Received 9 December 2023, accepted 1 January 2024, date of publication 4 January 2024,
date of current version 11 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3350038

RESEARCH ARTICLE

Inpainting Semantic and Depth Features to Improve Visual Place Recognition in the Wild

ILIA SEMENKOV[1,2], **ALEKSEI KARPOV**[2], **ANDREY V. SAVCHENKO**[3,4,5,6], **AND ILYA MAKAROV**[2,5,6]

[1]School of Data Analysis and Artificial Intelligence, HSE University, 109028 Moscow, Russia
[2]Artificial Intelligence Research Institute (AIRI), 105064 Moscow, Russia
[3]Sber AI Lab, 121170 Moscow, Russia
[4]Laboratory of Algorithms and Technologies for Network Analysis, HSE University, 603093 Nizhny Novgorod, Russia
[5]AI Center, NUST MISIS, 119049 Moscow, Russia
[6]ISP RAS Research Center for Trusted Artificial Intelligence, 109004 Moscow, Russia

Corresponding authors: Ilia Semenkov (isemenkov@hse.ru), Andrey V. Savchenko (avsavchenko@hse.ru), and Ilya Makarov (iamakarov@hse.ru)

This work was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Ivannikov Institute for System Programming of the Russian Academy of Sciences dated November 2, 2021 No. 70-2021-00142.

ABSTRACT Visual place recognition is one of the core modern computer vision tasks concerned with identifying location based on the image taken there. Modern state-of-the-art approaches heavily rely on RGB images which are largely affected by changes in the same scene such as varying daytime, illumination, seasonal changes, and presence of dynamic objects (people, vehicles). This results into a large difference between the images in the training dataset and the ones taken by a person in real life at the same place as a part of some application, rendering modern approaches less effective. To deal with this problem, we propose a novel approach that uses only geometrical information (shapes of buildings, terrains, trees, and their relevant positions) obtained from depth and semantic maps inpainted to remove dynamic objects. In this paper, we study two versions of the pipeline: the first one uses direct inpainting, and the second utilizes synthetic data to improve the inpainting process. Our most efficient model achieved 60.6% correct answers with synthetic refinement. With direct inpainting, it kept metrics high at 51.1%. With these compelling results, our approach offers a novel and effective alternative to known algorithms, making it an exciting avenue for future research in visual place recognition.

INDEX TERMS Visual place recognition, image retrieval, inpainting, semantic segmentation, monocular depth estimation, NetVLAD, CosPlace.

I. INTRODUCTION

One of the problems that recently gained popularity in computer vision is the visual place recognition [1], [2], [3], [4], [5]. This task aims to find a location where an image was taken based only on the image itself. It is typically done with the image retrieval approach [6], [7], [8], [9], [10]. More precisely, a large index database of images with known locations (for example, coordinates for every picture) should be collected preliminary. Then, a model is trained to create embeddings for images that preserve features important for location identification. During inference, a query image is fed into a model. An embedding of this query image is being compared with embeddings of images in an index database according to the chosen similarity metric (typically, Euclidean distance or cosine similarity). The closest image from a database is picked and considered from a similar location to the query. Consequently, a known location of a

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

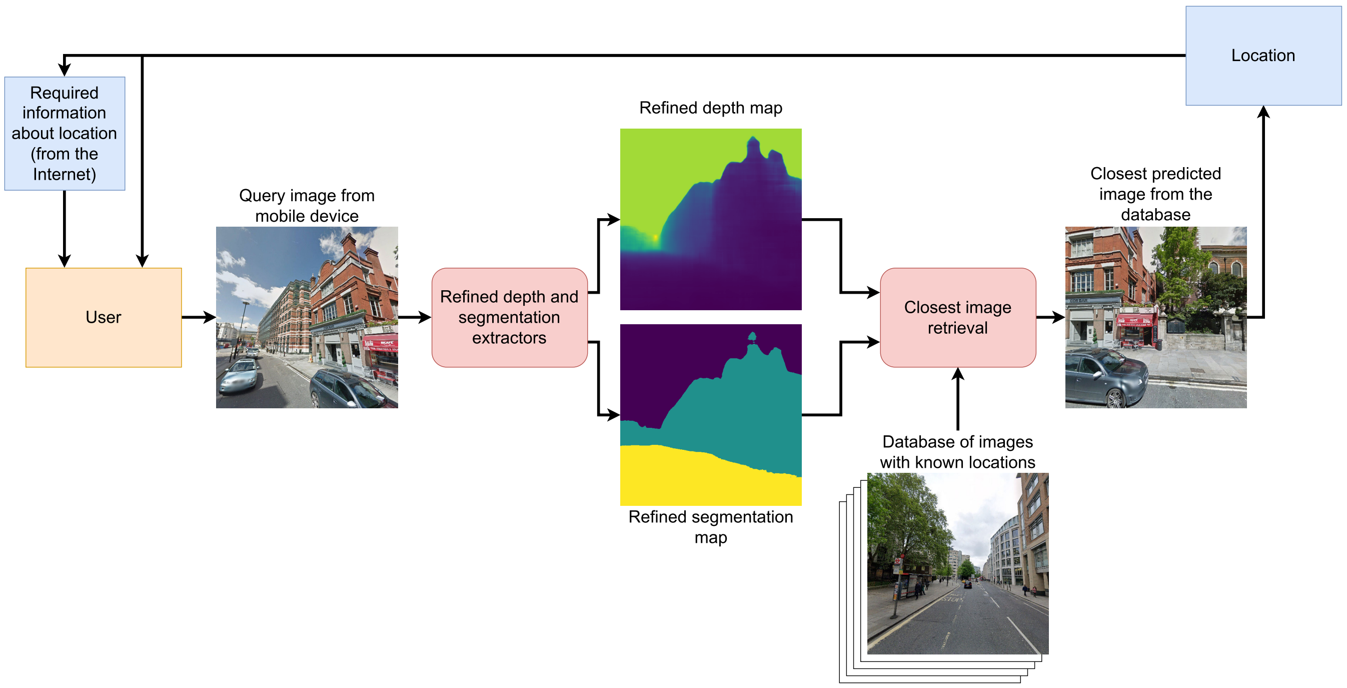

Location
Required information about location (from the Internet)
Refined depth map
Query image from mobile device
Closest predicted image from the database
User
Refined depth and segmentation extractors
Closest image retrieval
Database of images with known locations
Refined segmentation map

FIGURE 1. The overall pipeline schema with a proposed application. Firstly, a user takes a picture in a location they are interested in. Then, we extract refined depth and segmentation maps with neural networks. We call them refined as they are unaffected by dynamic objects (notice the lack of vehicles on depth and semantic maps). Then, an image retrieval model finds an image in the index database that is the most similar to the query. After, its location is being extracted. Finally, the location and additional information about it from the Internet are being sent back to the user.

picked image is output as an estimate of the location of a query image.

However, visual place recognition remains a challenging task due to several reasons. Firstly, even relatively small differences in distance or angle between two images can cause significant variations in a view. For example, if a camera operator stands in the same spot but turns camera 30°, a completely different set of buildings, roads, and green spaces might be presented. The same thing might happen if a cameraperson moves 10 meters from a current spot. A model should be able to handle all of these sensitive changes instead of dropping in performance because a camera is slightly carried away. That creates a need to collect many images taken very close to each other at various angles per location. Secondly, image illumination changes can significantly affect the quality of visual place recognition, mainly if the illumination in queries differs from a corresponding image in a database. For instance, recognition during nighttime might be challenging if a model is mainly trained on daytime images, especially if there are many colored illumination sources like neon lights, light poles (which in many countries use sodium lamps leading to yellow illumination), or digital billboards. Finally, images in RGB (Red-Green-Blue) format have a lot of fine details that may change reasonably often in the real world, such as billboards, road signs, advertisements, etc. Initially, hidden periodic processes were used to deal with the spatiotemporal dynamics of the environment, e.g., spectral analysis [11], [12]. Nowadays, such changes might be learned as they are effective location clues. For example, a model can use a pizzeria logo in an image to recognize a place because a small percentage of locations in an index database have pizzerias with similar logos in their images. If a restaurant shuts down, the model's performance will drop for this image because it still focuses on the now-defunct logo in a picture in a database. Additionally, natural details that are noticeable in RGB images, such as snow, change from time to time. Moreover, the number of people and vehicles differs depending on the daytime and seasons. At the same time, people and vehicles themselves in images vary all the time. Consequently, the "frozen" dataset results may not generalize well for real-world applications in dynamic environments.

