G-RIPS Sendai 2025

Mitsubishi-A Group

Final Report

Authors:
Zachary Fendler[1]
Keigo Horikoshi[2]
Ikkei Sato[3]
Samuel Scheuerman[4]

Mentors:
Dr. Xiwen Jiang[+]
Mr. Akinobu Sasada[*]
Mr. Takuya Saeki[*]
Dr. Masashi Yamazaki[*]

[1] Washington State University
[2] Rikkyo University
[3] Tokyo Metropolitan University
[4] Colorado State University
[+] Academic Mentor, University of California, Irvine
[*] Industry Mentor, MITSUBISHI Electric

August 7, 2025

Contents

1 Introduction

Relocalization is a computational method that estimates a position on a map using image data. Much like a lost person in a shopping mall determines their location by comparing visible landmarks such as stores or escalators with a floor map, relocalization determines the position and orientation of a camera by analyzing visual features in the image and matching them to a known map. More formally, relocalization involves the process of estimating the six degrees of freedom (6-DoF), including 3D position and orientation, of a camera within a known map. To do this, feature extraction plays a central role: it involves identifying distinctive patterns — such as corners, edges, or textures — within the image that can be reliably matched to corresponding features in the map. This is a fundamental problem in robotics, augmented reality (AR), autonomous vehicles, and computer vision.

Relocalization is essential for ensuring that localization systems remain robust and reliable, especially under real-world conditions such as occlusion, motion blur, or changing environments. Its presence allows devices to recover quickly from errors, maintain situational awareness, and continue operating safely and effectively.

In recent years, it has also been used in technologies such as autonomous vehicles, drones, augmented reality, and mixed reality. Therefore, improving matching accuracy and increasing computational speed have become increasingly important. A concrete example of relocalization is the estimation of self-location by autonomous mobile robots during navigation in environments where GPS is unavailable, allowing for more accurate route recognition and movement.

In this project, we focus on a practical relocalization scenario involving autonomous mobile robots navigating in environments where GPS signals are unavailable. Specifically, we consider the problem of estimating the robot's position by matching partial 3D shape data — represented as point clouds obtained from LiDAR sensors — with a preconstructed global 3D map. Point clouds, formed by measuring the reflections of laser beams, are inherently subject to occlusion, where parts of objects are hidden behind others. This means that only incomplete views of the environment are typically available to the robot. Consequently, the relocalization system must be robust to such occlusions, and capable of inferring the correct location from partial data. Variation in lighting levels is still a difficult problem in the field of relocalization, especially with image data input. Our project seeks to improve upon these weak points in large-scale indoor environments.

2 Background

First, we will establish some relevant vocabulary to understand the details of this paper. Then, we will briefly explain some concepts and tools used in our approach.

1. **Point cloud:** A point cloud is a set of finitely many points in three-dimensional Euclidean space. Together, they represent a surface of a three-dimensional object. Let $\mathcal{P}$ be a point cloud and

$$\mathbf{p}_i = (x_i, y_i, z_i),$$

be a point in Cartesian coordinates. The point cloud is represented as $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{N}$, where N is the number of points in $\mathcal{P}$ [20].

2. **RGB-D Image:** Normally, images are represented using RGB, i.e. Red, Green, and Blue. Here our images including depth data, so they are called an RGB-D image.

3. **Localization and Relocalization:** Localization refers to the general process of determining a device's position and orientation within a known map. In contrast, relocalization specifically denotes the ability to recover this pose after it has been lost, such as during temporary sensor failure or when the system is reinitialized. Figure 5 shows an example of what the relocalization pipeline may look like.

4. **Pose estimation:** refers to the process of estimating the six degrees of freedom (6-DoF) of a camera or sensor. The degrees of freedom are the three-dimensional position and a 3D vector describing the orientation.

2.1 Feature Based Approaches

Many methods for relocalization make use of visual features, so it is worth taking the time to define them. Broadly, visual features are any component of an image that can be used to identify objects or locations within the image [29]. For example, corners could be used to identify a monitor, or a region of text might identify a soda can. There are many categories of features, which have different constructions as well as different strengths and challenges. Features that we are interested in are

- Local features: features that are extracted from a single pixel or a small collection of pixels within the image.
- Global features: a way of quantifying the whole scene contained in an image. Often, global features are made by aggregating local features.
- More Complex features: Some programs look for higher level features, such as semantic features. Semantic features associate a descriptor to aspects of an image, and are often utilized in machine learning contexts [18].

2.1.1 Feature Detection Criteria

In order to make use of features, we first need to detect them. There are a few key hypotheses that detected features should satisfy [29]

- **Distinctiveness:** a key point/pixel should look very different from its neighbors,
- **Invariance:** similar images should produce similar feature,
- **Locality:** local features should only depend on their close neighbors, not the image as a whole,
- **Quantity:** a method for detecting local features should produce a sufficient number of features,
- **Accuracy:** the features should be appropriately localized to the area of interest they describe,
- **Efficiency**: features should be easy enough to compute that they can be useful in more complex algorithms.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning architectures that have demonstrated strong performance on a wide range of computer vision tasks. They are specifically designed to exploit the spatial structure of grid-like data such as images, enabling efficient learning of visual patterns through a combination of local connectivity, parameter sharing, and hierarchical feature abstraction.

A *convolution* is a fundamental operation in CNNs in which a small matrix of learnable weights—referred to as a filter or kernel—is applied across an input image or feature map to extract localized patterns. As shown in the IBM schematic 1, the filter slides spatially over the input (a process called convolution), computing a dot product between the filter weights and the corresponding input patch at each location. The result is a new representation known as a feature map, which highlights the presence and spatial locations of features such as edges, corners, or textures.

This operation enables CNNs to detect patterns independent of their exact position in the image, thereby providing translation invariance. Additionally, because the same filter is reused across all spatial locations, the number of parameters is greatly reduced, leading to efficient and scalable learning of hierarchical visual features.

Core Architecture A typical CNN consists of the following components (see Figure 2):

- **Convolutional Layers**: These apply learnable filters to local regions of the input, producing feature maps that highlight the presence of particular spatial patterns (e.g., edges, textures).

- **Activation Functions**: Non-linear functions such as ReLU are applied to the feature maps to introduce non-linearity, allowing the network to learn complex mappings.
- **Pooling Layers**: These reduce the spatial resolution of the feature maps, improving computational efficiency and providing a degree of translation invariance.
- **Fully Connected Layers**: In classification-oriented CNNs, the final feature maps are flattened and processed by fully connected layers to perform the final decision-making. In other applications such as image retrieval or localization, these layers may be replaced by global pooling or aggregation modules.

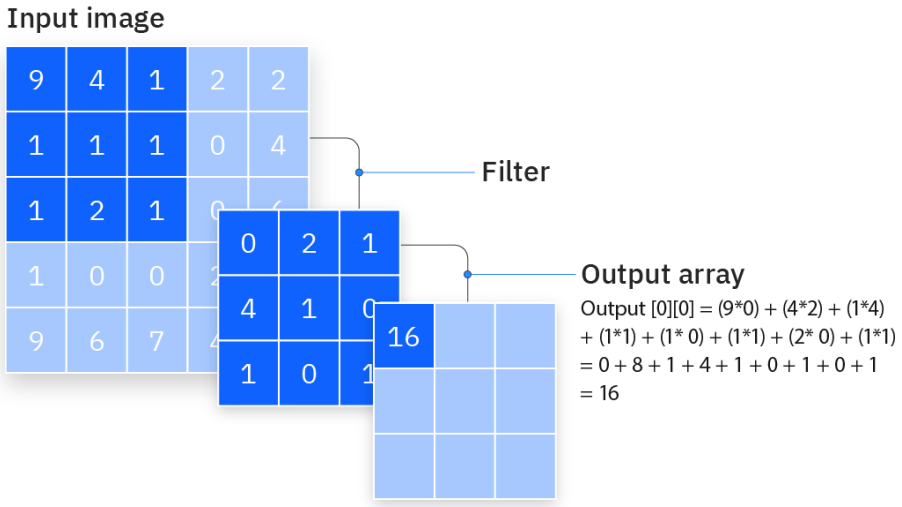


Figure 1: Illustration of a convolutional layer [11].

Advantages for Visual Recognition CNNs are particularly effective for visual recognition tasks due to several key properties. They exploit local spatial coherence by applying filters to localized regions of the input, allowing the network to detect low-level patterns such as edges and textures. The use of shared weights across spatial locations significantly reduces the number of trainable parameters, improving generalization and computational efficiency. Finally, CNNs naturally support hierarchical feature learning, whereby successive layers capture increasingly abstract and semantically meaningful representations of the input data.

2.2 Local Features

In image recognition, *local features* refer to information that characterizes a specific region within an image. These features play a crucial role in tasks such as object recognition, image matching, Simultaneous Localization and Mapping (SLAM), and Structure from Motion (SfM). Therefore, local features are required to be robust against changes in viewpoint, scale, partial occlusion, and illumination conditions.

In human vision, features are typically recognized not in flat regions but in parts of the image that exhibit significant changes when slightly shifted—such as corners, boundaries, or areas with different lighting. The goal is to replicate this form of perception computationally. In addition to detecting feature points, it is also necessary to extract *descriptors*—quantities that describe the nature of these points—so that the same features can be identified across different images. These descriptors are essential components in image recognition.

