electronics

MDPI

Article

Enhancing Visual Odometry with Estimated Scene Depth: Leveraging RGB-D Data with Deep Learning

Aleksander Kostusiak * and Piotr Skrzypczyński

Institute of Robotics and Machine Intelligence, Poznan University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland; piotr.skrzypczynski@put.poznan.pl
* Correspondence: aleksankostu@gmail.com

Abstract: Advances in visual odometry (VO) systems have benefited from the widespread use of affordable RGB-D cameras, improving indoor localization and mapping accuracy. However, older sensors like the Kinect v1 face challenges due to depth inaccuracies and incomplete data. This study compares indoor VO systems that use RGB-D images, exploring methods to enhance depth information. We examine conventional image inpainting techniques and a deep learning approach, utilizing newer depth data from devices like the Kinect v2. Our research highlights the importance of refining data from lower-quality sensors, which is crucial for cost-effective VO applications. By integrating deep learning models with richer context from RGB images and more comprehensive depth references, we demonstrate improved trajectory estimation compared to standard methods. This work advances budget-friendly RGB-D VO systems for indoor mobile robots, emphasizing deep learning's role in leveraging connections between image appearance and depth data.

Keywords: visual odometry; RGB-D cameras; depth estimation; deep learning; particle swarm optimization

Citation: Kostusiak, A.; Skrzypczyński, P. Enhancing Visual Odometry with Estimated Scene Depth: Leveraging RGB-D Data with Deep Learning. *Electronics* **2024**, *13*, 2755. https://doi.org/10.3390/electronics13142755

Academic Editors: Krzysztof Okarma and Piotr Lech

Received: 12 June 2024
Revised: 4 July 2024
Accepted: 11 July 2024
Published: 13 July 2024

1. Introduction

Visual odometry (VO) [1] is the process of estimating an agent's (e.g., a robot's) position and orientation by analyzing the sequence of images captured by its onboard visual sensors. VO allows autonomous robots to navigate and understand their environment by analyzing RGB or RGB-D input. The combination of RGB and depth (RGB-D) data has been particularly beneficial for improving the accuracy and robustness of VO [2]. Depth sensors such as the Microsoft Kinect have enabled acquiring RGB-D data relatively cheaply, facilitating various applications from augmented reality to service robotics [3].

However, one major challenge with using depth sensors is the incomplete and noisy nature of the depth maps they produce. Depth sensors often generate maps with missing data due to reflective, transparent, or irregular surfaces [4], which can significantly impair the performance of VO systems. The Kinect v1, introduced in 2010, has been widely used due to its low cost and sufficient measurement capabilities for indoor localization. However, its structured-light technology, shared by devices like Intel's RealSense family, often leads to depth artifacts and significant areas with missing data and in-depth images (areas without depth). Depth completion methods address this by filling in the missing data to produce dense and accurate depth maps. While much of the existing work in depth completion focuses on applications like 3D reconstruction [5], our research uniquely applies these methods to enhance RGB-D visual odometry.

Recent advancements have brought forth newer cameras like the Kinect v2 and Kinect Azure, offering improved performance and the ability to enhance measurements from older sensors, thereby boosting overall system performance. Our approach leverages depth maps obtained from Kinect v2 to train a depth completion model for Kinect v1. This cross-device training is relatively unexplored and demonstrates significant promise in improving the robustness of depth completion models.

Moreover, we introduce a novel aspect of parameter adaptation using population-based optimization, which further distinguishes our work from existing methods. This optimization technique allows us to fine-tune our model parameters effectively, ensuring better performance in application contexts and varying environmental conditions. Our contributions are threefold:

- Demonstrating the enhancement of older sensor data using deep learning methods, utilizing Kinect v2 depth frames for training to improve performance.
- Remarking that exploiting the relationship between RGB and depth images allows for better results in frame-by-frame trajectory estimation.
- Showing the ability to improve the results in a simple VO pipeline by optimizing the algorithm's parameters to particular input data characteristics with efficient population-based algorithms.