One possible way to overcome the challenges of dynamic environments (scene and illumination changes) is to build a large dataset containing images of the same places but collected at different times of the day in various weather conditions [13]. Such a task is challenging, expensive, and poorly scaled. For example, authors of the mentioned Oxford RobotCar Dataset spent over a year collecting 20 million images for the 10 km long route [13]. On top of that, even if such a dataset is available for a large area (e.g., a whole city), it will take enormous time and computing power to train a model on it. However, the problem with data collection is vastly amplified in real-life applications. It happens because a database must always be fresh and up to date to mitigate the impact of some long-term non-cyclic changes, which implies even more constant data collection. Therefore, the main drawback of this approach is that collecting large datasets is very time-consuming and expensive, even for a relatively small area, because not only does a dataset have to contain a vast amount of images, but it also needs to cover a whole town uniformly with reasonably low distances between scenes.

Consequently, a lot of information in RGB images is affected by changes in the same scene, such as varying daytime, illumination, seasonal changes, and the presence of dynamic objects (people, vehicles). And it is almost impossible to accommodate all of these changes by simply adding more data. Every location will have to be repeated a colossal number of times under different circumstances. In addition, a massive number of locations is needed to make sure there is an entry in the index database close to a potential query. This results into a big difference between the images in the used dataset and queries taken by a person in real life at the same place.

It seems that the issue of dealing with such dynamic changes during geolocation is not well reported. Most of the current works accept the paradigm with RGB images and popular benchmarks. Our paper tries to fill this research gap. In particular, we propose the novel pipeline that relies mainly on geometrical information and less on particular details (Fig. 1). Indeed, some problems mentioned above can be overcome using depth sensors [14] or depth estimation in supervised [15], [16] and self-supervised learning settings [17] instead of RGB images. Adding segmentation maps to depth information allows us to capture the main features of objects, such as shape and relative position. Furthermore, they are more resilient to visual changes that do not affect the shapes of buildings or roads. Additionally, depth sensors are more robust to illumination or time of day changes as they do not capture visual information.

Our approach performs localization based on depth and segmentation maps coupled with a refining strategy, the goal of which is to reduce the amount of dynamic objects in the scene. Firstly, a user takes a query picture. Then, we apply the model which extracts refined depth and segmentation maps. Refining means it will use inpainting to remove the effects of dynamic objects on depth and semantic maps. In the final part, a model will use already prepared refined depth and segmentation maps of images from an index database to find the closest to the presented query. Finally, when the model succeeds, it can return the known location of the closest database item to a user with optional information about it. Therefore, the main contributions of this paper are as follows:

- We propose a novel approach to GIS image retrieval that relies solely on geometrical information obtained from depth and semantic maps.
- We provide a novel pipeline based on inpainting techniques to minimize the impact of dynamic objects.
- We highlight the usefulness for synthetic data to improve the inpainting pipeline quality. In particular, it could allow reducing the amount of real images needed for training and an index database.

To sum up, we propose a novel image retrieval pipeline that can be used for geopositioning and navigating in the absence of the Global Positioning System (GPS). The novelty of our approaches lies in visual recognition systems based on depth estimation and semantic segmentation. The importance of such a system in practical applications lies in the novel opportunities for augmented reality, and Self-driving GIS (Geographic Information System) services robust to the absence of precise local geopositioning from GPS.

II. RELATED WORKS

A. SEMANTIC SEGMENTATION

The problem of semantic segmentation is one of the classic problems in computer vision [18], [19], [20], [21]. It challenges the algorithm to assign one of the predefined classes to every pixel on the image. This problem was successfully addressed with the development of convolutional neural network architectures.

One of the earliest and most important convolutional architectures, U-Net [22], was developed for biomedical images and received its name from a symmetrical U-shaped encoder-decoder structure. Another essential feature of this model was the usage of residual connections and transposed convolution operators. It is still used for various applications in some state-of-the-art models as a building block or an inspiration of the pipeline like [23], [24], [25], and [26].

Other crucial convolutional architectures are the ResNet family of architectures [27] and the VGG16 deep neural networks [28]. However, the research landscape noticeably changed after the release of the Visual Transformer (ViT) [18] architecture. After that, many strong transformer models were created [19], [20], [21].

We utilize one of these architectures, SegFormer-B5 [19], to extract semantic information from images. It has an encoder-decoder structure with a hierarchical transformer in the encoder and the multilayer perceptron (MLP) decoder. It is one of the most popular models for semantic segmentation. We chose it due to its high efficiency on various benchmark datasets and the availability of high-quality code and checkpoints (trained on large datasets with many outdoor images). The B5 version was chosen as it has the strongest metrics from the SegFormer family on benchmarks. We use a pre-trained model considering the complexity of collecting a dense ground truth for the dataset.

B. DEPTH ESTIMATION

Monocular depth estimation is another widespread computer vision problem [23], [29], [30], [31], [32], [33]. It is concerned with assigning a number to each pixel, which will indicate how far the object in this particular pixel is away from the camera. It became a vast field of research due to its usefulness in robotics, self-driving vehicles, 3D reconstruction, and augmented reality [30].

Initially, the state-of-the-art models were also fully convolutional [29]. Due to the complexity of collecting dense depth maps to train supervised models, the unsupervised and self-supervised models became the center of attention [23], [31], [34].

Considering the recent success of transformer-based architectures, we used the pre-trained state-of-the-art Dense Prediction Transformer (DPT) model [32] for depth map extraction. This model creates multiple embeddings for various parts of an input image. They are coupled with their positional embeddings and processed by a sequence of transformer blocks, which treat them similarly to tokens in natural language processing. Their outputs are being reassembled and fused to produce a final depth map. Similarly to SegFormer, it is a modern, strong architecture that achieved great success on many benchmarks.

C. INPAINTING

Another core part of our pipeline is the inpainting module. Overall, the inpainting task focuses on substituting a part of an image while keeping it natural. The task we are interested in for this paper is object removal. With it, we typically pass an image and a binary mask to the model. The model then has to switch masked pixels for parts that will look natural in this image. For example, if a car stands in front of the building and we mask this car, a model should be able to paint covered parts of the building instead of a car.

Considering the complexity of formally describing the natural plausibility of the inpainted images, the field is dominated by generative adversarial networks (GANs) [35], [36], [37], [38], [39].

We use the modern State-of-the-Art large mask inpainting (LaMa) model [39] for object removal. LaMa has a ResNet-like structure utilizing downscale and upscale blocks. However, its biggest feature is the usage of fast Fourier convolution-based residual blocks in the middle. This model shows state-of-the-art results across a range of inpainting benchmarks, making it our choice.

D. IMAGE RETRIEVAL AND VISUAL PLACE RECOGNITION

The final block that we use is based on the image retrieval problem. In the classic image retrieval task, a model is being trained to generate image embeddings in a space where the closeness of embeddings indicates original images having similarity in target features. Typically, a model has a large index database of images. When a new query image arrives, a model builds an image embedding and compares it to embeddings of database pictures. The closest ones are being retrieved from the database. In our case, we want to use an image retrieval block to generate embeddings from refined segmentation and depth maps, which will be close in a latent space if the original images were taken close to each other.

This is a less explored field. However, it still has various strong contenders based on modern architectures [6], [7], [8], [9], [40], [41], [42], [43], [44], [45], [46], [47], [48]. A lot of these works, however, focus on visual place recognition from raw RGB images and are mainly concerned with improving scores on existing ‘‘frozen’’ benchmarks or improving the training speed of the model [49], [50], [51], [52].

To a certain extent, advancements in navigation and visual place recognition have been achieved in simulated game environments [53], [54], which circumvent the issues associated with datasets. However, these environments differ substantially from real-life scenarios. These results inspired our idea of using synthetic data in one of the refining strategies proposed further in Section III-A.

Like the mentioned papers, we also utilize image retrieval as part of our solution. For the retrieval part, we chose to use the NetVLAD [6] and the CosPlace [9] models. NetVLAD was chosen as it is a time-tested and strong architecture. It relies on the NetVLAD layer inspired by the Vector of Locally Aggregated Descriptors (VLAD) image representation. We also used CosPlace because it is a modern and efficient image retrieval architecture that avoids the classic costly contrastive learning approach by casting training as a classification and achieves state-of-the-art results on various datasets.

III. OUR PIPELINE

The pipeline of the augmented reality application which might be completed with our solution is presented in Fig.1. Firstly, a user makes an image (query) with their mobile device [55]. Then, this image is processed by a neural network that extracts refined depth and segmentation maps (depth and segmentation maps without any dynamic objects: people and vehicles). An application has access to an index database of images with known GPS coordinates. They were preprocessed with the same neural network beforehand to reduce computational time during inference. Another neural network takes refined depth and semantic maps from the query image and retrieves an element of an index database which is supposed to be in the closest location to a query. After that, we output a location of the predicted index database element. We also extract some additional information about the GPS location (for example, interesting facts about landmarks) and provide it to the user with their location.

