process an image differs slightly from the best-performing YOLOv8n in this parameter.

Examples of the final annotations and masks obtained by color-based segmentation, Mask R-CNN, and YOLOv8s for images from the test set are shown in Fig. 5. It can be observed that the color-based segmentation masks and those obtained using the YOLOv8s neural network model are almost identical and are close to the ground truth masks, except in cases where objects of the same color are located nearby. Here, the color-based segmentation method erroneously merges them into one object. Additionally, the figure shows that YOLOv8s most accurately determines the masks of annotated objects, while the Mask R-CNN model can hardly detect the red cubes.

B. PERFORMANCE RESULTS OF THE LARGE BIMODAL TRANSFORMER MODEL ON THE ROBOTIC MANIPULATOR ARM

Given that the RozumFormer model necessitates input unification with only the prompt text and a single image, we have preprocessed the task-defining data in the following manner. The left text context incorporates the text prompt, which provides information about the number and color of cubes that need to be swept. The right text context is utilized for outputting actions. An action is characterized by two coordinates: the starting point and the endpoint of the spatula movement. To discretize the grid, we divided it into intervals of 2 cm, assigning a unique token in the dictionary for each value. As only one image can be input into the middle part, and a minimum of three images are required for a comprehensive task description (scene view, texture of the target cube, and texture of the area where the cube should end up), we amalgamated this essential information into a single image (refer to Fig. 7) and tokenized it using the frozen VQ-GAN model.

1) Simulated Observation approach

An illustration of the task execution in an SO mode is depicted in Fig. 6. Initially, we assess our approach in a simulator on 500 randomly sampled episodes from the VIMA-Bench and RozumCube dataset. The comparative results between the proposed RozumFormer model and VIMA are presented in Table 2. Following the adaptation of the model to the conditions of the actual robotic arm, we evaluated the success rate of 15 random episodes on the real robot, the results are shown in Table 3.

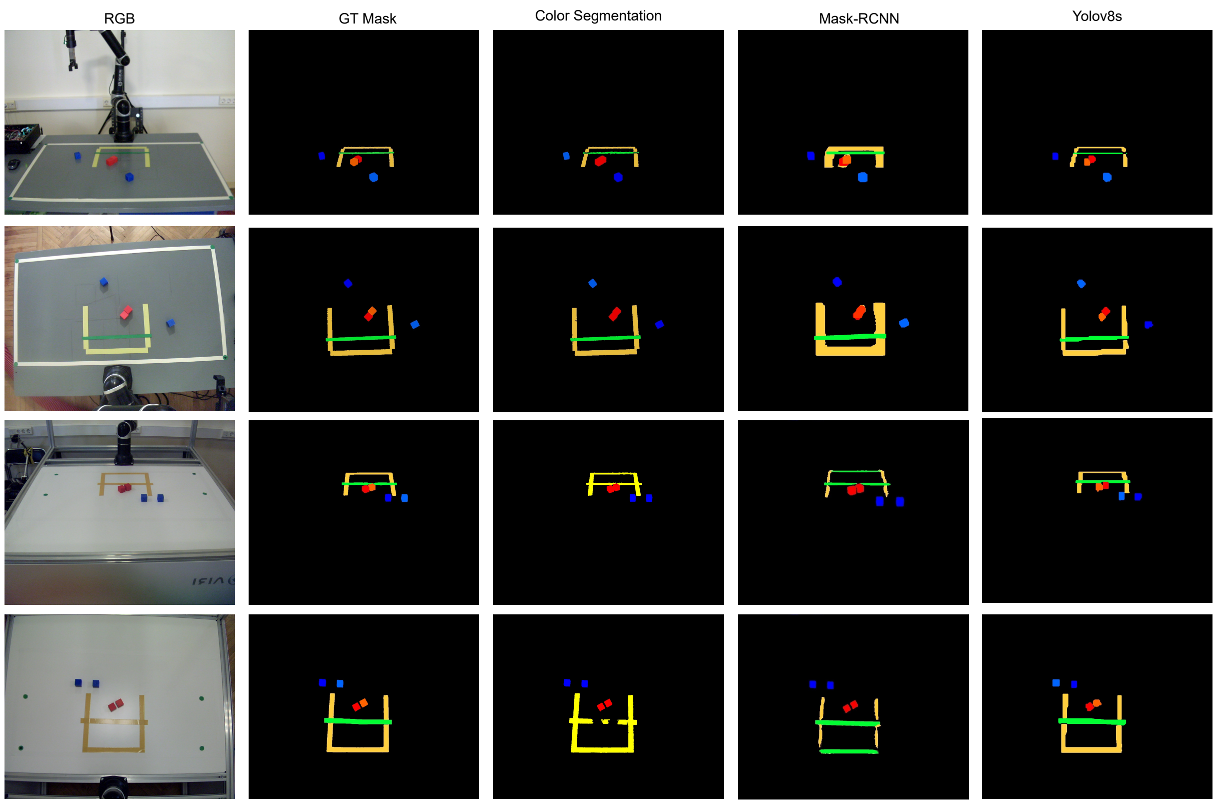


FIGURE 5. Examples of object segmentation on images from Sweep-Seg dataset for various methods.