The remainder of this paper is structured as follows. In Section 2, we discuss related work in three aspects of this system: depth completion methods, VO and SLAM with depth completion, and parameter optimization in VO and SLAM. Section 3 covers the structure of the VO system and the RGB-D dataset used in our research. In Section 4, we examine the applied parameter optimization methods, while in Section 5, we discuss the methods for scene depth estimation. Next, Section 6 describes the experiments conducted to investigate the role of estimated scene depth in VO and shows the results. Finally, Section 7 concludes the paper and outlines future work.

2. Related Work

2.1. Depth Estimation for Indoor Environments

Monocular depth estimation in indoor environments has obtained significant attention due to its practical applications in robotics, augmented reality, and 3D reconstruction. Conventional methods for depth data completion have often relied on geometric principles, such as structure from motion (SfM) and multi-view stereo, as well as classical image processing techniques. Bilateral filtering [6], energy minimization [7], and Fourier transform [8] were among the early approaches to address this problem by smoothing the depth maps while preserving edges. They were effective under controlled conditions, limited by their ineffective handling of large regions, and struggled with the complexity and variability of indoor scenes, facing challenges in dynamic or cluttered indoor settings [5].

With the rise of deep learning, more sophisticated methods have emerged, offering robust solutions that leverage large datasets and convolutional neural networks (CNNs). Eigen et al. [9] pioneered the usage of CNNs for depth estimation from a single image, demonstrating significant improvements over conventional methods. Zhang and Funkhouser [10] pioneered the usage of deep neural networks for depth completion. They proposed a two-step approach where local surface properties, such as occlusion boundaries and surface normals, are first predicted, followed by global optimization to reconstruct the depth map. Yu et al. [11] have shown that incorporating a contextual attention mechanism improves object edge preservation during depth reconstruction. Following these works, numerous architectures to enhance accuracy and efficiency, including encoder–decoder networks and attention mechanisms, have been proposed.

Comprehensive reviews [12,13] have discussed various state-of-the-art deep learning approaches, emphasizing the importance of large, annotated datasets for training and evaluating models. For instance, Ref. [14] introduced a self-supervised approach that utilizes stereo images during training but only requires monocular images during inference, thereby improving depth prediction without relying on additional sensors. Recent advancements have seen the integration of adaptive convolution operators for progressive depth map completion. Xian et al. [15] proposed a method that combines raw depth maps with RGB patches for refinement, showing significant improvements over previous methods. Senushkin et al. [16] introduced spatially-adaptive denormalization blocks to handle the statistical differences between acquired data and holes, further enhancing the depth

completion performance, while Ref. [17] proposed a multi-channel progressive attention fusion network for progressive recovery of high-resolution scene depth maps.

Recent works have refined these techniques further by integrating semantic information and leveraging pre-trained models on large-scale indoor datasets. Wu et al. [18] addressed the challenges of generalizing monocular depth estimation methods to complex indoor scenes. They proposed a structure distillation approach to produce structured but metric-agnostic depth. Combining this with a branch that learns metrics from left–right consistency allowed them to achieve metric and robust depth for generic indoor scenes with real-time inference capabilities. However, the requirement for stereo input is a drawback, as it complicates training and may limit applicability. GAM-Depth [19] is the latest advancement that addresses issues of inconsistent depth estimation in textureless areas and depth discrepancies at object boundaries exploiting semantics. GAM-Depth introduces a gradient-aware mask and semantic constraints, enhancing supervision in key areas and improving depth accuracy at object boundaries.

A similar approach to our work is presented by Castro et al. [20]. Their method also utilizes a U-Net architecture followed by a refinement module to complete depth maps obtained from low-cost RGB-D sensors. However, they employ a loss function based on the Euclidean distance transform, which contrasts with our approach, as we focus on integrating contextual information from the RGB image during the depth completion process, leading to visually coherent results. Moreover, our method directly incorporates depth maps from a more advanced RGB-D sensor (Kinect v2) in the training process, which are not explicitly used in Castro et al.'s framework.