In this section, we present the details of the modules that extract refined depth and segmentation maps and retrieve the closest image. To achieve this goal, we developed two novel pipelines shown in Fig. 2. Their difference is in the extraction of the refined depth and semantic features. In the first pipeline (strategy 1, inpainting-based refining), we preprocess an image with the Big-LaMa model [39] to inpaint all the dynamic classes, such as vehicles or people. After that, we extract the final depth and segmentation maps with pre-trained DPT [32] and SegFormer [19] networks. In the second pipeline (strategy 2, synthetic refining), we utilize available

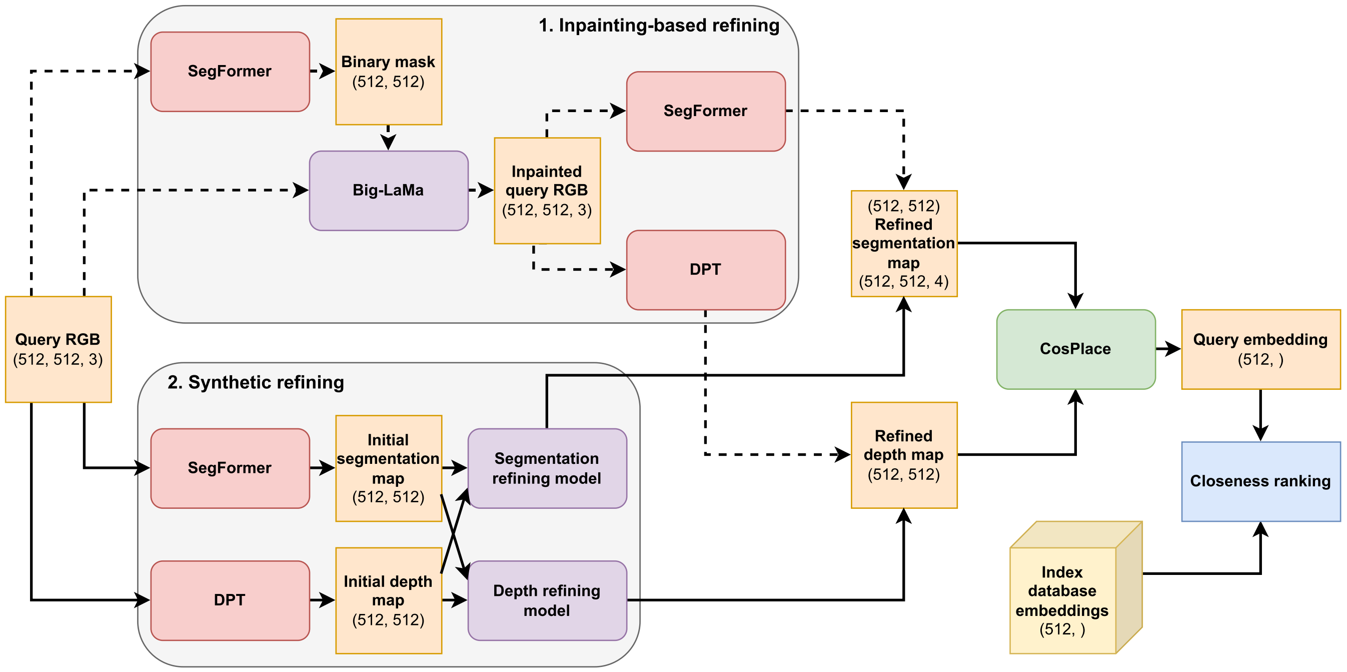

1. Inpainting-based refining
SegFormer
Binary mask (512, 512)
Big-LaMa
Inpainted query RGB (512, 512, 3)
SegFormer
DPT
Query RGB (512, 512, 3)
2. Synthetic refining
SegFormer
Initial segmentation map (512, 512)
Segmentation refining model
DPT
Initial depth map (512, 512)
Depth refining model
(512, 512) Refined segmentation map (512, 512, 4)
Refined depth map (512, 512)
CosPlace
Query embedding (512,)
Closeness ranking
Index database embeddings (512,)

FIGURE 2. The general pipeline. It can use two refining strategies based on the availability of the synthetic ground truth. The inpainting-based refinement is used when there is no ground truth available. In this module, the SegFormer generates masks for dynamic classes. Then, the Big-LaMa uses these masks and the original RGB to inpaint a query. This is followed by the semantic and depth maps extraction by the SegFormer and the DPT, respectively. The second refining strategy uses synthetic ground truth to remove dynamic objects. Here, we train two separate U-Net-inspired models. Each has the architecture presented in the Fig. 3. Firstly, we extract the initial segmentation and depth maps with the SegFormer and the DPT. This is followed by refining them with a respective refinement model. After achieving the refined maps, they are processed by the CosPlace, which generates embedding for the query. The final step is to choose a similar embedding from the index database. On outputs from each model, you can find their sizes. Note that Refined segmentation map has two shapes depending on the refining used: top size (512, 512) is obtained from the inpainting-based refining; bottom size (512, 512, 4) refers to the synthetic refining. This difference happens because the pre-trained SegFormer outputs 150 classes, which are being put into a single image with the class label in each pixel due to many output classes. Segmentation refining model outputs only four classes: sky, buildings, terrains, and trees. Therefore, here, for each class, a map of probabilities is being used, resulting in a third dimension of size 4.

static synthetic ground truth to refine depth and segmentation maps obtained without direct inpainting. The synthetic refining strategy uses supervised learning to inpaint depth and semantic maps and, therefore, works more efficiently and is preferable. However, it can only be used when synthetic depth and segmentation maps are available for all the required locations. As this might be rare, we also proposed the direct inpainting-based refining strategy.

A. REFINING FEATURES VIA SYNTHETIC DATA

Image inpainting and object removal are very challenging tasks, especially when a big part of the picture requires inpainting. This is precisely the setup that relatively often appears in images for visual place recognition due to a large number of people or vehicles (including huge vehicles) in many public areas. To make a model less noisy, we would like to utilize a supervised learning approach instead of direct adversarial learning, which might lead to noise and hallucinations of generative inpainting models. However, the ground truth for such images is typically not available.

Inspired by the success of visual place recognition in some simulated scenarios, we decided to use synthetic data as the ground truth. When a 3D map of an area of interest is available, it is relatively easy to scrape semantic and depth maps from any point on it. Therefore, by knowing in which direction the camera was looking and where the photo was taken, it is possible to obtain synthetic ground truth maps from the 3D map of an area of interest. Such 3D render will have only static classes such as buildings or roads. This will allow the training of a model in a supervised manner, taking depth and semantic maps from the original image and using them to obtain static, refined ones. Such an approach will lead to the efficient supervised inpainting of dynamic objects.

That is precisely what we do in the synthetic refining strategy (see Fig.2 (refining strategy 2)). Note, we apply our refining model to initial depth and semantic features and not to the original RGB image as it should be easier to inpaint them than the whole image with all fine details and colors.

For this approach, we used the HoliCity dataset [56] as it has synthetic depth and semantic maps scraped from the 3D map of London for every RGB. These maps are not identical to the actual data in terms of geometry. However, they only feature a few classes we want to obtain during preprocessing and are free of dynamic types. The classes that are present in the original semantic segmentation maps are the following: sky, buildings, roads, terrains, and trees.

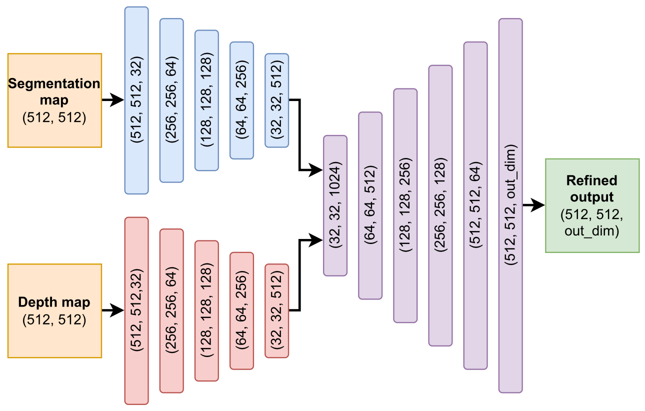


FIGURE 3. The refining model architecture. This is a U-Net-inspired model. It has similar blocks. However, instead of one encoder, we have two: one for depth and another for segmentation maps. Each of these encoders has dimensions twice smaller than the dimensions of the symmetric decoder layer. That is done to preserve the residual connections by concatenating the outputs of both encoders before passing them to the common decoder. The refined output is either refined depth or refined segmentation. We train one model for each of them to use in the synthetic refining strategy presented in Fig.2. On the network layers, one can see the sizes of outputs from this layer. The *out_dim* equals 1 for the depth map (grayscale) and 4 for the semantic map (4 segmentation classes).