The extraction of local features generally consists of two steps: *feature point* detection and subsequent feature description. Feature points are locations that represent highly repeatable local structures in the image, such as corners, boundary intersections, or highly textured regions. Typical detection methods include Harris

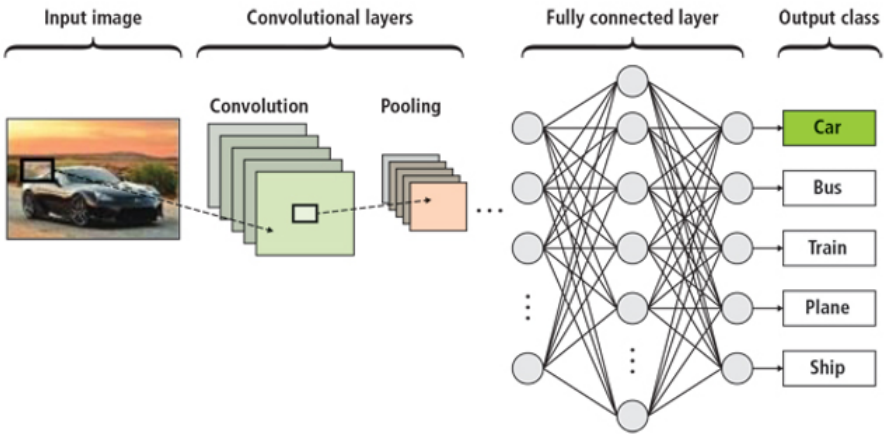


Figure 2: Schematic overview of a typical CNN workflow for image recognition [22].

corner detection and Scale-Invariant Feature Transform (SIFT)[19], which is known for its scale invariance. Depending on the context, such points may also be referred to as *interest points* or *keypoints*.

Once feature points are detected, descriptors are computed based on the local region surrounding each point. A descriptor numerically represents patterns such as intensity variations or gradient information in the neighborhood of the point, and is used for tasks such as matching corresponding points between images or evaluating similarity. In addition to classical descriptors like SIFT, Speeded-Up Robust Features (SURF)[4] and Binary Robust invariant scalable keypoints (BRISK)[16], recent methods such as Oriented FAST and Rotated BRIEF (ORB)[26] and learning-based approaches like SuperPoint[8] have emerged. These integrate detection and description into a single framework, enabling fast and robust feature extraction.

2.2.1 Scale-Invariant Feature Transform (SIFT)

To illustrate how local features are extracted, we present the Scale-Invariant Feature Transform (SIFT) as a representative example. The purpose of this example is to demonstrate what a feature extraction framework might look like.

SIFT works by first detecting points of interest within an image, then computing robust, unique features at each of these points. To detect the points of interest, SIFT first computes Scale-Space extrema. Given an input image $I(x, y)$, the scale space of the image is defined to be

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

where $G(x, y, \sigma)$ is a Gaussian function with variance σ, and $*$ represents convolution. In essence, the parameter σ controls the level of image blur, and the scale space is all the blurred images stacked together. The scale space can then be used to construct the difference-of-Gaussian function, which is an image representing the difference between two layers in the scale space, explicitly:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

for fixed k. These difference-of-Gaussian images also form a stack, and extrema are computed by finding minimal or maximal pixel intensity compared to their 8 spatial neighbors, 9 upper space scale neighbors, and 9 lower space scale neighbors[19]. Figures 3 and 4 summarize this process.

Once the points of interest are selected, a filtering process is applied. First, a second order Taylor approximation of D is fitted around each point, using pairwise differences between pixels to approximate the first and second derivative terms. Then, the extremum of the approximation is calculated. If the extremum

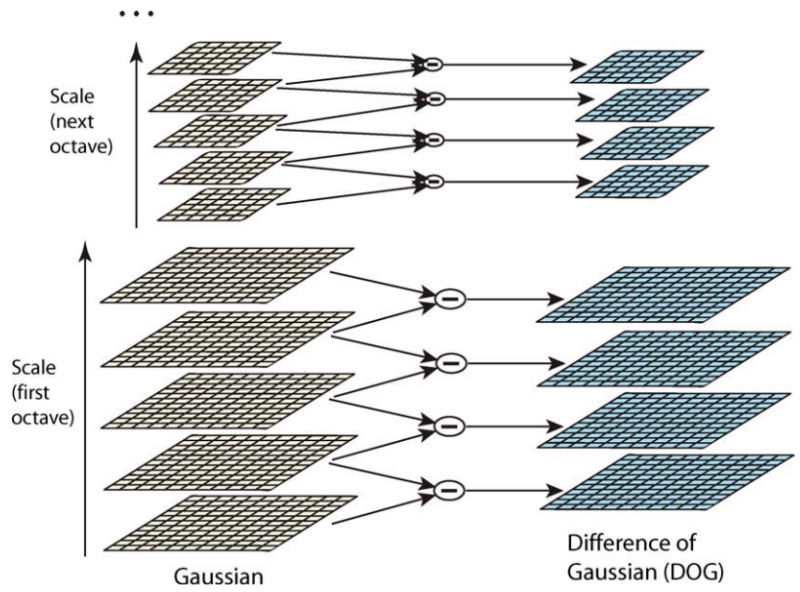


Figure 3: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.[19]

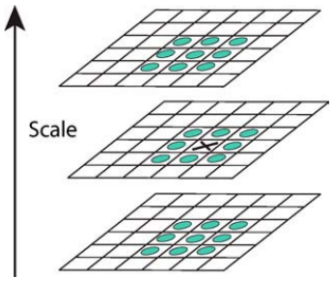


Figure 4: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with circles).[19]

lies far enough away from the center of the pixel of interest (i.e. closer to a neighboring pixel), then the neighboring pixel is selected instead, and the process repeated.

Certain points are less suited than others as key points, in particular points which are sensitive to noise. Points with low contrast are an example, but this alone is not enough. Some edge points will also be sensitive to noise, if the edge is poorly determined. However, these points will be detected as difference-of-Gaussian extrema. To eleminate these points, the ratio of the principle local curvatures is calculated as follows:

1. An approximation to the 2×2 Hessian matrix is computed from differences of nearby pixels.
2. The trace and determinant of the Hessian are computed
3. The ratio of curvatures give an upper bound for the ratio of the trace to the determinant, so it is sufficient to check if

$$\frac{\mathrm{Tr}(H)}{\det(H)} < \frac{(r+1)^2}{r}, \tag{1}$$

to eliminate poorly localized points. Once the keypoints are selected, they are assigned an orientation. This is done by computing the direction and magnitude of the gradient (approximated with pixel differences) around each keypoint. Additionally, the direction and magnitude of all nearby points are also computed, and these directions are added to a histogram weighted by the magnitude. This histogram can then be used to ascribe a direction to the keypoint. If there are multiple peaks in the histogram (at least 80% the size of the primary peak), then multiple directions are ascribed to the given keypoint[19].

The output of SIFT is a collection of keypoints, with orientations given to each key point. This algorithm is fairly robust because the features defined at each key point are very distinctive, and allow for robust matching of key points between frames.

This SIFT example highlights the complexity of local feature extraction. The benefits of these complex methods is there ability to target specific mathematical descriptions of a local features. They are also typically fast to compute, and do not require any known data to train with. The challenges of these methods is that they are often sensitive to environmental variables (for example, lighting conditions). Additionally, they can be sensitive to changes in the perspective of the camera.

2.2.2 SuperPoint

This section provides an overview of SuperPoint, a machine learning-based method for keypoint detection and description. SuperPoint was developed by Magic Leap Inc. [8] and is designed to detect keypoints and generate corresponding descriptors using a single deep neural network.

Unlike traditional handcrafted algorithms such as SIFT or ORB, which rely on manually designed features, SuperPoint learns optimal features directly from data. This enables more adaptive and robust performance in complex visual environments. One of its most notable characteristics is its use of self-supervised learning, which allows the model to be trained on large amounts of unlabeled data without the need for manual annotations.

Self-Supervised Learning Self-supervised learning is a training paradigm in which pseudo ground-truth labels are generated automatically from the structure of the data itself. These pseudo-labels are then used to supervise the learning process, mimicking conventional supervised learning without requiring human-labeled data. SuperPoint adopts a two-stage self-supervised training strategy based on homographic transformations and geometric consistency across image pairs. The overall procedure is as follows:

1. A CNN-based keypoint detector named “MagicPoint” is first trained on a synthetic dataset composed of simple geometric shapes such as circles, lines, and corners. This dataset provides manually labeled keypoints for initial supervised learning.
2. Preparing a large set of unlabeled real-world images. For each image, a random homography transformation is applied to generate a warped version of the image.

3. MagicPoint is used to detect keypoints in the warped image. These detected keypoints are then mapped back to the original image coordinates using the inverse homography. By applying this process across multiple homographies, a heatmap representing pseudo ground truth keypoints is generated for each original image.

4. Finally, the full SuperPoint network is trained using these pseudo-labeled image pairs. The model jointly learns to detect keypoints and compute corresponding descriptors from real images—without requiring manual annotations.