2.2. Visual Odometry and SLAM with Scene Depth Estimation

Visual odometry and SLAM are critical for the autonomous navigation of indoor robots. However, achieving accurate and robust localization with these approaches requires relatively high hardware costs, such as LiDAR, stereo cameras, or modern RGB-D sensors. Whereas in the classic approach to VO, single RGB images are often used [1], this monocular approach requires sophisticated algorithms to estimate the transformations between the consecutive frames from the corresponding sets of 2D points determined in the images [21]. However, monocular depth estimation integration into VO and SLAM systems presents a cost-effective alternative that simplifies hardware requirements and expands application scenarios [22].

Several studies have explored the fusion of monocular depth estimation with VO and SLAM frameworks. For example, Ref. [23] developed a method that combines depth prediction from a monocular camera with direct SLAM, enhancing the system's robustness in texture-less and dynamic environments. This approach uses a CNN to predict depth and uncertainty, which are then integrated into a direct SLAM pipeline to improve pose estimation accuracy.

Similarly, DeepVO [24] incorporates deep learning for depth estimation and visual odometry, achieving end-to-end learning of camera trajectories from monocular sequences. This method highlights the potential of using deep learning to jointly learn depth and motion estimation, thereby providing a unified framework for monocular SLAM.

Further advancements have also focused on leveraging monocular depth estimation to improve loop closure detection and map optimization in SLAM systems. By using predicted depth maps to enhance feature matching and reduce drift, these systems achieve higher accuracy and reliability in indoor environments. For instance, Ref. [25] integrated depth prediction with the well-known ORB-SLAM2 visual SLAM system [26], demonstrating improved performance in challenging indoor scenarios.

Because the integration of monocular depth estimation into VO and SLAM systems represents a promising direction for enhancing indoor navigation capabilities, this paper expands on this line of research by using deep learning to enhance depth information from older sensors like Kinect v1, highlighting the importance of refining data for cost-effective VO applications.

2.3. Optimization of Parameters in Visual Odometry and SLAM

Determination of the optimal structure and parameter selection for RGB-D VO systems is challenging. We addressed this issue in [27] at the building block level. We demonstrated that an advanced back end can not compensate for the poor performance of the RGB-D VO front end. That underscores the importance of parameter optimization for achieving accurate and robust localization.

Traditional methods often rely on manual parameter tuning [28,29], involving exhaustive searches in the parameter space for feature detectors, feature matching, and outlier detection, which is inefficient and time-consuming [30].

Researchers have explored various optimization techniques to address these challenges (including evolutionary algorithms and swarm intelligence methods). Seghal et al. [31] applied a genetic algorithm to optimize parameters of the LiDAR-monocular visual odometry (LIMO) method, reducing translation errors in the well-known outdoor KITTI odometry dataset. Wang et al. [32] introduced a 3D point cloud-based SLAM method using Particle Swarm Optimization (PSO) for enhanced scan matching accuracy by aligning extracted feature points with a global map, improving position estimation on the same KITTI benchmark. We demonstrated in [30] (our previous work) that the Evolutionary Algorithm and PSO effectively optimize RGB-D VO parameters in indoor scenarios, such as feature detection thresholds and RANSAC [33] inlier ratios, enhancing overall VO system performance.

The parameter optimization algorithms in SLAM extend to other aspects, such as loop closure detection and closing [34,35]. Recently, Zhou et al. [36] presented the SAPSO-AUFastSLAM algorithm, incorporating simulated annealing PSO to improve resampling and avoid local extrema in a particle filter-based SLAM for autonomous underwater vehicles. This method refines navigation accuracy by iteratively updating feasible solutions, outperforming conventional FastSLAM.

Despite the limited number of existing papers on parameter optimization, these examples indicate that this direction is promising for practical applications.

3. RGB-D Visual Odometry and the Used RGB-D Data

3.1. Visual Odometry Pipeline

Our research system utilizes a straightforward VO pipeline, following a feature-based approach to camera tracking [1], as depicted in Figure 1. The system leverages the popular OpenCV library for most RGB-D data processing tasks. We integrated it with the population-based optimization methods for parameter tuning. In parameter optimization, the VO pipeline serves as the main component of the fitness evaluation block.