Due to the inconsistency of road and terrain markup and the general similarity of the two categories, we merge these two classes to improve the robustness of the model. We utilize this synthetic data to build two refinement models: one should generate refined depth maps, while the other will provide refined semantic maps.

Firstly, we extract the initial depth and segmentation maps from the original RGB (x) by using pre-trained DPT $\mathbf{D}(.)$ [32] and SegFormer-B5 $\mathbf{S}(.)$ [19] (pre-trained on the ADE20K dataset [57], [58]) respectively:

$$d = \mathbf{D}(x) \tag{1}$$

$$s = \mathbf{S}(x) \tag{2}$$

Then, we apply refining models which are supposed to remove any dynamic objects from depth and semantic maps. However, segmentation maps alone could often be noisy while depth maps alone do not carry valuable information about distinction between different objects and can mix multiple different classes together if their instances happen to be similar distance from the camera and next to each other. This becomes especially problematic during refinement if they are refined completely separately. For example, any noise on a border of a building and a terrain in a segmentation map can propagate and get worse during removal of a vehicle object which stands on this terrain in front of this building. However, depth information can help to clarify a proper border between a building and a terrain because a wall of a building at any point typically has a similar distance to the camera. Adding depth information not only helps to improve refinement, but also to smooth any noisy edges from the initial segmentation procedure. On the other hand, with depth maps alone a model can have issues finding a difference between a dynamic and a static object if they are close to each other and similar distance away from the camera. A failure like this will obviously damage model's capability to properly remove dynamic objects.

Therefore, we utilize the refinement architecture which relies on both depth and semantic initial features to reconstruct static maps. One can see its structure in Fig. 3. It is the U-Net-inspired architecture. However, instead of a single encoder, we use two encoders of the same structure, which output twice less channels than a symmetric single decoder. The first encoder processes only the segmentation map, while the second is used only for the depth map.

We create embeddings for semantic and depth features separately because we want to encourage a model to properly extract and use information from both of them. Each of these encoders can focus on its specific modality minimizing chances of falling into a local optimum where a model fails to extract any useful information from one of the modalities.

When such well-formed separate embeddings are already created, they are mixed in the decoder to ensure that it learns to utilize all of the knowledge for future refinement. It is important to mix them as both depth and segmentation carry unique geometric information about a scene.

The first block of each encoder consist of the double convolution layer made of two convolutional layers with a kernel size of 3. Each of them is followed by the batch normalization and the ReLU activation function. The first convolution of the first layer maps the amount of channels to 32, while the second preserves this amount. Consecutive encoder blocks consist of poolings with MaxPool followed by the same double convolution layer. Like in the Unet, each consecutive encoder block decreases image size twice during MaxPool and increases the number of channels two times during the first convolution in the double convolution layer. After being processed by respective encoders, we receive embeddings of size (32, 32, 512) of a segmentation map (from the segmentation encoder) and of a depth map (from the depth encoder).

Then, we mix these features by concatenating them in channel dimension into a single embedding of size (32, 32, 1024). This combined embedding with both depth and semantic information is then processed by the decoder. This is represented by the first decoder block. Each consecutive block of the decoder, apart from the final one, has the same structure. Firstly, it applies the transposed 2D convolution to the input, reducing twice the number of channels and upsampling an image two times. Then, it takes outputs from the symmetric encoder blocks and concatenates them together similarly to the final encoder outputs. This residual embedding is then being concatenated in channel dimension with the upsampled decoder data, forming a new embedding. Finally, this new embedding is being passed into the double convolution layer, which reduces the number of channels twice once again.

For example, the third decoder block in Fig. 3 takes as the main input a tensor of shape (64, 64, 512) from the second decoder block and two residual tensors of shape (128, 128, 128) which are the outputs of the third blocks of two encoders. Both residual inputs are being concatenated into the tensor of shape (128, 128, 256). At the same time, the main input is being upsampled with transposed convolution from size (64, 64, 512) into the (128, 128, 256). Now, both the concatenated residual tensor and the main tensor have the same shape. Together they are being concatenated into a single embedding of size (128, 128, 512). This embedding is being passed into the double convolution resulting in a shape (128, 128, 256), which becomes the output of this block. The final decoder layer only applies a single convolution with kernel size 1 and maps 64 received channels into the required output amount.

Such architecture has separate encoders, but the same decoder which helps the neural network to learn features from both initial depth and segmentation maps properly without one of them dominating the other in the encoder. However, both embeddings are further mixed in the decoder to ensure that it is able to utilize all of the available unique geometric information.

We use two of such models: one to output depth and another to output segmentation maps. Notice that both of these networks take non-refined depth and semantic maps as an input, but are being trained separately: first to output a refined depth map and second to output refined segmentation. The network that outputs depth has an output channel dimension of one by the nature of depth maps. The segmentation network has 4 output channels representing 4 semantic classes (sky, building, terrain, and trees).

$\mathbf{F}_d$ refinement network outputs a refined depth map (3), while $\mathbf{F}_s$ results in a segmentation map (4). The overall results are obtained in the following way:

$$d_r = \mathbf{F}_d(d, s), \tag{3}$$
$$s_r = \mathbf{F}_s(d, s), \tag{4}$$

where d_r and s_r are refined depth and semantic maps.

The models are being trained independently in a supervised fashion with synthetic ground truth. We used the masked L1 loss (masking and ignoring the depth values that are too large in the ground truth depth maps due to the common issue of noisiness of depth estimation at large distances) for depth estimation and the Cross-Entropy loss for semantic segmentation. In both cases, we utilize the Adam optimizer with a starting learning rate of 10^{-3} coupled with a scheduler, which reduces the learning rate ten times if there are no improvements on the validation dataset for three epochs. Additionally, we use the early stopping to avoid any overfitting.

B. REFINING VIA INPANTING

High-quality synthetic ground truth, especially for a large area, is often unavailable for supervised training. Therefore, this section introduces its alternative, used in the pipeline at Fig.2 (refining strategy 1). The goal of the refinement described in Section III-A is to remove the dynamic objects (people and vehicles) from the depth and segmentation maps. However, where no ground truth is available, we propose a framework that utilizes the RGB inpainting model instead.

Inpainting consists of substituting parts of the image with different content while keeping the image look natural. Object removal is one of the most popular forms of inpainting, where the neural network replaces an object with its background. If we can remove dynamic objects from the original RGB image and pass them to the DPT and the SegFormer, we should also obtain depth and segmentation maps that do not contain these objects.

Therefore, one can see our inpainting procedure in Fig. 2 (refining strategy 1). Firstly, we apply the SegFormer-B5 model to the original RGB image to obtain the segmentation maps. These maps contain 150 known classes, some are static, and others are dynamic. We get binary masks of dynamic objects by marking every pixel having any dynamic object class (5). After that, we pass these binary masks and the original RGB image into the pre-trained Big-LaMa [39] inpainting model $\mathbf{L}(.,.)$. Therefore, the Big-LaMa produces images that are supposed to have only static classes (6).

$$\text{mask} = \mathbb{I}_{\mathbf{S}(x)\in N}, \tag{5}$$
$$y = \mathbf{L}(x, \text{mask}), \tag{6}$$

where x is the original RGB image, and N is the set of classes of dynamic objects among SegFormer output classes.

The resultant inpainted image y is then passed into the DPT and the SegFormer models to obtain final depth and segmentation maps without dynamic objects (7), (8):

$$d_r = \mathbf{D}(y) \tag{7}$$
$$s_r = \mathbf{S}(y) \tag{8}$$

Such an approach is promising and compares well with the synthetic data refinement described in Section III-A. Some comparisons of the segmentation maps obtained by the synthetic and inpainting-based refining are available in Fig. 4. One can notice on these images that it is simpler to inpaint semantic maps than original RGB images. Some dynamic objects on the original RGB do look blurry or noisy on the inpainted RGBs. However, this becomes significantly less pronounced on the segmentation maps obtained from these inpainted RGBs. Synthetic and inpainting-based refining produce different colors of semantic classes in Fig.4. This happens because SegFormer has way more static output classes than the synthetic refinement model (which only leaves buildings, terrains, sky, and trees). The main goal of this image is to show that both strategies produce maps with removed dynamic classes. Synthetic-based refining is still overall more robust as it uses additional high-quality data for supervised training of the refinement models.

Another positive of an inpainting-based approach is that it only relies on the pre-trained networks, and there is no need to train new large models from scratch, which would require a lot of time and computational resources.