This approach exploits the geometric consistency of keypoints under homographic transformations, enabling large-scale training using only unlabeled data. It eliminates the need for labor-intensive annotation while preserving the structure of supervised training through automatically generated labels.

Mathematical Formulation Let $\mathbf{x}$ denote the set (or heatmap) of detected keypoints, and let $f_\theta(\cdot)$ be the MagicPoint detector. For an input image I, keypoints are detected as:

$$\mathbf{x} = f_\theta(I) \tag{2}$$

Let $\mathcal{H}$ denote a homography transformation applied to the image. Then the transformed keypoints are written as:

$$\mathcal{H}\mathbf{x} \tag{3}$$

The keypoints detected from the transformed image $\mathcal{H}(I)$ are:

$$f_\theta(\mathcal{H}(I)) \tag{4}$$

Ideally, these should match the transformed original keypoints, i.e.,

$$\mathcal{H}\mathbf{x} = f_\theta(\mathcal{H}(I)) \quad \Rightarrow \quad \mathbf{x} = \mathcal{H}^{-1} f_\theta(\mathcal{H}(I)) \tag{5}$$

In practice, exact agreement is not guaranteed due to noise and model imperfections. To improve robustness, multiple homographies $\mathcal{H}_i$ are applied, and the results are averaged:

$$F(I; f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} f_\theta(\mathcal{H}_i(I)) \tag{6}$$

This averaged output serves as a pseudo ground-truth heatmap for the original image and is used to supervise the training of the SuperPoint model.

Performance and Comparison SuperPoint offers an excellent trade-off between accuracy and computational efficiency. While traditional methods like SIFT provide high precision, they are computationally expensive due to operations such as scale-space construction and gradient histograms. ORB, by contrast, is fast and lightweight but often less accurate and robust.

SuperPoint achieves real-time performance on VGA-resolution images when executed on a GPU, making it particularly well-suited for applications such as visual SLAM and drone navigation where both speed and robustness are critical. Compared to other learning-based methods like D2-Net or R2D2, SuperPoint is relatively lightweight, reducing the computational burden for real-time deployment.

Nevertheless, several limitations exist. Because SuperPoint relies on a convolutional neural network, it requires a GPU or similar computational resources—even during inference. Although the model is relatively efficient, it is still more demanding than traditional methods such as ORB.

Additionally, in scenarios with extreme scale changes, SIFT may offer more stable performance. In CPU-only environments, the inference cost of SuperPoint can become a bottleneck. Furthermore, fine-tuning or retraining the model requires expertise in deep learning.

Summary SuperPoint combines the benefits of self-supervised learning with high accuracy and low annotation costs. It is capable of real-time performance, maintains robustness under viewpoint variation, and provides a practical and scalable solution for keypoint detection and description in computer vision applications.

2.3 Global Features

In contrast to local features, a *global feature* represents the entire image as a single feature vector. Rather than focusing on individual regions, global features summarize the overall *statistical*, *structural*, and *visual* properties of the image. This makes them especially suitable for tasks such as image retrieval and classification, where measuring holistic similarity is crucial.

Typical examples of global features include color histograms, texture statistics computed over the whole image, and shape or morphological descriptors. These representations provide a compact, quantifiable summary of the image's overall appearance or structural layout.

In recent years, deep learning-based methods have been proposed that integrate both local feature extraction and aggregation into an end-to-end trainable framework. These methods offer improved representational power and flexibility compared to traditional hand-crafted features. These models have demonstrated state-of-the-art performance in image retrieval and classification tasks.

2.3.1 Bag-of-Visual-Words (BoVW)

The Bag-of-Visual-Word (BoVW) model [27] is a classical method for constructing global image representations. In this approach, local features such as SIFT are first extracted from an image and then quantized into visual words using k-means clustering. The image is finally represented by a histogram that counts the occurrences of these visual words.

2.3.2 NetVLAD

NetVLAD is a technique introduced in [3] that aggregates local features extracted by a CNN to form a global representation, leveraging weakly supervised learning. Specifically, NetVLAD takes as input a set of local descriptors obtained from intermediate CNN layers and outputs a single L^2-normalized vector, serving as a compact and discriminative global feature. By combining these two learning-based components, NetVLAD enables seamless end-to-end optimization and aims to achieve greater flexibility and robustness than traditional hand-crafted global features.

Training Weak supervised learning is used to optimize the parameters of the NetVLAD layer, which is based on the *triplet loss* framework. For a given query image q, collect a set of geographically close images $\{p_q^i\}$ (positive examples) and a set of geographically distant images $\{n_q^j\}$ (negative examples). The loss function is then defined as:

$$\mathcal{L}_\theta := \sum_j \max\left\{0, \min_i d_\theta^2(q, p_q^i) - d_\theta^2(q, n_q^j) + m\right\} \tag{7}$$

Here, θ represents all trainable parameters within the NetVLAD layer, including the cluster centers and aggregation coefficients. The term $d_\theta(q, p_q^i)$ denotes the distance between the global features of the query image q and a positive image p_q^i, while $d_\theta(q, n_q^j)$ denotes the distance to a negative image n_q^j. The scalar m is a margin parameter.

Minimizing the loss $\mathcal{L}_\theta$ encourages the network to reduce the distance between the query and its closest positive examples, while increasing the distance to negative examples. As a result, the model learns global features that reflect spatial similarity and exhibit strong discriminative power.

Comparison Traditional approaches, such as the BoVW model, first extracts local features like SIFT from the image and then applies k-means clustering to partition them into k groups. Each group is associated with a cluster center (centroid). In the VLAD (Vector of Locally Aggregated Descriptors) method [14], global features are formed by computing the residuals between local features and their corresponding cluster centers, and then summing these residuals within each cluster.

NetVLAD generalizes this idea by optimizing the cluster centers and the assignment coefficients for residual aggregation through a weakly supervised learning process based on triplet loss. For pretraining, the dataset must include GPS or other location information associated with each image. The optimization goal

is to make the global features of geographically close images similar, while those of distant images remain dissimilar.

In our project pipeline, the NetVLAD architecture is trained through a weakly supervised learning process using location information from the images that constitute the 3D map. After training, image retrieval is performed against these reference images for a given query image.

2.4 Other Tools

2.4.1 Random Sample Consensus (RANSAC)

Random sample consensus (RANSAC) is an algorithm we use to deal with noisy data. RANSAC works by randomly selecting a subset of our data, then fitting a model to the small subset. The fitted model is then evaluated on the entire model, and inliers are determined by computing error for each data point and comparing to a threshold. A parameter determines the number of inliers required to declare the model a success (this could be something like 66% of the data). If the model is successful, all inliers are used to compute an improved model, and error is calculated. If the new error is better than the previously reported best error, the new model is accepted, otherwise it is rejected. We use RANSAC in multiple places to prevent outier data from skewing our pose estimation.

First, four pairs of points are randomly selected from the matched feature points, and a projective transformation (homography) that overlaps the two images is estimated based on them. Next, this transformation is applied to all corresponding points, and a determination is made as to whether the deviation between the transformed position and the actual corresponding point is within a given threshold (e.g. 5 pixels). Points with small deviations are adopted as inliers, and points outside the threshold are removed as outliers. This process is repeated multiple times, and the transformation that results in the greatest number of inliers is ultimately adopted, eliminating false matches and extracting only reliable corresponding points. As for the mechanism of RANSAC, this method is implemented by the findHomography function of OpenCV.

2.4.2 Kabsch

The Kabsch algorithm is a method used to compute the optimal rotation matrix that aligns two sets of corresponding points in Euclidean space, typically in three dimensions. It minimizes the root mean square deviation (RMSD) between the point sets by finding the rotation (and optionally a translation) that best aligns them in a least-squares sense. The algorithm involves centering both point sets at their respective centroids, computing the covariance matrix, and then performing singular value decomposition (SVD) to extract the optimal rotation. The Kabsch algorithm is widely used in computer vision and structural biology, particularly for tasks such as point cloud registration and pose estimation.