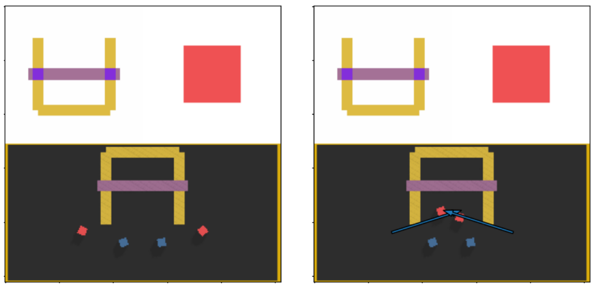


FIGURE 6. Example of the execution of the "sweep_without_exceeding" task in SO mode. On the left is the initial scene configuration in the format that is fed to RozumFormer. On the right is the result of the episode, with each arrow indicating the action taken.

TABLE 2. Success Rate of models trained on simulated data in a simulation environment on the different datasets.

Model	VIMA-Bench test	RozumCube
VIMA 2M	0.79	0.40
VIMA 20M	0.91	0.87
VIMA 200M	**0.96**	0.90
RozumFormer	0.92	**0.94**

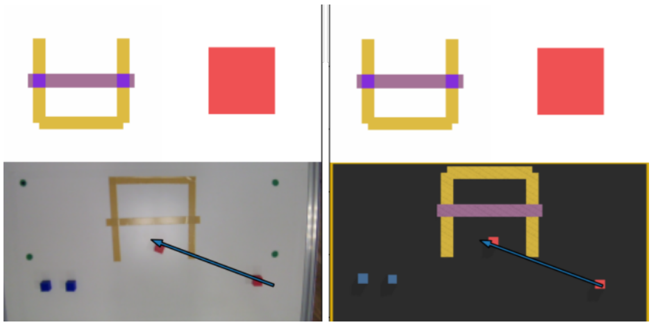


FIGURE 7. An example of adapting the "sweep_without_exceeding" task to the conditions of the real robot using the RozumFormer-SO approach. On the left is the original image from the top camera, and on the right is how the reconstructed scene looks. The arrow shows the action suggested by the model.

TABLE 3. Success Rate when using the Simulated Observation approach. To assess the model's quality, cube configurations similar to those in the VIMA-Bench dataset were placed in the workspace of the robotic arm. Models are trained on simulation data only.

Model	VIMA-like real poses
VIMA 200M	0.80
RozumFormer	**0.93**

2) Real Observation approach

We also test RozumFormer in Real Observation mode. To do this, we additionally fine-tuned the model used in the Simulated Observation approach on the Sweep-Plan dataset collected on a real robot. After that, we tested RozumFormer on 15 episodes on a real robot. For this purpose, cube configurations corresponding to the configuration from the RozumCube dataset were manually set in the robot's workspace. An episode was successful if the robot moved cubes of the

TABLE 4. Results on text-visual tasks and Atari Breakout with different batch generation methods during training.

Task	Metric	Atari only	Equal mixed	VL-tasks
TextQA	F1	0.003	0.259	**0.272**
VQA	METEOR	0.006	0.356	**0.364**
MathQA	F1	0.005	0.301	**0.315**
ImageCap	METEOR	0.008	**0.240**	0.237
ImageGen	FID	0.006	0.273	**0.281**
TextRec	NED	0.004	**0.368**	0.363
Breakout	Reward	**173.4**	171.8	—

target color in the desired area without exceeding boundaries and did not move cubes of a different color. The model fine-tuned on the Sweep-Plan dataset shows a Success Rate of **0.73**, which demonstrates a good generalization to a real robot given the small size of the collected on the real robot dataset.

C. RESULTS OF TESTING ON THE ATARI GAME ENVIRONMENT DURING TRAINING WITH A MIXED BATCH

Table 4 shows the results of training the bimodal transformer only on the Atari Breakout game (Atari only), with a mixed batch (Equal mixed), and only on the original tasks (VL-tasks).

Regardless of the training mode, RozumFormer demonstrates impressive performance in the game Breakout, scoring 173.4 when trained only on Atari data and 171.8 when using an equally mixed batch. This is significantly superior to the results obtained by Decision Transformer [6], an approach to Offline Reinforcement Learning, which shows a score of 76.9 $\pm$ 27.3 on Breakout.