C. IMAGE RETRIEVAL FOR VISUAL PLACE RECOGNITION

After obtaining the final depth and segmentation maps by refining them with synthetic data or direct inpainting, they are used for image retrieval. This block is the final in both pipelines at Fig.2. Image retrieval involves building a latent space $L = \mathbb{R}^n$ for depth and segmentation inputs d, s. This latent space should have the following properties:

$$\forall s_q, d_q \in D_q : s_j, d_j = \underset{s_i, d_i \in D_{index}}{\operatorname{argmin}} \mathbf{L2}(\mathbf{f}(s_q, d_q), \mathbf{f}(s_i, d_i)) \Rightarrow$$
$$j = \underset{i \in index}{\operatorname{argmax}}\, \mathbf{closeness}(q, i), \tag{9}$$

where D_q and D_{index} stand for the queries and index databases, respectively. In other words, we have a set of depth and segmentation maps named index database for which we know the required information (in our case, a GPS location of the image or the name of the landmark). We also have a similarly structured set of potential queries. It has depth and semantic maps for images for which we do not know the required geographical information. Therefore, our goal is to find a mapping $\mathbf{f} : D \rightarrow L$ to the latent space such that the closeness of vectors in the latent space means the similarity of images in the real world regarding the geographical information we are interested in. For example, the closest vector from an index database to a query was obtained from an image taken geographically close to the place in a query image. This closeness relation is represented in (9). That way, we can take known information about the closest index database image and predict it for a query image. For example, we can find index database images from the same location as a query and give their location as an answer.

In this work, we explore two architectures for this task. The first is the classic NetVLAD architecture [6]. It uses the end-to-end convolutional neural network with a particular generalized VLAD layer inspired by the Vector of Locally Aggregated Descriptors image representation. The second model is the modern state-of-the-art architecture named CosPlace [9]. It takes a different approach during training and casts the training as a classification problem. This approach helps to reduce the computational complexity and leads to high results on the visual place recognition task. For the CosPlace, we tested backbones based on the ResNet-18, ResNet-50, ResNet-101, ResNet-152, and VGG16 architectures [27]. The NetVLAD model, however, has VGG16 [28] backbone.

All architectures were trained for 50 epochs. We used triplet margin loss with a margin of $\sqrt{0.1}$. As an optimizer, we used the stochastic gradient descent with momentum with a starting learning rate of 10^{-4} with a StepLR scheduler, which decreased the learning rate two times every five epochs. For the CosPlace family of models, we used the Large Margin Cosine Loss [59] for every group partitioned, as explained in detail in the original CosPlace paper [9].

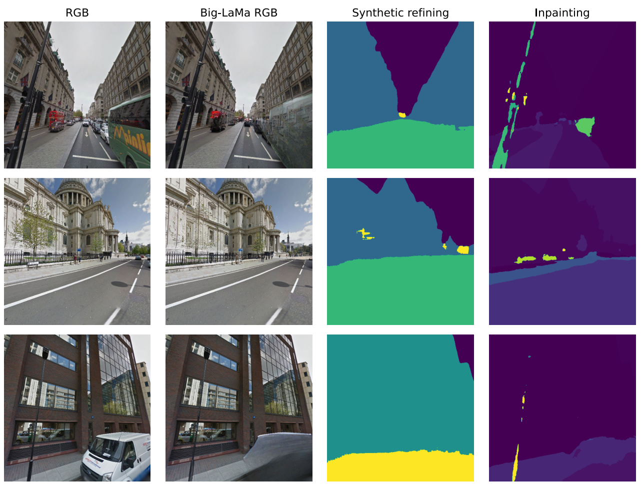


FIGURE 4. The segmentation maps were compared with different refining strategies and relevant RGB images. The first column displays the original images, while the second column shows the inpainted photos used in the pipeline with inpainting-based refining. The fourth column depicts the segmentation maps extracted from the inpainted images employed in the inpainting-based refining. The third column shows the same refined segmentation maps obtained using synthetic refining instead of direct inpainting. Inpainting is not perfect. However, the segmentation maps are still quite good, as even some areas that appear blurry on RGBs after inpainting are still properly segmented (notice a blurry bus in the first row of column 2 and its absence in column 4). The inpainting does not introduce much noise to the pipeline because we use inpainted segmentation maps, which are significantly less noisy than inpainted RGB images. However, it is also clear that the quality of synthetic refinement is higher. That is expected as the additional artificial ground truth was used here to improve the models. Note a color difference between segmentation maps in columns 3 and 4. That happens because SegFormer (which is used to obtain segmentation maps in the inpainting scenario, column 4) is trained to predict significantly more classes than the refining model used in the synthetic refining (column 3). Due to the existence of static synthetic ground truth, synthetic refining models are trained to predict only classes: sky, building, terrain, and tree, while the SegFormer outputs 150 classes. The main focus of this image is to show how well the segmentation maps represent RGB images and how dynamic objects are being inpainted on semantic maps in both refinement scenarios.

A model was trained using the Adam optimizer with 10^{-5} learning rate.

IV. EXPERIMENT SETUP

A. HOLICITY DATASET

The first dataset that we explore is the HoliCity dataset [56]. It contains images taken outdoors in the dense area of over 20 km^2 in London. More importantly, it contains a lot of additional information, such as GPS coordinates, synthetic semantic segmentation, and depth maps. Synthetic data for this dataset was scraped from the computer-aided design (CAD) model of this area and did not include people or vehicles.

The total size of the dataset is 47528 images representing 5941 different locations (8 images per location taken with horizontal rotations of the camera by 45°). Forty-five thousand thirty-two images (5629 locations) were taken as a training dataset, and 2496 images (312 locations) were used for testing.

The only objects present on synthetic semantic and depth maps are the sky (29.38% of pixels for the class), buildings

(44.88% of pixels), roads (0.43% of pixels), terrains (19.65% of pixels), and trees (5.66% of pixels). Therefore, no dynamic objects were present in them. Consequently, the synthetic semantic segmentation maps consist of the same five classes: sky, building, road, terrain, and tree. Sky and buildings are self-explanatory. Roads are supposed to represent the area for driving vehicles, while terrains represent any walkable section. Trees outline only trees themselves. Other green spaces, like grass, are not included in this class. During our experiments, we merged roads and terrains on segmentation maps into a single class due to their semantic similarity and due to the noticed inconsistencies in the ground truth markup between these two classes (we saw various examples of driving roads being classified as terrains in ground truth semantic maps). We report the results for this dataset on the validation split.

This dataset is essential as it allows us to test our synthetic refinement pipeline and to check the difference in performing visual place recognition while having dense and noiseless ground truth depth and segmentation maps versus making the alternative inpainting approach.

To indicate closeness, we use GPS coordinates. Considering the relatively small area and dense collection of GPS locations, we thought two places would be close if there was at most a 25-meter difference between them.

B. GOOGLE LANDMARKS V2 DATASET

As the second dataset, we chose the Google Landmarks V2 dataset [60]. This is a large dataset of images of worldwide landmarks collected from Wikimedia resources. It does not have synthetic data. However, each image has its landmark and a link to the Wikimedia knowledge graph.

Considering the inequality of landmarks among countries in the dataset, we decided to work with a single state from the dataset, a popular tourist destination. We chose to work with landmarks from France as they seem to fit this criterion and are well-represented in the dataset. To imitate the situation with the lack of training data (as not for every country, there is a wide variety of pictures available), we only took 1/4 of the train images available for France. The total size of the dataset was 79004 images representing 10157 unique landmarks. Out of this list, there were 49370 training images (4770 unique landmarks) and 29634 test images (5387 unique landmarks). From this, the test set consisted of query-index pairs, representing completely new landmarks not present in the train set, making this dataset more challenging. There were about 10.35 images per landmark in the training dataset and 5.5 in the testing.

In this dataset, we used landmarks as a binary indicator of closeness. Namely, images are close if they are of the same landmark and distant otherwise.

C. METRICS

For this experiment, we used traditional Recall@N metrics. They measure the percentage of queries with at least one close element among the model's top N predictions. It can be written as follows:

$$\textbf{Recall@N} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{I}_{\min(\mathbf{dist}_{i,j},\ldots,\mathbf{dist}_{i,N})<t} \tag{10}$$

where $\mathbf{dist}_{i,j}$ stands for the distance between two places: first is a place where a query image q_i was taken, and second is a place where an image from an index database was taken, which model predicted to be j-th closest to a query q_i (for example, $\mathbf{dist}_{i,1}$ is the distance between a query q_i and an index database element that model thinks is the closest to a q_i); t is a distance threshold; M is a number of queries.

For the HoliCity dataset, the distance function is the distance in meters between two points from which the original images were taken, and the threshold is 25 m.

For Google Landmarks V2, the distance is binary: one if the original images represent different landmarks and 0 if they are from the same landmark. The threshold is 1. In other words, the indicator function will only count cases where landmarks of a predicted image and a query are the same.

We utilize only Recall@1, Recall@5 and Recall@10 following the best practices from evaluation pipelines in localization literature [6], [7], [8], [9], [46], [47], [48], [49], [50], [51], [52]. These metrics provide the most meaningful information for the image retrieval-based geolocation. Recall@1 shows the most interesting data and addresses how well a model can perform on the first try. Recall@5 and Recall@10 show model's accuracy when a relatively small amount of mistakes is allowed.