Algorithm 1 Kabsch Algorithm

Require: Two sets of corresponding points: $\mathbf{P} = \{p_1, \ldots, p_n\}$ and $\mathbf{Q} = \{q_1, \ldots, q_n\}$
Ensure: Optimal rotation matrix R minimizing RMSD between $\mathbf{P}$ and $\mathbf{Q}$
Compute centroids $\bar{p}$ and $\bar{q}$ of $\mathbf{P}$ and $\mathbf{Q}$
Center the points: $\tilde{p}_i = p_i - \bar{p}$ and $\tilde{q}_i = q_i - \bar{q}$ for all i
Form centered matrices $\tilde{\mathbf{P}}, \tilde{\mathbf{Q}}$
Compute covariance matrix: $H = \tilde{\mathbf{P}}^T \tilde{\mathbf{Q}}$
Perform singular value decomposition: $H = U \Sigma V^T$
if $\det(VU^T) < 0$ **then**
    Set $D = \text{diag}(1, 1, -1)$
else
    Set $D = I$
end if
Compute optimal rotation: $R = VDU^T$
Compute optimal translation: $t = -\tilde{p} + \tilde{q}$

3 Our Approach

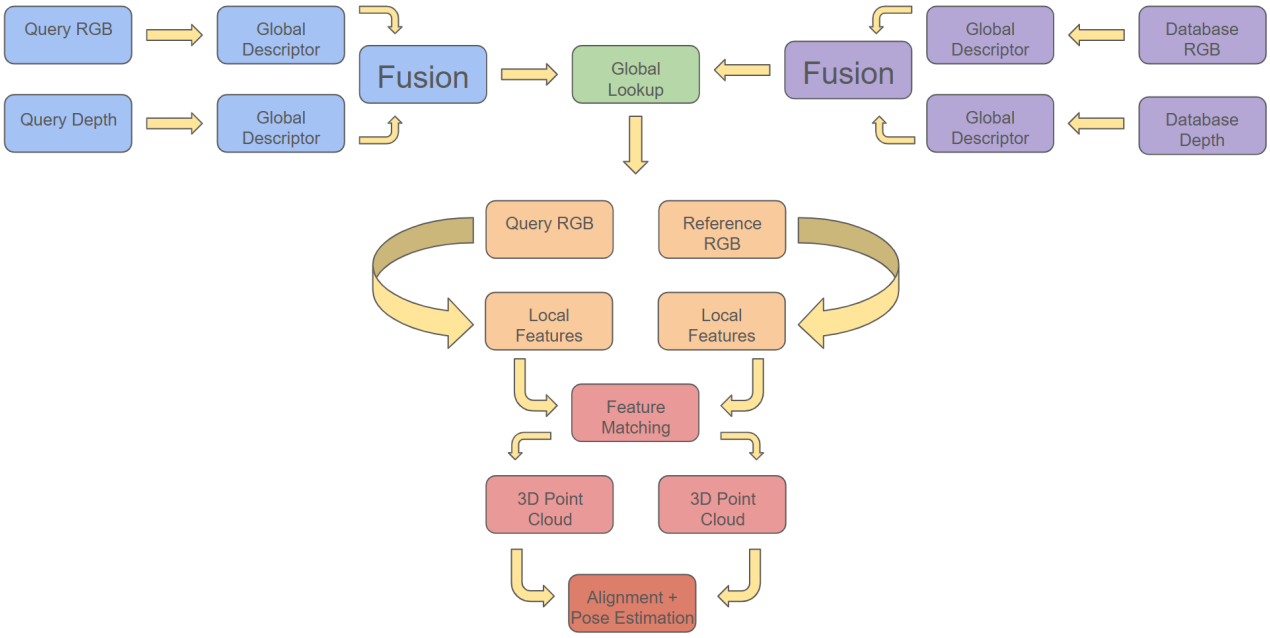


Figure 5: Pipeline showing the stages of relocalization from query image to predicted pose.

This section describes our approach to achieving highly accurate relocalization. We implement the relocalization process by breaking it down into several parts, and we call this series of steps a *pipeline*. See Figure **??**. The pipeline is as follows: First, global matching is performed to search for similar images in the dataset as a map for the input image, roughly narrowing down the images. Next, local matching is performed to recognize and identify the position of identical objects in the narrowed down images. The corresponding matching information is projected onto a 3D point cloud created from the depth information, and the position in 3D and the camera position, i.e., 6DoF, are estimated.

3.1 Global Matching

This section describes the method of global matching employed in the project's processing pipeline. The ultimate goal of this project is to estimate the 6-DoF parameters representing the position and orientation of a robot. Prior to this estimation, however, it is necessary to narrow down candidate locations at a coarse level—a task handled by global matching.

Specifically, this involves computing similarity between an image captured by a robot that has lost its positional information and images stored in a database used for constructing a 3D map. Through this image retrieval process, candidate locations that are visually similar to the query image are identified. In this process, each image is represented by a global feature vector that captures its overall visual characteristics. The similarity between images is then evaluated by computing distances between these vectors.

Accordingly, this section explains the construction of global features and the computation of distances between them. In particular, we focus on the techniques designed to achieve relocalization that is robust to changes in illumination.

3.1.1 Fusion of RGB and Depth Data

RGB data represents the color information captured by a camera. In the context of object recognition and image retrieval, keypoints are typically extracted from regions exhibiting significant color variation, and the surrounding information is analyzed to identify objects within the image. However, in indoor environments where robots typically operate, lighting conditions are often suboptimal. As a result, captured images may

be dark or have low contrast, making it difficult to accurately extract keypoints. This, in turn, can lead to a significant decline in the accuracy of object recognition and image retrieval.

Depth data, by contrast, provides information about the distance between the camera and the objects in the scene. Since it is less sensitive to variations in lighting, it is expected to serve as a more robust source of information under challenging visual conditions. This project therefore aims to incorporate depth data into the global matching process used in image retrieval, with the goal of achieving more reliable performance.

A related study on relocalization that employs a similar pipeline is presented in [31]. While that study also performs global matching, it relies solely on RGB data. In contrast, the present work integrates depth data from the global matching stage onward. This approach is expected to enable more robust image retrieval and relocalization, particularly in environments with varying illumination, compared to existing RGB-only methods.

Fusion in Convolutional Neural Networks (CNNs) can be categorized into early, middle, and late fusion, depending on when different types of data are integrated during processing.

In **early fusion**, multiple modalities are concatenated at the input stage and processed as a single input. This method is simple and computationally efficient, but it often fails to fully exploit the unique characteristics of each modality. In **middle fusion**, each modality is first processed independently, and then fused at intermediate layers of the CNN. Since features are partially extracted before integration, this approach allows a more balanced and effective combination of modality-specific information. In **late fusion**, each modality is processed independently until the final stages, and integration is performed at the level of final features or predictions. While this method offers greater flexibility and robustness, it tends to incur higher computational costs.

In this project, we aim to improve image retrieval performance by enabling a robot to flexibly determine whether RGB or depth information is more reliable depending on the scene captured in the query image. To this end, we adopt a late fusion approach, which is well-suited for such adaptive selection based on scene conditions.

3.1.2 Global Feature Extraction

This section describes the method for extracting global features from RGB and depth data, as well as computing similarity between images. The diagram below illustrates the overall process of feature extraction for a query image i and a dataset image j. Here, i_{RGB} and i_{Depth} denote the RGB and depth data of image i, respectively, while j_{RGB} and j_{Depth} represent the corresponding data for image j.

A key aspect of our approach is that **RGB and depth data are processed using separate and potentially specialized feature extraction pipelines**, reflecting the distinct nature of these modalities. Specifically, for the RGB inputs i_{RGB} and j_{RGB}, we apply a combination of **ResNet18** and NetVLAD to obtain the global descriptors $f_{i_{\mathrm{RGB}}}$ and $f_{j_{\mathrm{RGB}}}$. ResNet18 is a convolutional neural network consisting of 18 layers based on residual learning. Despite its lightweight architecture, it effectively achieves high performance while maintaining computational efficiency. This backbone was selected for our NetVLAD pipeline due to its favorable trade-off between computational efficiency and representational power. As highlighted in the benchmark study by Berton et al.[5], ResNet-based architectures remain competitive in visual geo-localization tasks, and ResNet-18 in particular offers low inference latency while retaining sufficient depth to extract semantically meaningful features. Chen et al.[6] further emphasize that lightweight backbones such as ResNet-18 can serve as effective encoders when paired with strong aggregation modules like NetVLAD, particularly in scenarios with limited computational resources. Additionally, recent work by Lee et al.[15] reinforces that compact architectures like ResNet-18 are especially suitable for real-time or resource-constrained applications, while still enabling meaningful descriptor learning in retrieval settings.

For the depth inputs i_{Depth} and j_{Depth}, we also employ a separate ResNet18 + NetVLAD pipeline, resulting in the global descriptors $f_{i_{\mathrm{Depth}}}$ and $f_{j_{\mathrm{Depth}}}$. This modality-specific processing enables the network to better capture the unique characteristics of RGB and depth information independently.

3.1.3 Weighted Function

This section describes a method to compute and integrate similarity (or distance) between images based on global features extracted from both RGB and Depth data.

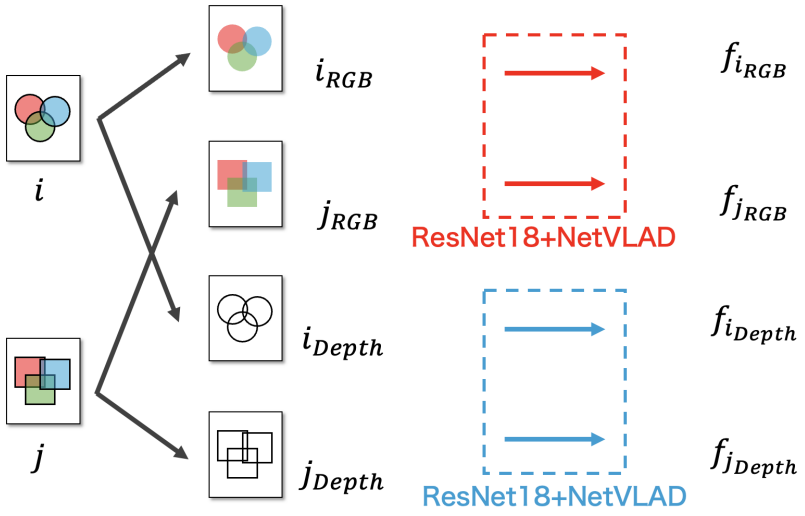


Figure 6: An overview of the global feature extraction process from both RGB and depth data.