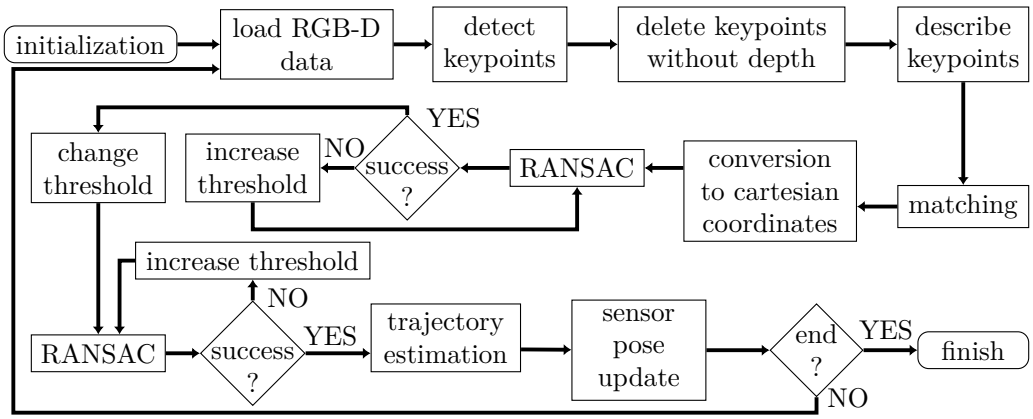

Figure 1. Block scheme of the simple VO system used in this research.

We have deliberately chosen a simple VO pipeline to ensure that advanced features, such as windowed local optimization [2,27], do not obscure the effects of depth completion or parameter optimization on the final results. This approach allows us to isolate and observe the impact of these specific elements within the VO process.

Initially, the system detects, describes, and cross-matches features using the AKAZE detector/descriptor [37], ensuring consideration of only keypoints with corresponding depth information on two consecutive frames. Discarded are points located in regions with degraded depth data. The AKAZE detector and descriptor pair have been selected based on the results of tests on representative RGB-D sequences [38].

Subsequently, the system employs a dual RANSAC filtering process to eliminate bad matches. The RANSAC procedure is applied twice with optimizable minimal distance thresholds and inliers-to-outliers ratios, as detailed in [30], using the Particle Swarm Optimization (PSO) algorithm. The PSO or an adaptive Evolutionary Algorithm (EA) are used to achieve further AKAZE detector threshold optimization [30].

Finally, the transformation between the consecutive frames from the set of corresponding 3D feature points augmented with the depth data is estimated [39], and the camera pose is updated by concatenating the frame-to-frame estimates.

3.2. Dataset Characteristics

In the experiments, we used the PUTKK [40] dataset containing eight different trajectories, recorded by Kinect v1 and v2 cameras paired and moved together (Figure 2a). The registered sequences consist of 60–2855 frames. The collection of all images from both Kinects and the motion-capture system (used for ground-truth retrieval) has been time-synchronized. A more detailed description of the dataset and the test environment can be found in [40], in which Kraft et al. demonstrate that the missing depth data areas in Kinect v1 are an essential source of problems for VO systems because of the reduced number of useful point features.

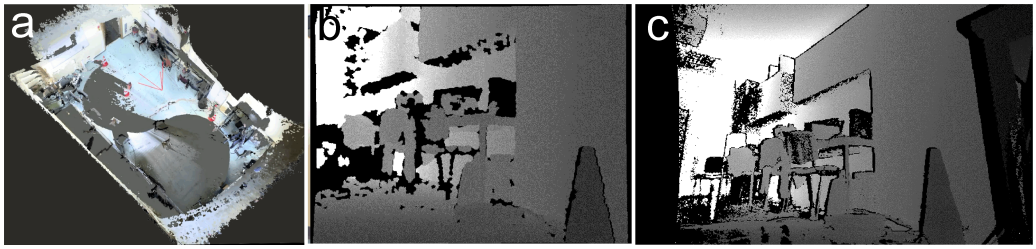

Figure 2. PUTKK dataset examples: Visualization of RGB-colored point clouds registered with the Kinect v1 and ground-truth camera poses for one of the PUTKK dataset sequences (**a**), sample Kinect v1 depth frame (**b**), and sample Kinect v2 depth frame (**c**) from this dataset.