D. HARDWARE

Our models were trained with a single NVIDIA A100 with 80GB VRAM. For reference, here are the repositories for the state-of-the-art models used in our pipeline: 1) SegFormer B5

TABLE 1. Results on the HoliCity dataset. The refinement column indicates the type of refinement used: synthetic indicates the type described in Section III-A; inpainting indicates the type described in Section III-B. In the middle are the results of the ablation study, where only depth maps or semantic maps are used in the synthetic refinement scenario. The CosPlace with the ResNet-50 works stronger in both inpainting scenarios. The only architecture that has slightly better performance is the ResNet-152-based CosPlace. However, it is significantly larger and takes more time to train. VGG16-based architectures (NetVLAD and VGG16 CosPlace) perform worse than the ResNet-based CosPlace. All models show stronger results when synthetic data is used. The models using only depth or semantic features with synthetic refinement are stronger than inpainting models. However, they cannot reach the level set by CosPlace with ResNet-50, which utilizes both depth and segmentation maps.

Model	Backbone	Refinement	Recall@1	Recall@5	Recall@10
NetVLAD	VGG16	synthetic	46.6%	68.1%	75.9%
CosPlace	ResNet-18	synthetic	57.4%	74.6%	83.2%
CosPlace	ResNet-50	synthetic	60.6%	79.3%	85.6%
CosPlace	ResNet-101	synthetic	57.4%	78.4%	84.2%
CosPlace	ResNet-152	synthetic	60.8%	78.9%	85.3%
CosPlace	VGG16	synthetic	52.5%	73.0%	79.6%
CosPlace depth	ResNet-50	synthetic	57.4%	76.2%	82.4%
CosPlace segmentation	ResNet-50	synthetic	58.3%	76.2%	85.5%
NetVLAD	VGG16	inpainting	8.3%	23.8%	33.7%
CosPlace	ResNet-18	inpainting	47.3%	67.2%	74.1%
CosPlace	ResNet-50	inpainting	51.1%	73.4%	81.0%

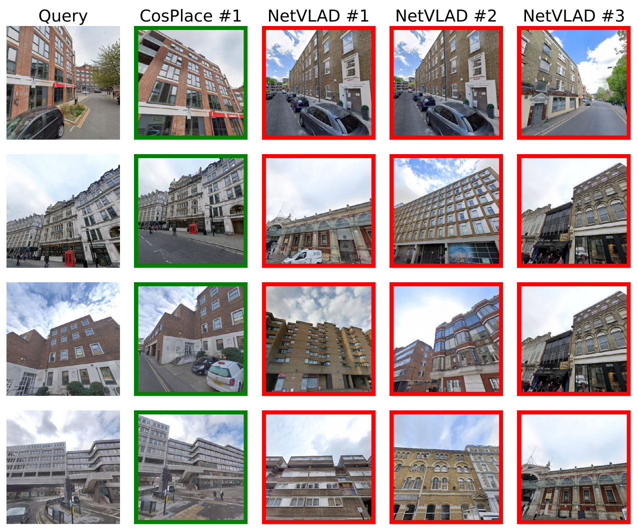


FIGURE 5. Queries from the HoliCity where the CosPlace (ResNet-50) shows significantly stronger results than the NetVLAD. In the first column one can see RGBs of queries. Note: we show RGBs to achieve more insightful visualizations; during training and inference both models were working with depth and semantic maps. The second column shows top-1 predictions of the CosPlace (they are all correct, shown in green frames). For these queries all top-10 predictions of the NetVLAD were wrong. For compactness we show only top-3 predictions in the final three columns. One can see that CosPlace's predictions seem to be more robust to differences in orientation of the camera between queries and index database elements.

(checkpoint: Segformer MIT-B5 512 × 512, ADE20K); 2) DPT (checkpoint: dpt-hybrid-midas-501f0c75.pt); 3) LaMa (checkpoint: big-lama.zip); 4) NetVLAD (trained ourself); 5) CosPlace (trained ourself).

V. RESULTS

We started with the HoliCity dataset due to the availability of both refinement techniques. Therefore, we can see the potential differences between the two techniques. We tested the NetVLAD with the VGG16 backbone and the CosPlace with the ResNet-18, ResNet-50, ResNet-101, ResNet-152, and VGG16 backbones. The overall results are in the Table 1. During the synthetic refinement experiment, NetVLAD had decent metrics but is still being outperformed by the CosPlace. Almost all CosPlace variants perform similarly, having almost identical recall values for all tested N. An obvious outlier is the VGG16 backbone. It still outperforms the NetVLAD, but the ResNet-based CosPlace models have better metrics. The ResNet-50 still claims overall victory as it performs better for the lower N values, which are the main point of interest. The only model that slightly outperforms it is the CosPlace with the ResNet-152 backbone. However, the difference is slight, while the ResNet-50 is noticeably faster to train and less affected by potential overfitting. It seems like it could be the issue for larger models based on the ResNet-101 lower performance. The CosPlace winning against the NetVLAD is an expected behavior as the CosPlace is a modern state-of-the-art approach, and it is supposed to outperform the NetVLAD, which came out a couple of years before the CosPlace. Also, the ResNet-50 has more parameters and, therefore, more degrees of freedom to learn complicated features than the ResNet-18. However, the ResNet-50, especially with the fast training proposed in the CosPlace paper, is still not too heavy. The ResNet-101 might have worse results due to being more affected by the overfitting. Due to the similarity in performance, we prefer lighter models, which might run inference better on simpler hardware. Therefore, we focus on the ResNet-18 and the ResNet-50 backbones in other experiments.

TABLE 2. Results on the Google Landmarks V2 dataset. The inpainting refinement was used for all experiments described in Section III-B. The CosPlace with ResNet-50 backbone still tops the ranking and wins by a large margin.

Model	Backbone	Recall@1	Recall@5	Recall@10
NetVLAD	VGG16	1.1%	3.4%	5.0%
CosPlace	ResNet-18	12.4%	22.4%	27.4%
CosPlace	ResNet-50	17.9%	29.9%	35.5%

In Fig. 5 one can see some of the queries where the CosPlace significantly outperformed the NetVLAD. More specifically, for all of these queries top-1 predictions of the CosPlace were correct every time, but all top-10 predictions of the NetVLAD were wrong (only top-3 shown in Fig. 5 to avoid overloading it). One can see from the figure that the CosPlace is being noticeably more robust to camera orientation. For all presented queries NetVLAD's guesses were in a similar orientation to the query. In the first row both the query and NetVLAD's predictions have the camera looking straight while the CosPlace predicted an image with the camera looking up. The opposite situation can be seen in the other rows. The CosPlace retrieves images where the camera looks straight, while the NetVLAD prefers upwards camera angles like in given queries. Also, the NetVLAD relies too much on general features and positions of objects rather than on some specific traits of presented buildings. For example, in the first row one can see that the NetVLAD is searching for the image where a building with a roughly similar shape is on the side of an image like in the provided query. On the other hand, the CosPlace relies more on details of the building and retrieves an image where the same building is in the center of the picture. This might be happening due to the way the model is trained. The approach proposed in [9] relies on building CosPlace groups from the data. Images from different classes in the same group comply with minimal distance or angle thresholds. This approach can help CosPlace to differentiate images with similar angles, but different content.

During the inpainting experiment, the situation changes significantly. The NetVLAD receives a significant drop in all metrics and cannot perform well in this setting. On the other hand, both CosPlace models handle the transition rather well. They still experience a performance drop. However, it is expected while moving from the supervised refining scenario with dense ground truth maps. Nevertheless, the decrease is not as dramatic, especially for the ResNet-50 backbone.

Additionally, we performed a series of experiments to test various branches of our pipeline. In particular, we tried to use only depth maps or segmentation maps instead of both in the synthetic refinement scenario. The results are indicated in Table 1 with "CosPlace depth" and "CosPlace segmentation" titles, respectively. Both of them perform similarly and manage to show high metrics. They even outperform CosPlace with ResNet-50, which utilizes a direct inpainting approach. Depth and semantic maps are expected to provide models with important geometrical features that sometimes share some common information. For example, by looking for dramatic jumps in depth between neighboring pixels, one might estimate the boundaries of objects. They also utilize synthetic data in the refining, improving their quality. However, CosPlace, with ResNet-50 and synthetic refinement, which uses semantic and depth maps, performs noticeably better. This happens because both modalities still provide the model with unique geometrical information. Also, having them share features during the synthetic refinement allows for more efficient inpainting.