Let $f_{i_{\mathrm{RGB}}}, f_{j_{\mathrm{RGB}}}$ denote the global features extracted from the RGB data of images i and j, and $f_{i_{\mathrm{Depth}}}, f_{j_{\mathrm{Depth}}}$ the corresponding features from the Depth data. Then, the distance (or similarity) between the two images is computed using the L^2 norm as follows:

$$s_{\mathrm{RGB}}(i,j) = \left\| f_{i_{\mathrm{RGB}}} - f_{j_{\mathrm{RGB}}} \right\|, \tag{8}$$

$$s_{\mathrm{Depth}}(i,j) = \left\| f_{i_{\mathrm{Depth}}} - f_{j_{\mathrm{Depth}}} \right\| \tag{9}$$

To integrate the RGB and Depth-based scores, we define the final similarity score $s(i,j)$ as a weighted combination using a parameter $\alpha \in [0,1]$:

$$s(i,j) \coloneqq \alpha \cdot s_{\mathrm{RGB}}(i,j) + (1-\alpha) \cdot s_{\mathrm{Depth}}(i,j) \tag{10}$$

In this report, we refer to this fusion of RGB data and depth data as **Net-α-Fusion** or **Net-AF**. In the present study, the weighting parameter α was fixed to a constant value (e.g., $\alpha = 0.5$) due to time and implementation constraints. However, we consider that dynamically adapting α based on the input query image would be a promising future direction.

Inspired by the approach in [7], which adaptively computes similarity measures according to the content of the query image, we aim to develop a mechanism that learns a query-dependent weighting function $\alpha = \alpha(q)$. Such a function could determine the relative contribution of different modalities (e.g., RGB vs. depth) depending on the visual characteristics or reliability of each input.

This extension would enhance the flexibility and robustness of the retrieval system, particularly in challenging scenarios such as varying illumination or sensor noise, where either RGB or depth information may be less reliable. Future work will investigate training such a function using weakly supervised learning from RGB-D datasets with pose annotations.

3.2 Local Matching

This section describes our implementation of local matching. To recognize corresponding objects between two images, we extracted local features and performed matching based on their descriptors. We refer to the process of extracting local features and identifying matching points using descriptors as **local matching**. This allows us to locate the same object in both images being compared. When applied to image pairs already filtered by global matching, local matching can be used to estimate the relative camera pose.

In this study, we employed two approaches for extracting and matching local features:

1. SIFT, a traditional hand-crafted method, along with its extension Affine SIFT (ASIFT)[30].

2. SuperPoint[8], a deep learning-based method.

In both approaches, keypoints are detected in the input images, and descriptors are computed for these keypoints. Feature correspondences are then established based on descriptor similarity, and corresponding points are visualized by drawing lines between them. To further increase reliability, outliers in the matches are removed using RANSAC.

In the following sections, we provide a more detailed explanation of the ASIFT and SuperPoint methods used for local matching.

3.2.1 Handcrafted Method: ASIFT

In this study, we focused on SIFT as a method for feature extraction and local matching. SIFT is invariant to image scale and rotation, and exhibits relative robustness to illumination changes and image noise. These properties enable reliable feature detection and matching in applications such as scene recognition and relocalization, which demand high robustness.

Other hand-crafted feature extraction methods—such as SURF, BRISK, and ORB—are also widely used, each with its own trade-offs. For instance, SURF operates faster than SIFT but is restricted by patents, and its matching accuracy may degrade under certain conditions. While BRISK and ORB offer computational efficiency through binary descriptors, their feature representation accuracy tends to be slightly lower.

Based on these considerations, we implemented an image matching method using SIFT, emphasizing accuracy and robustness. SIFT effectively extracts local features that are invariant to rotation and scale, and has been widely applied to identify correspondences between images. However, SIFT alone may struggle with significant viewpoint variations, particularly those involving affine transformations due to large camera angle changes.

To address this limitation, we adopted the approach of **Affine SIFT (ASIFT)**, which applies multiple affine transformations to the input image to simulate changes in viewpoint. SIFT-based matching is then performed on each transformed image. This technique follows the method proposed in ASIFT[30], which involves simulating all possible camera poses—especially the two degrees of freedom in viewpoint orientation—through image deformation. The implementation was carried out in Python using the *OpenCV* library.

The steps of our implementation are as follows:

1. Apply affine transformations to the comparison image to simulate various viewpoints (i.e., variations in elevation and azimuth angles).

2. Extract SIFT features from both the target image and each transformed image, and perform feature matching.

3. Evaluate the quality of each match (e.g., by the number or quality of matches) and select the transformation with the best result.

4. Visualize the matched keypoints for the optimal transformation, revert the transformed image to its original state, and display the corresponding points on the original image.

To further improve the reliability of the correspondences, we applied geometric outlier rejection using RANSAC based on epipolar constraints.

3.2.2 Learning-Based Method: SuperPoint

In this study, we conducted experiments on keypoint extraction using a publicly available PyTorch implementation of SuperPoint[12]. This implementation is based on the original SuperPoint framework[8] and its subsequent improvement[13]. Specifically, we used a pretrained SuperPoint model trained on the COCO dataset[17] to extract keypoints and their corresponding descriptors from input images.

The model was applied to a pair of images taken from different viewpoints. Keypoints and descriptors were extracted independently from each image, and feature point correspondences were obtained through descriptor matching. The resulting matches were visualized to qualitatively evaluate the consistency of detected keypoints across different views. Both keypoints and descriptors were visualized, and the detection performance was evaluated qualitatively through visual inspection.

Summary of Local Matching

ASIFT is a handcrafted method, which makes its internal mechanisms transparent and its behavior relatively easy to interpret. It is generally capable of detecting features that are also distinguishable by human observers. However, it is computationally intensive, as it requires feature comparison at each image transformation. On the other hand, SuperPoint is a machine learning-based approach that allows for rapid image comparison once training is complete. Nevertheless, as with many learning-based methods, the feature detection process tends to function as a "black box," making it less interpretable. Considering the respective advantages and limitations of these methods, we employed both in our experiments. Specifically, we applied global matching to narrow down candidate image pairs, followed by local matching using either ASIFT or SuperPoint. Note that both SIFT and SuperPoint assume grayscale input images.

3.3 2D to 3D Projection

Local features extracted from a 2D image were projected onto a depth image, and the corresponding points were also projected onto a point cloud generated from the depth information.

First, local features were extracted from a normal 2D image. The same points were projected onto an image representing depth information. Figure 7 is an example.

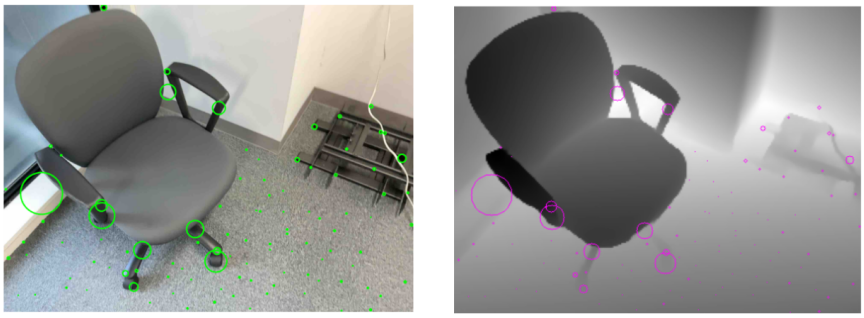


Figure 7: The image on the left is a normal 2D image. Note that the local feature extraction method we used assumes a grayscale image, but here it is colored for visibility. The image on the right is an image that expresses depth information. The closer the object is, the blacker it is, and the further back it is, the whiter it is.

This allowed us to assign features and descriptors corresponding to each depth point. We then used this to map the same points onto the 3D point cloud data. See Figure 8. Now the point cloud has the same feature

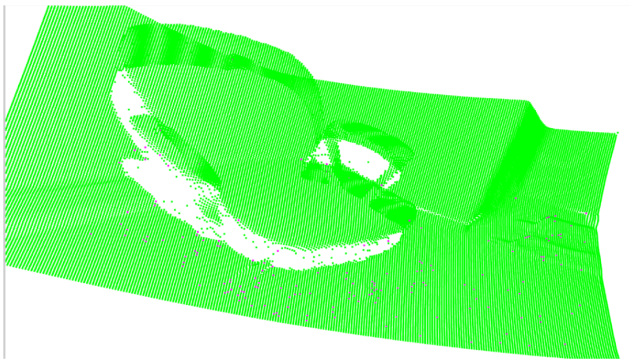


Figure 8: Pink points are same feature points as the normal one.

points assigned to it, which will be used to estimate the camera position.