The main difference between the two versions (despite resolution and fields of view) of the Kinect sensor is the method of collecting depth information. The earlier version used a specially crafted dot pattern, which resulted in significant no-depth areas, particularly close to edges of the scene objects (Figure 2b), versus the ToF (Time of Flight) approach present in the newer one, which provides more consistent scene depth maps for the same vantage points (Figure 2c).

4. Parameter Optimization

In our approach to parameter optimization, we have chosen Particle Swarm Optimization and an adaptive Evolutionary Algorithm due to their complementary strengths. PSO is known for its rapid convergence and simplicity, while the adaptive EA offers robust performance with minimal parameter tuning and maintains population diversity better. Although Ref. [30] compares these methods in the context of VO parameter optimization, this study introduces scene depth completion, which alters the requirements for feature detectors, potentially affecting the optimization results.

4.1. Particle Swarm Optimization

To find optimal VO parameters, we employ the Particle Swarm Optimization algorithm [41]. The algorithm (Figure 3), inspired by nature, models a "flock searching for a cornfield".

Firstly, it randomly initializes m sets of particles, each constituted by n parameters, along with their corresponding velocities. The velocities have no direct physical interpretation in the VO algorithm but control the exploration of the search space—the higher the velocity, the more significant the parameter changes between consecutive iterations.

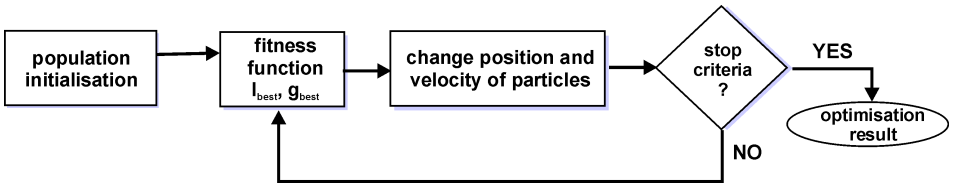

Figure 3. Block scheme of the Particle Swarm Optimization algorithm.

Next, it evaluates how well each particle meets the fitness criteria, identifying the best parameters for each particle and determining the best set of parameters globally. Having the VO pipeline in the loop of the optimization algorithm, we used the popular ATE (Absolute Trajectory Error) or RPE (Relative Pose Error) localization accuracy metrics defined in [42] to assess the performance of visual odometry, computing the root mean square error (RMSE) for both metrics. RMSE is a statistical measure that represents the square root of the average of the squared differences between predicted and actual values and is used to assess the accuracy of the predictions. The ATE requires the obtained trajectories to be aligned (to the ground-truth trajectories) and takes the difference between the estimated and ground-truth camera poses. At the same time, the RPE shows only the local differences between estimated and ground-truth trajectories. The ATE value is the Euclidean distance between the corresponding points of the estimated trajectory and the ground-truth trajectory. For the entire trajectory, the RMSE of the ATE metrics is computed. Given two trajectories (the ground-truth trajectory $\mathbf{T}^{\mathrm{gt}} = \{\mathbf{T}_1^{\mathrm{gt}}, \mathbf{T}_2^{\mathrm{gt}}, \ldots, \mathbf{T}_n^{\mathrm{gt}}\}$ and the estimated trajectory $\mathbf{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$) with the same number of n computed poses, where homogeneous matrices represent $\mathbf{T}_i$ and $\mathbf{T}_i^{\mathrm{gt}}$, we can calculate the ATE metric $\mathbf{E}_i$ for the i-th frame as follows:

$$\mathbf{E}_i^{\mathrm{ATE}} = \left(\mathbf{T}_i^{\mathrm{gt}}\right)^{-1} \mathbf{T}_i \tag{1}$$

and subsequently determine the ATE value for the entire trajectory from the RMSE of (1) for all poses. To calculate the RPE for the i-th pose, we can use the following equation:

$$\mathbf{E}_i^{\mathrm{RPE}} = \left((\mathbf{T}_i^{\mathrm{gt}})^{-1} \mathbf{T}_{i+1}^{\mathrm{gt}}\right)^{-1} \left(\mathbf{T}_i^{-1} \mathbf{T}_{i+1}\right). \tag{2}$$