However, As can be seen from the results, using only Atari data for training significantly degrades the quality of the original tasks. This degradation can be mitigated by mixing the data of the original tasks in equal proportions during training. Thus, training the model on a new task and incorporating data from the original tasks into the batch allows us to broaden the range of tasks the model can solve with only a minor reduction in performance on the original tasks.

VII. CONCLUSION

The paper proposes an approach to adapt a large pre-trained bimodal (text-image) transformer architecture, trained on textual and visual-linguistic tasks, to control an intelligent agent, which we call RozumFormer. RozumFormer involves the formation of a specially designed input token sequence consisting of a multimodal description of the task, the agent's observations, and its actions while preserving the input structure dictated by the architectural characteristics of the transformer used. This approach is demonstrated in solving the object manipulation task using a robotic manipulator arm in both virtual and real environments.

To transfer a model trained on simulator data to a real robot, we propose Simulated Observation and Real Observation. The Simulated Observation approach does not require additional fine-tuning of the model on real robot data but includes an additional step of generating synthetic images based on the real state of the environment. In the Real Observation approach, there is no step in generating synthetic images. Still, it is necessary to fine-tune the model on real data, which requires collecting trajectories on a robot stand. Using the Real Observation approach, RozumFormer demonstrates a Success Rate of 0.73 with additional fine-tuning on the Sweep-Plan dataset collected on a real robot. This demonstrates the successful transfer of the model from the simulator to a real robot stand, considering the size of the collected dataset.

We also present results for controlling an intelligent agent in the Atari game environment. RozumFormer shows results superior to such a reinforcement learning approach as Decision Transformer. We also show that when training with an equally mixed batch containing not only Atari data but also the data from the original tasks, we obtain comparable results on the target task only with a slight degradation on the original tasks. This demonstrates that using a mixed batch helps to efficiently expand the set of solvable tasks.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[5] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[7] A. Suglia, Q. Gao, J. Thomason, G. Thattai, and G. Sukhatme, "Embodied BERT: A transformer model for embodied, language-guided visual task completion," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2108.04927

[8] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[9] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[10] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[11] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[12] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[13] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," 2022. [Online]. Available: https://arxiv.org/abs/2210.03094

[14] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[16] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[17] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters*, 2023.

[18] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.

[19] K.-H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarrama, I. Fischer, E. Jang, H. Michalewski *et al.*, "Multi-game decision transformers," *arXiv preprint arXiv:2205.15241*, 2022.

[20] Z. Jiang, T. Zhang, M. Janner, Y. Li, T. Rocktäschel, E. Grefenstette, and Y. Tian, "Efficient planning in a compact latent action space," *arXiv preprint arXiv:2208.10291*, 2022.

[21] A. Bessonov, A. Staroverov, H. Zhang, A. K. Kovalev, D. Yudin, and A. I. Panov, "Recurrent memory decision transformer," *arXiv preprint arXiv:2306.09459*, 2023.

[22] Y. Sun, S. Ma, R. Madaan, R. Bonatti, F. Huang, and A. Kapoor, "Smart: Self-supervised multi-task pretraining with control transformers," *arXiv preprint arXiv:2301.09816*, 2023.

[23] K. Hu, R. C. Zheng, Y. Gao, and H. Xu, "Decision transformer under random frame dropping," *arXiv preprint arXiv:2303.03391*, 2023.

[24] V. Micheli, E. Alonso, and F. Fleuret, "Transformers are sample efficient world models," *arXiv preprint arXiv:2209.00588*, 2022.

[25] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[26] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[27] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[28] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[29] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, 2021.

[30] Y. Liu, H. Xu, D. Liu, and L. Wang, "A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102365, 2022.

[31] N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang, "Real–sim–real transfer for real-world robot control policy learning with deep reinforcement learning," *Applied Sciences*, vol. 10, no. 5, p. 1555, 2020.

[32] Y. Chen, C. Zeng, Z. Wang, P. Lu, and C. Yang, "Zero-shot sim-to-real transfer of reinforcement learning framework for robotics manipulation with demonstration and force feedback," *Robotica*, vol. 41, no. 3, pp. 1015–1024, 2023.

[33] S. W. Abeyruwan, L. Graesser, D. B. D'Ambrosio, A. Singh, A. Shankar, A. Bewley, D. Jain, K. M. Choromanski, and P. R. Sanketi, "i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops," in *Conference on Robot Learning*. PMLR, 2023, pp. 212–224.

[34] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[35] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "Yolactedge: Real-time instance segmentation on the edge," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9579–9585.

[36] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, 2023.

[37] J. Jain, J. Li, M. T. Chiu, A. Hassani, N. Orlov, and H. Shi, "Oneformer: One transformer to rule universal image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2989–2998.