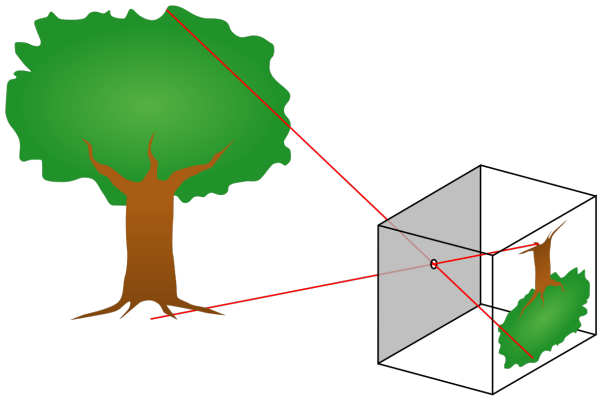


Figure 9: Cartoon depicting the pinhole camera model. Source: Wikipedia

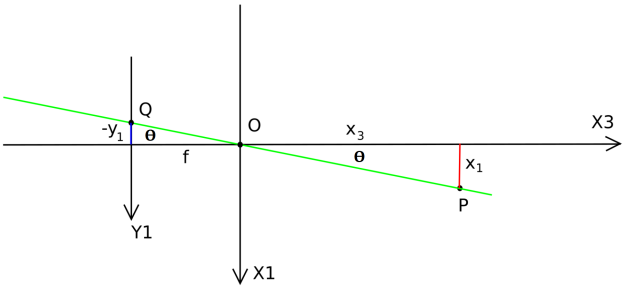


Figure 10: Diagram showing the geometry of a pinhole camera image. Source: Wikipedia

3.4 Depth to Point Cloud

RGB-D data is inherently three dimensional, and we exploit that by constructing explicit point clouds from the depth data of our images. A useful simplified model for doing this is the pinhole camera model (see figure 9). The pinhole camera model assumes that all light rays from the image are focused through a single point before reaching the detector (which detects a scaled, flipped version of the image). The distance from the detector to the lens is known as the focal length, and the angle of the light cone is know as the field-of-view or FOV. These values are intrinsic to the camera, and must be known to use this model [10].

To reconstruct the original image from the picture is actually just a trigonometry problem. In fig 10, we can see that θ is the same for both the detected image and the original scene. The distance from the lens to the detector is the focal length f, and so we can derive θ as

$$\theta = \arctan\left(\frac{-y_1}{f}\right). \tag{11}$$

Then, using the fact that that distance d from the point in the original image to the lens is the value recorded in the depth data, we get that

$$x_1 = d\tan(\theta), \tag{12}$$

$$= \frac{-y_1 d}{f}. \tag{13}$$

A similar calculation allows us to determine x_2, and we set $x_3 = d$, thus achieving a 3D representation of each pixel in our depth image.

3.5 Camera Pose Estimation

To estimate our camera pose, we use the fact that we have 3D matched keypoints allows us to reformulate the problem as follows: given two collections of 3D points $K = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_n\}$ and $K' = \{\mathbf{k}'_1, \mathbf{k}'_2, ..., \mathbf{k}'_n\}$, compute the rotation matrix R and translation vector $\mathbf{t}$ that minimizes

$$\sum_{i=1}^{n} ||(R\mathbf{k}_i + \mathbf{t}) - \mathbf{k}'_i||_2^2. \tag{14}$$

A standard method for doing this is the Kabsch algorithm [2]. In particular, the Kabsch algorithm allows us to match a few correlated points, rather than trying to align whole point clouds. One known weakness, however, is that Kabsch can be susceptible to noise of depth data. To remedy this, we also employ a RANSAC algorithm when computing R. This allows us to fit an accurate rigid transform to the data, without being skewed by high noise points.

4 Experiments

4.1 Data & Preprocessing

To evaluate our method, we utilize the Standford 2D-3D-Semantics dataset (2D-3D-S). This dataset offers a large-scale, diverse, and realistic environment with ground truth data for testing. We used the Area3 subset of 2D-3D-S. We chose to work with this dataset because it is a large collection of interior RGB-D data, with some variance in shadows and lighting conditions.

We split the dataset into subsets called "train" and "query" with 80% and 20% of the data, respectively. Each subset contains RGB, depth, and ground truth pose data. The depth data had to be preprocessed to purge any unrealistic values that could interfere with good training. For triplet loss training, we performed **triplet mining** by using the ground truth pose data to find the distances between all images. For all images, called anchors, we find good positives and negatives determined by how spatially close or far other images are to the anchor. The threshold values were chosen such that we would get sufficient matches for positives and negatives which are 0.5 meters and 10.0 meters, respectively, storing the top ten negatives. This triplet mining was performed for both the RGB and preprocessed depth data.

To make the dataset better for testing lighting variance, we also artificially modified the data. We chose to leave 40% of the data unmodified, and the remaining 60% are assigned a random brightness value from 0.4 to 1.0, and a random gamma value from 0.6 to 1.4. We then choose whether to adjust the color of the image uniformly at random. We either do not modify the color, make the color warm, or make the color cool. We adjust the image color using the cv2.addWeighted function. This lighting modified data was used only for testing, not training, due to time constraints.

4.2 Training

As discussed before, we trained NetVLAD with each modality (RGB and depth) separately using a *triplet loss* function [23]. We loaded NetVLAD with the ResNet18 backbone and its pretrained weights. The data is transformed into 224 by 224 images then loaded into ResNet18+NetVLAD. NetVLAD has 64 clusters, batch size 32 for 10 epochs with a triplet loss margin of 0.7 using the L2 norm. Gradient descent is performed with Adam and a learning rate of 1e-4.

4.3 Results

We performed several experiments to validate the methods we used in this project. We did small scale experimentation to evaluate each component individually, then performed large scale tests on the full pipeline.

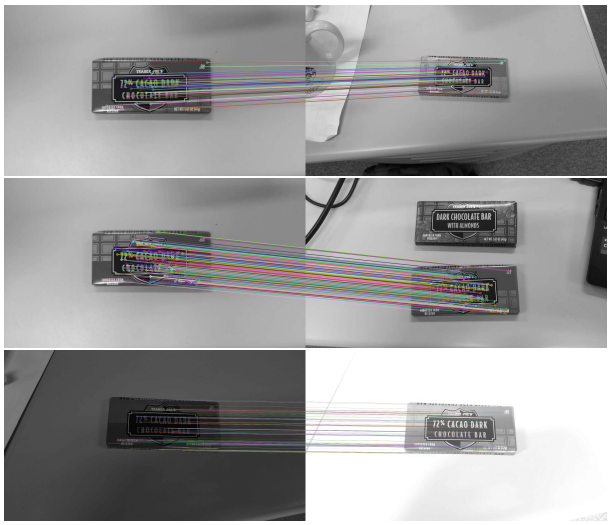
4.3.1 Local Matching

First, we conducted some experimentation to validate each component of our method. For local matching, we conducted small scale experiments where the position, camera angle, and illumination of various objects

were adjusted, and then tried to use ASIFT and SuperPoint to match features between the two objects. Figure 11a is an example result of one such experiment.

See Figure 11a, which shows the successful identification of identical objects in the image.

We performed matching between two images using the descriptors obtained by SuperPoint, and some results are shown in Figure 11b.



(a) The matched points are connected by lines. The first image shows the same object but at different sizes on the screen. The second image shows matching when a similar object is present on the screen. The third image compares images taken under different lighting conditions.

(b) The matched points are connected by lines. The first image shows the same object but at different sizes on the screen. The second image shows matching when a similar object is present on the screen. The third image compares images taken under different lighting conditions.

Here are some comments.

- Successfully identifies identical objects within an image.
- Matches well even when objects of different sizes or similar objects are included.
- No problems with images under different lighting conditions.

Note that extracting too many feature points does not necessarily improve accuracy. What is important is that highly discriminative feature points are appropriately distributed throughout the image. Redundant and weak feature points can cause computational overhead and false positives. SIFT is a conservative design that returns a small number of carefully selected, high-quality points. SuperPoint returns dense points through training, but redundancy is suppressed. Superiority should be determined not by the score, but by matching accuracy and suitability for downstream tasks.

4.3.2 Pose Estimation

To validate pose estimation, we performed experiments where we convert two nearby images to point clouds, align them, and then compare the predicted 6DoF to the ground truth. Based on a number of small tests, we determined our pose estimation is effective in situations where a sufficient number of good feature points are matched between the two images.

4.3.3 Pipeline

In order to validate our full method, we conducted several large-scale tests on the Stanford 2D/3D/Semantics dataset area 3. We also performed tests on the illumination modified dataset, denoted (IM). We tested our entire pipeline with different module configurations, specifically using a pretrained NetVLAD checkpoint + ASIFT, and checkpoint + SuperPoint. We also trained NetVLAD on area 3 using RGB and Depth, then tested our weights with the following configurations:

Configuration	Dataset	Translation success (%)	Rotation Success (%)	Full Success (%)
Base NetVLAD + ASIFT	area 3	97.15	86.92	86.16
Base NetVLAD + SuperPoint	area 3	95.17	86.69	84.75
Trained Net-AF + SuperPoint	area 3	6.76	23.31	6.76
Base NetVLAD + SuperPoint	IM area 3	23.81	81.72	23.81
Untrained Net-AF + SuperPoint	IM area 3	23.17	75.18	23.11

Table 1: Table showing the results of several large scale experiments on the Stanford 2D/3D/S dataset. IM means the data was artificially modified to have illumination variance. Translation success means within 5cm of true position. Rotation success means within 5° of true rotation. Full success means both are true simultaneously.