From this, we can determine the relative translational error $\mathrm{RPE}_{t(i)}$ or the relative rotational error $\mathrm{RPE}_{r(i)}$ for the i-th pose by taking the translational or rotational component of $\mathbf{E}_i^{\mathrm{RPE}}$ and then computing the Euclidean norm or the Euler angle, respectively.

Based on the evaluation of the ATE or RPE metrics, the algorithm updates the parameters of each particle. Here, the ATE or RPE metrics serve as alternative fitness criteria, depending on the configuration of the PSO algorithm. As the RPE criteria separately compute values for local translation and rotation errors, we only use the translational part in the evaluation.

Equation (3) summarizes this process. The algorithm operates until it meets a stopping condition, which can include achieving a satisfactory fitness score for the best particle,

no progress in fitness value over several iterations, or reaching the maximum number of iterations:

$$\begin{aligned} \mathbf{v}_m^{i+1} &= \mathbf{v}_m^i + c_1 \cdot \text{rand}() \cdot (\mathbf{l}_{\text{best}} - \mathbf{p}_m^i) + c_2 \cdot \text{rand}() \cdot (\mathbf{g}_{\text{best}} - \mathbf{p}_m^i) \\ \mathbf{p}_m^{i+1} &= \mathbf{p}_m^i + \mathbf{v}_m^{i+1}, \end{aligned} \tag{3}$$

where $\mathbf{v}_m^i$ is the current velocity of the m-th particle $\mathbf{p}_m^i$, c_1 and c_2 are constant values (here $c_1 = c_2 = 2$), $\mathbf{l}_{\text{best}}$ is the vector of the best parameters found up to the current iteration of the algorithm for the current particle, $\mathbf{g}_{\text{best}}$ is the set of globally best parameters, $\mathbf{p}_m^{i+1}$ and $\mathbf{v}_m^{i+1}$ are the updated m-th particle in the next iteration and its velocity, respectively, while rand() denotes a function generating pseudo-random numbers. The $\mathbf{g}_{\text{best}}$ component allows the particle to take any value between $\mathbf{p}_m$ and $\mathbf{p}'_m$. The same applies to the $\mathbf{l}_{\text{best}}$ component.

For the PSO algorithm, we have chosen five parameters for optimization. The initial Euclidean error thresholds $d_{E,1}$ and $d_{E,2}$, a satisfying ratio of inliers to outliers $\Gamma_{o,1}$ and $\Gamma_{o,2}$ for the two RANSAC loops, as well as the constant AKAZE detector threshold value τ_A. The PSO algorithm randomly sets these five parameters for 40 particles and tries to find the best parameter set during a maximum of 9 iterations for the 5 parameters and 20 for the detector threshold. The choice of the number of particles was related to the maximal number of threads that can work in parallel.

The implemented PSO quickly computes the updates to the particle and has rapid convergence. Unfortunately, fast convergence can lead to the loss of diversity among the particles, resulting in premature convergence. Several solutions to this problem exist in the literature, such as the perturbed PSO algorithm [43], which introduces additional perturbation to the global best solution to maintain diversity. However, more complicated PSO variants are more time-consuming because they perform additional computations for each particle update. Hence, we stick with the canonical PSO variant.

4.2. Evolutionary Algorithm

As an alternative to the PSO algorithm, we have chosen a variant of the Evolutionary Algorithm (EA), a self-adapting algorithm inspired by natural ecosystems [44]. Contrary to typical genetic algorithms, it requires setting a minimal number of parameters. We made a few changes to adapt the algorithm described in [44] to our problem. We used real numbers (of the float type) instead of binary strings to encode the genome. Initially, we specify the maximum size of the population r and the number of individuals in the first generation, for which we draw the initial parameters (genomes).