[38] H. Zhang, F. Li, X. Zou, S. Liu, C. Li, J. Gao, J. Yang, and L. Zhang, "A simple framework for open-vocabulary segmentation and detection," *arXiv preprint arXiv:2303.08131*, 2023.

[39] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[41] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[42] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 104–114.

[43] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," 2018. [Online]. Available: https://arxiv.org/abs/1806.03822

[44] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," *arXiv preprint arXiv:1904.01557*, 2019.

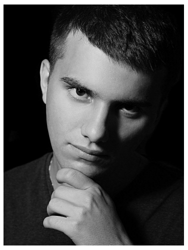
[45] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014. [Online]. Available: https://arxiv.org/abs/1405.0312

[46] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International journal of computer vision*, vol. 123, pp. 32–73, 2017.

[47] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909

[48] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

[49] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics



ALEKSEI STAROVEROV received an M.S. degree from Bauman Moscow State Technical University, Moscow, Russia in 2019. He is currently pursuing a Ph.D. degree in computer science at the Moscow Institute of Physics and Technology, Moscow, Russia. His research thesis involves the methods and algorithms for the automatic determination of subgoals in a reinforcement learning problem for robotic systems.

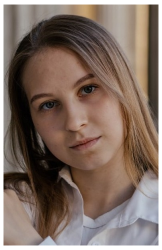
From 2022 to the present, he has been working as a Researcher at the Artificial Intelligence Research Institute, Moscow, Russia. His research interests include reinforcement learning, deep learning, and robotic systems.

ANDREY S. GORODETSKY received Master's degree in mechatronics and robotics from Bauman Moscow State Technical University, Moscow, Russia in 2021.

Since 2021, he has been a PhD Candidate at the Cognitive Dynamic Systems Laboratory at the Moscow Institute of Physics and Technology, Moscow, Russia. His research interests include model-based reinforcement learning.

ANDREY S. KRISHTOPIK received Bachelor's Degree in applied mathematics and physics in Moscow Institute of Physics and Technology, Moscow, Russia, in 2020.

Since 2019, he is a Researcher at the Cognitive Dynamic System Laboratory at the Moscow Institute of Physics and Technology, Moscow, Russia. His research interests include SLAM algorithms.



ULIANA A. IZMESTEVA received a Bachelor's Degree with a specialty "software engineering" at Kalashnikov Izhevsk State Technical University, Izhevsk, Russia, in 2023.

From august 2023, she is an Intern Researcher at the Center of Cognitive Modeling of the Moscow Institute of Physics and Technology, Moscow, Russia. Her research interests include image segmentation and object detection.

DMITRY A. YUDIN received the engineering diploma in automation of technological processes and production in 2010 and the Ph.D. degree in computer science from the Belgorod State Technological University (BSTU) named after V.G. Shukhov, Belgorod, Russia in 2014.

From 2009 to 2019, he was a Researcher and Assistant Professor with Technical Cybernetics Department at BSTU n.a. V.G. Shukhov. Since 2019, he has been the head of the Intelligent Transport Laboratory at the Moscow Institute of Physics and Technology, Moscow, Russia. Since 2021, he has been a Senior Researcher at AIRI (Artificial Intelligence Research Institute), Moscow, Russia. He is the author more than 100 articles. His research interests include computer vision, deep learning, and robotics.

ALEXEY K. KOVALEV received the engineering diploma in robots and robotic systems from MIREA – Russian Technological University, Moscow, Russia in 2012, Master's Degree in control in technical systems from MIREA – Russian Technological University, Moscow, Russia in 2015, and the Ph.D. degree in computer science from HSE University, Moscow, Russia in 2022.

Since 2017, he is a Junior Researcher at the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia. Since 2021, he is a Researcher at AIRI, Moscow, Russia. His research interests include embodied AI, offline reinforcement learning, and multimodal models.

Dr. Kovalev has been a member of the Russian Association of Artificial Intelligence since 2018.

ALEKSANDR I. PANOV earned an M.S. in Computer Science from the Moscow Institute of Physics and Technology, Moscow, Russia, 2011 and a Ph.D. in Theoretical Computer Science from the Institute for Systems Analysis, Moscow, Russia, in 2015.

Since 2010, he has been a research fellow with the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia. Since 2018, he has headed the Cognitive Dynamic System Laboratory at the Moscow Institute of Physics and Technology, Moscow, Russia. He authored three books and more than 100 research papers. In 2021, he joined the research group on Neurosymbolic Integration at the Artificial Intelligence Research Institute, Moscow, Russia. His academic focus areas include behavior planning, reinforcement learning, embodied AI, and cognitive robotics.

• • •