1. α fusion + ASIFT,
2. α fusion + SuperPoint,
3. RGB only NetVLAD + ASIFT,
4. RGB only NetVLAD + SuperPoint.

Our results are summarized in table 1. We used the ground truth 6DoF data to validate our pose estimation. We count a pose estimation as successful if the estimated location is less than 5cm (0.05m) off from the ground truth location and the estimated camera direction is less than 5° off from the ground truth camera direction.

These results show that our pipeline shows good accuracy for well illuminated images, but poor accuracy for dark images. A likely cause of this is the fact that SuperPoint is not trained on our data, so it cannot match features well with large lighting changes. A goal for future work would be to train SuperPoint to recognize similar features in different illumination conditions.

5 Conclusion

The point of this project was to develop a novel method for relocalization that is robust to lighting changes. Although we were unable to achieve state of the art accuracy, we still learned several valuable lessons about relocalization that can hopefully be of use in the future.

RGB-D data is likely to become much more common in the next few years, do to the rise in LiDAR sensors on modern smart phones. While this data makes it easier to extract 3D information (such as camera 6DoF), the quality of information is subject to the quality of the RGB-D data. Camera phones and tablets are not highly sensitive tools for measuring, and we found that often it was difficult to validate the ground truth of such images for training and validation. The lesson we learned here is that, depending on what type of sensor is used, data processing and cleaning may be more impactful towards improving accuracy than using more advanced tools.

We also learned that machine learning based methods require careful training. We tried multiple times to train NetVLAD on area 3, and found that without properly processing our data the results were not good. We learned that properly choosing parameters for training a neural network is a difficult task, and requires substantial time to determine.

In the end, we are not computer scientists, nor machine learning experts, and though this hampered our ability to complete this task, we still hope our proposed method can be useful as a starting point for future work. To achieve accurate and robust relocalization is a very difficult problem, but one that is nonetheless very important to explore. We appreciate all the support this program has provided for doing just that.

6 Future Work

6.1 Local Matching

6.1.1 For Extremely Distorted Images

The method we adopted is robust enough to handle some degree of viewpoint changes. However, this is not the case with extremely distorted images, and there are cases where objects that appear to the human eye are not detected properly. In images like the one shown in Figure 12, neither ASIFT nor SuperPoint were able to match properly (Figure 13).

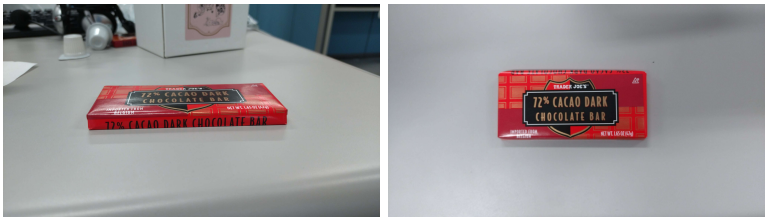


Figure 12: Example of distorted images. In this case, the first photograph was taken from an extreme angle.

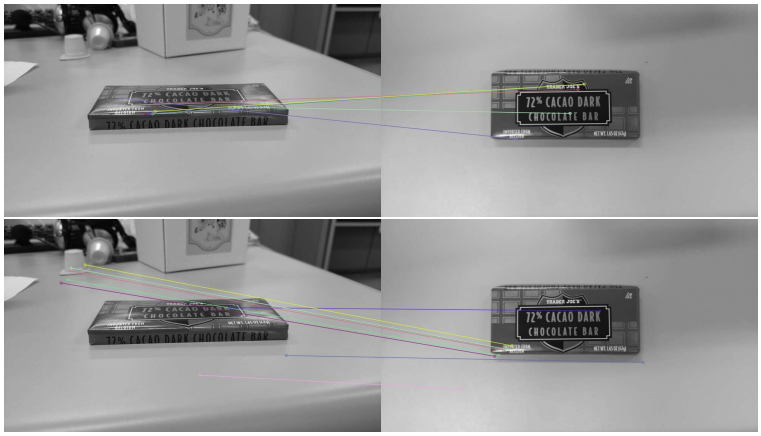


Figure 13: Fail to match. Upper is conducted by ASIFT, Lower is conducted by SuperPoint.

However, instead of directly comparing the images, we were able to detect the same object by first matching it via a slightly closer image. This was successful for both ASIFT and SuperPoint, as shown in Figure 14,15. Looking at these results, it seems possible to deal with even extreme distortions if an algorithm can be constructed that automatically selects the image to be used when a successful match is not made, and then performs matching again to identify the same object.

6.1.2 For Color Information

In this study, we conducted local matching experiments using feature extraction algorithms such as SIFT and SuperPoint, which assume grayscale images as input. These methods are specialized in detecting local structures based on intensity gradients and are highly robust to variations in illumination and instability in color spaces. Therefore, they were appropriate choices for our experiments. On the other hand, it has been pointed out that color information—such as hue and saturation—can contribute to distinguishing objects that are difficult to differentiate based on shape information alone, and can be effective in specific application domains. As a future work, it would be meaningful to conduct comparative experiments using color-based descriptors, such as **Color SIFT (CSIFT)**[1], or learning-based methods that directly handle color images, in order to evaluate their impact on the discriminative power and matching performance of local features.

Figure 14: ASIFT

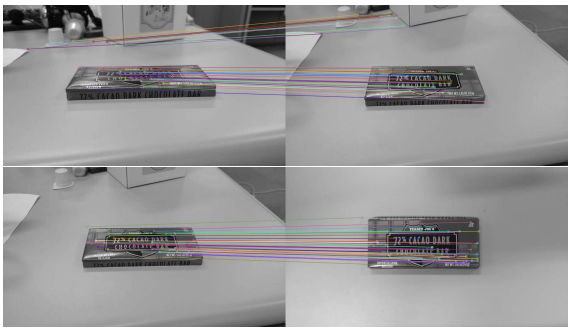


Figure 15: SuperPoint

6.1.3 SuperPoint with Depth

It is conceivable that training SuperPoint using depth images could enable more direct extraction of local features that incorporate depth information. Currently, SuperPoint is trained on standard image datasets, and therefore, applying it to depth images does not necessarily guarantee the extraction of meaningful features. However, SuperPoint is capable of being trained through self-supervised learning, which allows the use of unlabeled images during training. Given this characteristic, we consider it valuable to train SuperPoint using depth images generated from the depth data we have acquired. This approach has the potential to enhance the model's ability to extract depth-aware features suitable for our target environments.

It is conceivable that training SuperPoint on depth images could enable more direct extraction of local features that incorporate depth information. Currently, SuperPoint is trained on standard image datasets; therefore, applying it to depth images does not necessarily guarantee the extraction of meaningful features. However, SuperPoint supports self-supervised learning, which allows it to be trained using unlabeled images. Given this characteristic, we consider it valuable to train SuperPoint on depth images generated from the depth data we have collected. Such an approach has the potential to enhance the model's ability to extract depth-aware features tailored to our target environments.

6.2 Dynamically determined α

In this project, global matching is performed by combining the similarity scores between global features extracted from RGB data and those from Depth data using a weighted sum, where the weight is determined by a constant parameter α. The primary goal of this project is to achieve indoor relocalization that is robust to variations in lighting conditions. In indoor environments, accurately capturing the semantic content of objects is essential. For example, objects such as "spoons" and "forks" may have similar shapes but differ in meaning. To distinguish between such objects, semantic information—such as color and texture—is particularly important, and RGB data is more effective than Depth data in capturing such information.

However, in low-light conditions, it becomes difficult to extract sufficient features from RGB data alone. In contrast, Depth data can be acquired independently of lighting conditions, and is therefore expected to provide stable performance even when the query image lacks adequate illumination. Consequently, under the assumption that lighting conditions may vary, it is important to combine RGB and Depth data appropriately. In particular, it is desirable for the parameter α to adapt dynamically depending on the characteristics of the query image.

As future work, we plan to learn α as a function of the visual properties (e.g., brightness and contrast) of the query image. To achieve this, we intend to utilize a dataset that satisfies the following conditions:

- RGB-D images captured in indoor environments,
- Associated 6DoF camera pose information for each image,
- For each scene, images captured at different times of day (e.g., day and night).

Using such a dataset, we aim to learn a mapping from the visual conditions of the query image to the optimal α value using a method based on the triplet loss framework.

6.3 Training

Given the limited time available for training and our relative inexperience, we have identified several avenues for improving the training process in future iterations. Optimizing the existing codebase for faster execution and introducing parallelization strategies would substantially enhance computational efficiency. As noted by Chen et al.[6], increasing the descriptor dimensionality to 1024—particularly when using deeper backbones such as ResNet-50, ResNet-101, or VGG—can lead to significant improvements in retrieval accuracy. Additionally, Patel et al.[24] demonstrate that employing substantially larger batch sizes during training contributes to superior performance, particularly in triplet-based metric learning. It would also be valuable to experiment with higher input resolutions, such as 480×480 or 640×480, in order to better preserve fine-grained visual details. However, the increased memory requirements associated with such configurations currently exceed the computational resources available to us during this project.