Successive generations are created and evaluated in a loop (Figure 4) until one of the stopping criteria, which are the same as those used for PSO, is met. The individuals are assessed based on the translational part of the RPE or ATE error. In the loop, we draw individuals who will interact with other individuals while the remaining individuals undergo a mutation process. The individuals with the probability P_i (where i is the iteration number), equal to the ratio between the population size in the i-th generation and the maximum allowed population size, are drawn. Among the individuals that have to interact with the others, we draw with the same probability P_i those that will fight with others, while the remaining ones can reproduce. In the fight interaction, the stronger individual (having better fitness value) always wins, while the weaker one disappears. The reproduction is implemented by randomly exchanging parts of the genomes in a single-point crossover operation. The crossover point is drawn randomly from a normal distribution. Mutation is accomplished as the initiation of a new individual with one gene randomly changed concerning its "parent" individual, which in this action is also preserved. If any of these actions leads to exceeding the maximum population size limit, then the weakest individual from them disappears. It can be one of the parents (or a single parent in the mutation case) or a child.

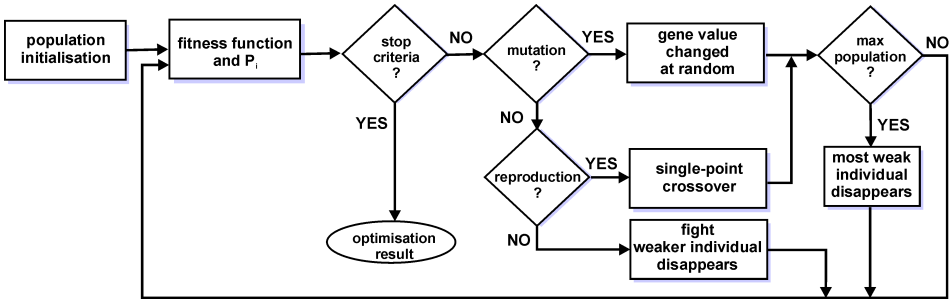

Figure 4. Block scheme of the adaptive Evolutionary Algorithm.

For a fair comparison, we set the parameters of EA to operate on a population of approximately the same size as in PSO. In our experiments, the initial number of individuals equals 10, and their maximum number is 40. We set the maximum number of iterations to 20.

5. Scene Depth Estimation Methods

5.1. Classic Inpainting Methods for Depth Estimation

One can fill the missing depth areas in Kinect v1 images using depth values estimated from the surrounding data. The Telea algorithm [45] or a method based on the Navier–Stokes equations [46] are available in the OpenCV library and can be applied for this purpose, requiring image masks highlighting the regions to be inpainted.

The Telea algorithm uses the fast marching method to select the next pixels to be inpainted. It computes a weighted average of all known points in a neighborhood defined by a user-chosen radius. The weights are set to propagate the pixel depth values and maintain the sharp details of the image.

On the other hand, the algorithm from [46] uses a fluid dynamics model based on Navier–Stokes equations to propagate the image Laplacian along the isophote directions, which are lines in the inpainting region that must be parallel to the level curves of the depth image values' smoothness.

As the no-depth areas in Kinect v2 images are relatively small, we hypothesize that inpainting can substitute missing depth areas by leveraging neighboring depth data to create a continuous and coherent depth map—this is crucial for applications that depend on accurate depth information.

5.2. Deep Learning for Scene Depth Estimation

For the purpose of deep learning-based depth estimation, we employ the Monodepth model [14], which follows the U-Net architecture by using ResNet18 with weights pre-trained on ImageNet (Figure 5). This neural network has an encoder–decoder structure with convolutional layers for the encoder and up-convolutional layers for the decoder, incorporating skip connections between corresponding layers to preserve high-resolution details. The encoder progressively downsamples the input image to capture high-level features, while the decoder upsamples these features to reconstruct the depth map at multiple scales. The loss function follows the combined loss proposed in [14], with appearance matching loss and disparity smoothness loss, with the new loss component related to the Kinect v2 ground-truth depth images for fine-tuning on the PUTKK dataset.