Knowing previous experiments, we tested all three models on the significantly more complicated and diverse Google Landmarks V2 dataset. The results are presented in the Table 2. This dataset shows a similar picture to the one in the inpainting-based refining of the HoliCity dataset. In particular, NetVLAD performs way worse than its competitors in this scenario. The CosPlace with the ResNet-50 remains the strongest and confidently claims victory over the same model with the ResNet-18 as a backbone. It is again expected due to a large number of parameters and utilization of the state-of-the-art approach to visual place recognition. It is worth mentioning that the metrics of all models on the Google Landmarks V2 dataset are lower than on the HoliCity. This is because this dataset has a much more complex structure and a lot of landmarks in the test set that were not present in the training set.

VI. DISCUSSION

From the experiments on the HoliCity dataset, it is clear that the synthetic refinement scenario gives stronger performance than direct inpainting. This is expected because, with synthetic refinement, we utilize our Unet-inspired models, which are carefully trained to utilize both depth and semantic features. Additionally, we train them in the supervised scenario using noiseless synthetic ground truth data, which does not include any dynamic objects in a scene. On the other hand, direct inpainting of a large number of images will typically introduce some noise into some of them. This noise is especially likely if there are many large dynamic objects in a scene and the inpainting model is required to recover a large part of the picture. Even though such noise is significantly less problematic on depth and semantic maps than on original RGB images, the SegFormer and the DPT models may still have issues dealing with it.

Among the tested models, CosPlace shows stronger performance than NetVLAD. This is expected, as a similar trend was illustrated in the classic RGB-only scenario in the original paper [9]. The Recall@1 difference between the two models on multiple datasets was at least 10%+, which is consistent with our experiments. Overall, VGG16-based models performed worse than ResNet-based networks. However, the VGG16-based NetVLAD is still noticeably far from the VGG16-based CosPlace. This consistency with the RGB-based geolocation performed in the original paper [9] is maintained because both RGB and semantic/depth maps provide models with similar geometrical features. The core difference is the amount of additional dynamic details provided in the RGB.

The ResNet-50-based CosPlace performed the best overall, with similar quality to the larger models. This is because, at this point on HoliCity, the performance was already strong enough, and a larger model was only occasionally providing marginal improvements. Additionally, these larger models start overfitting sooner. The best quality ResNet-50-based CosPlace was able to provide in a scenario with depth and segmentation maps refined by our Unet-based models.

Additionally, ablation experiments clearly show the importance of using both depth and semantic features. While models that rely only on one of them are reasonably effective, they still fall behind the best-performing model, which utilizes both.

VII. LIMITATIONS

A. REQUIRED SYNTHETIC DATA

the strongest pipeline in our experiments relies on synthetic ground truth data. While this is not necessary for the pipeline with direct inpainting, the pipeline with Unet-based refinement models provides a significant improvement in terms of quality. It is also more robust as there is no noise that could be introduced during direct inpainting.

B. SCENES WITH SIMILAR GEOMETRIC FEATURES

some of the potential scenes a model will have to deal with might have less valuable geometric information, which will be reflected on depth or semantic maps. For example, a scene might mostly include a generic-looking building. Such a building will be easier to blend with another without having some of the visual details available, like in RGB images.

C. SINGLE VIEW LOCALIZATION

like most of the current models, we utilize a single image for geolocation. This is less effective as sometimes the views could be very different if a person turns around. In addition, some different places might look similar. To some extent, this is being mitigated by the possibility of sampling from a 3D map to generate new synthetic data representing more views. However, the shared information from views taken as a short video might lead to significantly easier identification. In this case, a model could also know about the relative positions of more objects in a scene, which is especially useful when dealing with only static objects.

VIII. CONCLUSION

In this work, we explored the possibility of performing the task of visual place recognition using geometrical information from segmentation and depth maps only. Such a solution helps to focus on the stable features of the image, such as shapes, and is not affected as much by the time of the day, time of the year, billboards, and other factors that might change over a reasonably small amount of time. This saves a lot of resources required to keep the index database up to date.

We tested two potential refining strategies for extracting high-quality depth and segmentation maps using the synthetic data and the inpainting module. The synthetic refining strategy used two U-Net-inspired refining modules that mix the initial depth and segmentation features to build the refined maps. One of these modules was trained to output segmentation maps and the other to output depth maps. The training was supervised based on the synthetic ground truth depth and segmentation maps obtained from the 3D map of the area. The inpainting-based refining did not require the ground truth or the additional training procedure. It used SegFormer's masks and the Big-LaMa inpainting model to remove dynamic objects from the image. Then, the final maps were obtained from the inpainted image.

Firstly, we analyzed both strategies on the HoliCity dataset, which has all the required data for both refining options. Our findings on the HoliCity dataset show that the quality of the inpainting-based approach depends on the model. While the classic NetVLAD approach has troubles and its metrics plummet, the CosPlace algorithm performs well with inpainting for both tested backbones. With synthetic refining, the metrics are better across all of the models. Here, the NetVLAD achieves more robust results and comes closer to the CosPlace-based models. The CosPlace with the ResNet-50 as a backbone is overall the most robust model in this comparison. It does not drop its metrics significantly even in a more challenging scenario without synthetic ground truth. The ablation study on the HoliCity dataset revealed the importance of using depth and semantic maps instead of only one of them.

Further, we explore the possibility of using the same direct inpainting approach on the complex and diverse Google Landmarks V2 dataset for France. The overall trend continued as the CosPlace models were significantly stronger using inpainting, and the CosPlace with the ResNet-50 as a backbone became the winner in all tested recall metrics.

This shows the potential of such an approach. However, there is a clear room for improvement in terms of metrics. Additionally, in this work we focused on introducing the novel pipeline and trying to find the most efficient version of it using one of the few publicly available datasets which have synthetic depth and semantic maps available for localization. We compared different backbones and retrieval methods as well as various refinement techniques. However, more tests against more models is an important part of the future research. As the leading direction of potential improvements, we see the possible usage of video sequences instead of single images. From a practical point of view, it typically does not require a lot of additional work for the user to move the camera for several seconds. Therefore, such a solution will not make the user experience significantly worse. On the other hand, a video feed that is several seconds long can provide more information about a location and its surrounding area. It will allow us to use image sequences as pseudo-stereo images to improve the depth estimation and, consequently, predictions.

REFERENCES

[1] S. Schubert, P. Neubert, S. Garg, M. Milford, and T. Fischer, "Visual place recognition: A tutorial," 2023, *arXiv:2303.03281*.

[2] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," *Pattern Recognit.*, vol. 113, May 2021, Art. no. 107760.

[3] I. Grechikhin and A. V. Savchenko, "User modeling on mobile device based on facial clustering and object detection in photos and videos," in *Proc. 9th Iberian Conf. Pattern Recognit. Image Anal. (IbPRIA)*. Springer, 2019, pp. 429–440.

[4] H. Zhang, X. Chen, H. Jing, Y. Zheng, Y. Wu, and C. Jin, "ETR: An efficient transformer for re-ranking in visual place recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 5654–5663.

[5] A. V. Savchenko, K. V. Demochkin, and I. S. Grechikhin, "Preference prediction based on a photo gallery analysis with scene recognition and object detection," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108248.

[6] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5297–5307.

[7] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4470–4479.

[8] S. Hausler, S. Garg, M. Xu, M. Milford, and T. Fischer, "Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14136–14147.

[9] G. Berton, C. Masone, and B. Caputo, "Rethinking visual geo-localization for large-scale applications," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4878–4888.

[10] F. Lu, L. Zhang, S. Dong, B. Chen, and C. Yuan, "AANet: Aggregation and alignment network with semi-hard positive sample mining for hierarchical place recognition," 2023, *arXiv:2310.05184*.

[11] T. Krajník, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4537–4542.

[12] V. V. Savchenko and A. V. Savchenko, "Criterion of significance level for selection of order of spectral estimation of entropy maximum," *Radioelectronics Commun. Syst.*, vol. 62, no. 5, pp. 223–231, May 2019.

[13] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 3–15, 2017.

[14] A. V. Savchenko and Y. I. Khokhlova, "About neural-network algorithms application in viseme classification problem with face video in audiovisual speech recognition systems," *Opt. Memory Neural Netw.*, vol. 23, no. 1, pp. 34–42, Jan. 2014.

[15] A. Korinevskaya and I. Makarov, "Fast depth map super-resolution using deep neural network," in *Proc. IEEE Int. Symp. Mixed Augmented Reality Adjunct (ISMAR-Adjunct)*, Oct. 2018, pp. 117–122.

[16] I. Makarov, D. Maslov, O. Gerasimova, V. Aliev, A. Korinevskaya, U. Sharma, and H. Wang, "On reproducing semi-dense depth map reconstruction using deep convolutional neural networks with perceptual loss," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 1080–1084.