6.4 How to handle depth data

6.4.1 Adding Edge Information

In this project, depth data was initially processed using a single-channel CNN. However, further performance improvement may be achieved by additionally incorporating edge information extracted from the depth data as input to the network. Edge information, in this context, refers to regions in an image where abrupt changes in intensity or brightness occur—typically corresponding to object boundaries and contours. Such information can also be derived from depth data. In a previous study aiming to improve the accuracy of fingertip detection by combining RGB and depth data [9], it was demonstrated that incorporating edge information, in addition to depth data, into the CNN led to significantly better detection performance compared to conventional methods.

6.4.2 PointNetVLAD

In the global matching stage of this project, depth data are directly input into the CNN and NetVLAD modules. However, CNNs are primarily designed for processing image data with a regular 2D grid structure, making it challenging to effectively extract spatial and geometric information from data with inherently three-dimensional structures, such as depth maps. Moreover, RGB and depth images differ significantly in their data characteristics. In RGB images, each pixel represents color intensity, and adjacent pixels exhibit strong visual continuity, which CNNs can exploit to learn meaningful features. In contrast, in depth images, each pixel encodes the distance to the observed object. Depth values tend to be discontinuous at object boundaries and are more susceptible to noise and missing data, which can hinder the learning capability of CNNs.

To better handle 3D structural information such as depth data, a method known as **PointNetVLAD** [28] has been proposed. This approach combines PointNet, which extracts local features directly from 3D point clouds, with NetVLAD, which aggregates these features into a global descriptor. Given that the input to our system includes RGB-D images, it is reasonable to expect that converting the depth images into 3D point clouds using models such as the pinhole camera model, and then feeding them into the PointNetVLAD architecture, would yield global features that more effectively capture the underlying three-dimensional structure of the scene.

6.5 Other ideas

6.5.1 Light invariant grayscale

If properties about the camera sensor and light type are known, it may be possible to remove illumination variance from an image [21]. This could be useful as a preprocessing step before local feature matching. The challenge here is to know the type of light the robot will see. This paper [21] assumes sunlight, and produces a simplified model based on the properties of sunlight. Interior lights may have different properties, and this may change the model. Some knowledge of optics and physics would be useful to progress this idea.

6.5.2 Geodesic-aware

Geodesic-Aware Local Feature[25] proposes utilizing depth information to measure geodesic curves on an object's surface, which are then employed as robust local features. This approach enables the identification of even non-rigid objects across various transformations.

7 Contributions

7.1 Sam Scheuerman

I was the project manager, and worked with each group member to divide up research. I also made sure that everyone else's code could work together, and built the framework to run the full pipeline. I also set up the large scale tests we ran on the pipeline. In addition, I worked on the 3D projection and alignment portions of the pipeline, writing code to allow us to convert depth data to 3D pointclouds and align pointclouds using the Kabsch algorithm and RANSAC.

7.2 Ikkei Sato

I proposed a novel late-fusion method (Net-AF) for combining RGB and Depth features in relocalization, mathematically formulating the approach to guide implementation, and also conducted a review of existing RGB-Depth multimodal fusion literature.

7.3 Keigo Horikoshi

I have implemented local matching for 2D images. In particular, I contributed to the team by building ASIFT and introducing SuperPoint. In both methods, I have written the python code to extract local feature points and descriptors to detect identical objects in the images.

7.4 Zach Fendler

I led the machine learning component of the project, conducting an extensive literature review on state-of-the-art methods, later supported by Ikkei. I was responsible for implementing the NetVLAD architecture and implementing the initial framework for our proposed method, Net-AF. Additionally, I developed and tested several foundational concepts that informed the current design of the system.

References

[1] A.E. Abdel-Hakim and A.A. Farag. "CSIFT: A SIFT Descriptor with Color Invariant Characteristics". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1978–1983. DOI: 10.1109/CVPR.2006.95.

[2] Sérgio Agostinho et al. "(Just) a spoonful of refinements helps the registration error go down". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6108–6117.

[3] Relja Arandjelovic et al. "NetVLAD: CNN architecture for weakly supervised place recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5297–5307.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[5] Gabriele Berton et al. *Deep Visual Geo-Localization Benchmark*. https://github.com/gmberton/deep-visual-geo-localization-benchmark. GitHub repository. 2022.

[6] Wei Chen et al. "Deep Learning for Instance Retrieval: A Survey". In: *arXiv preprint arXiv:2101.11282* (2021). Revised version submitted 30 October 2022. URL: https://arxiv.org/abs/2101.11282.

[7] Yanhua Cheng et al. "Query Adaptive Similarity Measure for RGB-D Object Recognition". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 145–153. DOI: 10.1109/ICCV.2015.25.

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236.

[9] Hengkai Guo, Guijin Wang, and Xinghao Chen. "Two-stream convolutional neural network for accurate RGB-D fingertip detection using depth and edge information". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA, Dec. 2016, pp. 2608–2612.

[10] Richard. Hartley and Andrew. Zisserman. *Multiple view geometry in computer vision*. eng. 2nd ed. Cambridge, UK ; Cambridge University Press, 2003. ISBN: 9786610458127.

[11] IBM. *What are Convolutional Neural Networks?* https://www.ibm.com/think/topics/convolutional-neural-networks. Overview explanation and accompanying schematic figure. n.d. (Visited on 07/31/2025).

[12] You-Yi Jau and Rui Zhu. *pytorch-superpoint: A PyTorch implementation of SuperPoint*. https://github.com/eric-yyjau/pytorch-superpoint. Accessed: 2025-07-29. Based on TensorFlow implementation by Rémi Pautrat and Paul-Edouard Sarlin, and the official SuperPointPretrainedNetwork. Thanks to Daniel DeTone.

[13] You-Yi Jau et al. "Deep Keypoint-Based Camera Pose Estimation with Geometric Constraints". In: *CoRR* abs/2007.15122 (2020). arXiv: 2007.15122. URL: https://arxiv.org/abs/2007.15122.

[14] Hervé Jégou et al. "Aggregating local descriptors into a compact image representation". In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 3304–3311.

[15] Youngmin Lee et al. "Hybrid Descriptors for Efficient Visual Place Recognition". In: *arXiv preprint arXiv:2409.19293* (2024). URL: https://arxiv.org/abs/2409.19293.

[16] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *2011 International Conference on Computer Vision*. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[17] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[18] Wei Liu et al. "High-Level Semantic Feature Detection: A New Perspective for Pedestrian Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[19] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[20] Kan Luo et al. "3D point cloud-based place recognition: a survey". In: *Artificial Intelligence Review* 57.4 (Mar. 2024), p. 83. ISSN: 1573-7462. DOI: 10.1007/s10462-024-10713-6. URL: https://doi.org/10.1007/s10462-024-10713-6.

[21] Will Maddern et al. "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles". In: *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*. May 2014.

[22] NVIDIA. *Convolutional Neural Network*. https://www.nvidia.com/en-us/glossary/convolutional-neural-network/. Glossary definition and illustrative figure. n.d. (Visited on 07/31/2025).

[23] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703.

[24] Yash Patel, Giorgos Tolias, and Jiří Matas. "Recall@k Surrogate Loss with Large Batches and Similarity Mixup". In: *arXiv preprint arXiv:2108.11179* (2021). Also published as CVPR2022. URL: https://arxiv.org/abs/2108.11179.

[25] Guilherme Potje et al. "Learning geodesic-aware local features from RGB-D images". In: *Computer Vision and Image Understanding* 219 (June 2022), p. 103409. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2022.103409. URL: http://dx.doi.org/10.1016/j.cviu.2022.103409.

[26] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[27] Josef Sivic and Andrew Zisserman. "Video Google: A Text Retrieval Approach to Object Matching in Videos". In: *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2003, pp. 1470–1477. DOI: 10.1109/ICCV.2003.1238663.

[28] Duc Thanh Uy et al. "PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4470–4479.

[29] Martin Weinmann. "Visual Features—From Early Concepts to Modern Computer Vision". In: *Advanced Topics in Computer Vision*. Ed. by Giovanni Maria Farinella, Sebastiano Battiato, and Roberto Cipolla. London: Springer London, 2013, pp. 1–34. ISBN: 978-1-4471-5520-1. DOI: 10.1007/978-1-4471-5520-1_1. URL: https://doi.org/10.1007/978-1-4471-5520-1_1.

[30] Guoshen Yu and Jean-Michel Morel. "ASIFT: An Algorithm for Fully Affine Invariant Comparison". In: *Image Processing On Line* 1 (2011). https://doi.org/10.5201/ipol.2011.my-asift, pp. 11–38.

[31] Dmitry Yudin et al. "HPointLoc: Point-Based Indoor Place Recognition Using Synthetic RGB-D Images". In: *Neural Information Processing (ICONIP 2022)*. Vol. 13625. Lecture Notes in Computer Science. Springer, Cham, 2022, pp. 471–484. DOI: 10.1007/978-3-031-30111-7_40.