[17] I. Makarov, M. Bakhanova, S. Nikolenko, and O. Gerasimova, "Self-supervised recurrent depth estimation with attention mechanisms," *PeerJ Comput. Sci.*, vol. 8, p. e865, Jan. 2022.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16 $\times$ 16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–21.

[19] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Advances in Neural Information Processing Systems*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, 2021, pp. 12077–12090.

[20] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022, *arXiv:2112.01527*.

[21] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, "One-Former: One transformer to rule universal image segmentation," 2022, *arXiv:2211.06220*.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," 2015, *arXiv:1505.04597*.

[23] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," 2019, *arXiv:1806.01260*.

[24] E. Ershov et al., "NTIRE 2022 challenge on night photography rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 1286–1299.

[25] Z. Yue, Q. Zhao, L. Zhang, and D. Meng, "Dual adversarial network: Toward real-world noise removal and noise generation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 41–58.

[26] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general U-shaped transformer for image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 17683–17693.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*.

[29] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," 2016, *arXiv:1606.00373*.

[30] I. Vasiljevic, N. Kolkin, S. Zhang, R. Luo, H. Wang, F. Z. Dai, A. F. Daniele, M. Mostajabi, S. Basart, M. R. Walter, and G. Shakhnarovich, "DIODE: A dense indoor and outdoor depth dataset," 2019, *arXiv:1908.00463*.

[31] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 8001–8008.

[32] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12159–12168.

[33] D. Maslov and I. Makarov, "Online supervised attention-based recurrent depth estimation from monocular video," *PeerJ Comput. Sci.*, vol. 6, p. e317, Nov. 2020.

[34] A. Karpov and I. Makarov, "Exploring efficiency of vision transformers for self-supervised monocular depth estimation," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2022, pp. 711–719.

[35] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-form image inpainting with gated convolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4470–4479.

[36] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5505–5514.

[37] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "EdgeConnect: Generative image inpainting with adversarial edge learning," 2019, *arXiv:1901.00212*.

[38] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with deep generative models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6882–6890.

[39] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with Fourier convolutions," 2021, *arXiv:2109.07161*.

[40] H. Wu, M. Wang, W. Zhou, Y. Hu, and H. Li, "Learning token-based representation for image retrieval," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 3, pp. 2703–2711.

[41] F. Tan, J. Yuan, and V. Ordonez, "Instance-level image retrieval using reranking transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12085–12095.

[42] P. Weinzaepfel, T. Lucas, D. Larlus, and Y. Kalantidis, "Learning super-features for image retrieval," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–19.

[43] S. Shao, K. Chen, A. Karpur, Q. Cui, A. Araujo, and B. Cao, "Global features are all you need for image retrieval and reranking," 2023, *arXiv:2308.06954*.

[44] M. Oquab et al., "DINOv2: Learning robust visual features without supervision," 2023, *arXiv:2304.07193*.

[45] B. Arcanjo, B. Ferrarini, M. Milford, K. D. McDonald-Maier, and S. Ehsan, "An efficient and scalable collection of fly-inspired voting units for visual place recognition in changing environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2527–2534, Apr. 2022.

[46] M. Leyva-Vallina, N. Strisciuglio, and N. Petkov, "Data-efficient large scale place recognition with graded similarity supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 23487–23496.

[47] S. Zhu, L. Yang, C. Chen, M. Shah, X. Shen, and H. Wang, "R^2Former: Unified retrieval and reranking transformer for place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 19370–19380.

[48] Y. Shen, S. Zhou, J. Fu, R. Wang, S. Chen, and N. Zheng, "StructVPR: Distill structural knowledge with weighting samples for visual place recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 11217–11226.

[49] R. Wang, Y. Shen, W. Zuo, S. Zhou, and N. Zheng, "TransVPR: Transformer-based place recognition with multi-level attention aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 13638–13647.

[50] G. Berton, G. Trivigno, B. Caputo, and C. Masone, "EigenPlaces: Training viewpoint robust models for visual place recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Jun. 2023, pp. 11080–11090.

[51] Q. Zhang, Z. Xu, Y. Kang, F. Hao, Z. Ren, and J. Cheng, "Distilled representation using patch-based local-to-global similarity strategy for visual place recognition," *Knowl.-Based Syst.*, vol. 280, Nov. 2023, Art. no. 111015.

[52] A. Ali-bey, B. Chaib-Draa, and P. Giguère, "MixVPR: Feature mixing for visual place recognition," 2023, *arXiv:2303.02190*.

[53] I. Makarov and P. Polyakov, "Smoothing Voronoi-based path with minimized length and visibility using composite Bezier curves," in *Proc. Int. Joint Conf. Anal. Images, Social Netw. Texts*, 2016, pp. 191–202.

[54] I. Makarov, M. Tokmakov, P. Polyakov, P. Zyuzin, M. Martynov, O. Konoplya, G. Kuznetsov, I. Guschenko-Cheverda, M. Uriev, I. Mokeev, O. Gerasimova, L. Tokmakova, and A. Kosmachev, "First-person shooter game for virtual reality headset with advanced multi-agent intelligent system," in *Proc. 24th ACM Int. Conf. Multimedia*, Oct. 2016, pp. 735–736.

[55] A. V. Savchenko, L. V. Savchenko, and I. Makarov, "Fast search of face recognition model for a mobile device based on neural architecture comparator," *IEEE Access*, vol. 11, pp. 65977–65990, 2023.

[56] Y. Zhou, J. Huang, X. Dai, S. Liu, L. Luo, Z. Chen, and Y. Ma, "HoliCity: A city-scale data platform for learning holistic 3D structures," 2021, *arXiv:2008.03286*.

[57] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5122–5130.

[58] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 302–321, Mar. 2019.

[59] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "CosFace: Large margin cosine loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5265–5274.

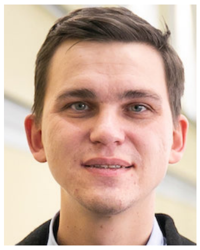
[60] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google Landmarks Dataset v2—A large-scale benchmark for instance-level recognition and retrieval," 2020, *arXiv:2004.01804*.

ILIA SEMENKOV received the M.S. degree in computer science from HSE University, Moscow, Russia, in 2021, where he is currently pursuing the Ph.D. degree.

From 2020 to 2021, he was a Research Assistant with the Vision Systems Laboratory, Institute for Information Transmission Problems (Kharkevich Institute) of the Russian Academy of Sciences (IITP RAS), Moscow; and the Laboratory for Intelligent Systems and Structural Analysis (ISSA Lab), HSE University. Since 2021, he has been a Junior Research Scientist with the Moscow's Artificial Intelligence Research Institute (AIRI). His research interests include computer vision, deep learning in neuroscience, statistical learning, and optimal transport.

ALEKSEI KARPOV received the M.S. degree in computer science from the Moscow Institute of Physics and Technology (National Research University), Moscow, Russia, in 2022.

From 2019 to 2021, he was a Deep Learning Engineer with the Huawei Russian Research Technology Institute, Moscow. He was also a Junior Research Scientist with the Artificial Intelligence Research Institute (AIRI), Moscow, from 2021 to 2022. His research interests include computer vision, depth estimation, image matching, and classification.



ANDREY V. SAVCHENKO received the B.S. degree in applied mathematics and informatics from Nizhny Novgorod State Technical University, Nizhny Novgorod, Russia, in 2006, the Ph.D. degree in mathematical modeling and computer science from State University Higher School of Economics, Moscow, Russia, in 2010, and the Doctor of Science degree in system analysis and information processing from Nizhny Novgorod State Technical University, in 2016. Since 2022, he has been with Sber AI Laboratory, where he is currently the Scientific Director. He is also a Leading Research Fellow with the Laboratory of Algorithms and Technologies for Network Analysis, HSE University, Nizhny Novgorod. He has authored or coauthored one monograph and more than 100 articles. His research interests include statistical pattern recognition, image classification, and biometrics.

ILYA MAKAROV received the Specialist degree in mathematics from Lomonosov Moscow State University, Moscow, Russia, and the Ph.D. degree in computer science from the University of Ljubljana, Ljubljana, Slovenia.

Since 2011, he has been a Lecturer with the School of Data Analysis and Artificial Intelligence, HSE University, where he was the School Deputy Head, from 2012 to 2016. He is an associate professor and a senior research fellow. He was also the Program Director of BigData Academy MADE from VK and a Researcher with the Samsung-PDMI Joint AI Center, St. Petersburg Department of V. A. Steklov Mathematical Institute of the Russian Academy of Sciences, Saint Petersburg, Russia. Currently, he is a Senior Research Fellow with the Artificial Intelligence Research Institute (AIRI), Moscow, where he leads the research in industrial AI. He became the Head of the AI Research Center and Data Science Tech Master Program in NLP, National University of Science and Technology MISIS.