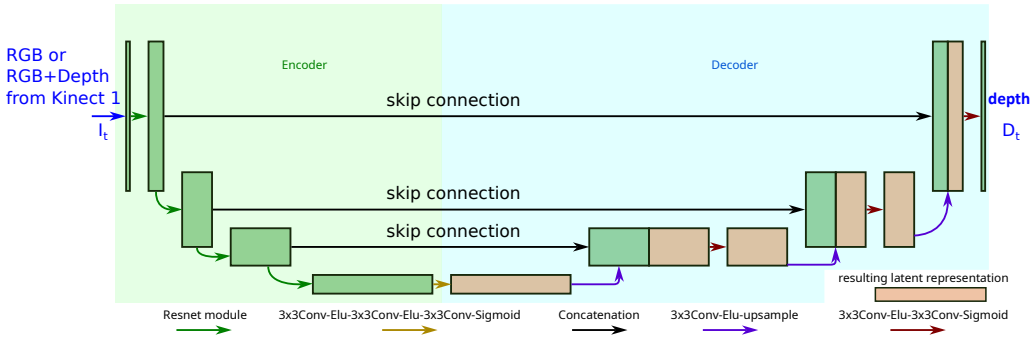

Figure 5. The U-Net architecture of a CNN is based on the Monodepth network, which is used for depth completion with RGB or RGB-D input from Kinect v1.

This open-source neural network model suits our purpose of enhancing visual odometry with estimated scene depth. That is due to its self-supervised learning approach, which eliminates the need for extensive labeled depth data. Key improvements, such as minimum reprojection loss to handle occlusions and a full-resolution multi-scale sampling method to reduce artifacts, ensuring superior depth estimation. These features, combined with its simplicity and effectiveness, make Monodepth a robust and efficient choice for integrating accurate depth information into visual odometry. Moreover, the Monodepth architecture is reliable in dynamic and unstructured environments and efficient in processing and memory usage on edge devices [47], which makes it a reasonable choice for practical usage in low-cost indoor mobile robots.

We tested the Monodepth model in three variants that differ in the method of their training and the input data used at the inference (depth prediction) stage:

- The original open-source model, with the weights trained by its authors, was used to predict scene depth images from Kinect v1 RGB frames.
- The model was fine-tuned on a PUTKK dataset sequence using transformed Kinect v2 depth frames as ground truth in the loss function and used to predict scene depth images from Kinect v1 RGB frames. We call this variant learned depth with RGB inference.
- The model was fine-tuned on a PUTKK dataset sequence using transformed Kinect v2 depth frames as ground truth in the loss function and Kinect v1 RGB and depth frames as input. This model is then used to predict scene depth images from pairs of Kinect v1 RGB and depth frames. We call this variant learned depth with RGB-D inference.

We used the FastAi v1 framework [48] to fine-tune this network with Euclidean RMSE error with respect to the Kinect v2 ground-truth images as a new component of the loss function. We changed the first layer of the Monodepth network to accept 4-channel RGB-D input instead of standard 3-channel RGB images.

For training, we used images from the PUTKK dataset: Kinect v1 as input and Kinect v2 (appropriately transformed to the viewpoint of Kinect v1) as ground truth. To convert the data from the Kinect v2, we used camera calibration data, distortion parameters, and a depth factor coefficient (informing how many distance units equal 1 m) to transform RGB-D data into a point cloud. Then, we used the known roto-translation matrix between the Kinect v2 and Kinect v1 sensors mounted to the cart used for acquiring the dataset [40] to transform the Kinect v2 point cloud to the view of Kinect v1 and further into an image (by reversing the process used to create point clouds). Because Kinect v2 has a larger field of view, after projection, some points can be located beyond the Kinect v1 field of view. These points are not considered for further processing. Because Kinect v2 has a higher resolution, more than a single pixel of its depth image can represent a corresponding pixel in the paired Kinect v1 depth image. Therefore, we interpolate the depth values between neighboring pixels of the Kinect v1 depth image. For each Kinect v2 depth value falling into a given pixel of the Kinect v1 image, we assign a weight equal to its absolute distance

from the center of this pixel. The final depth value of the interpolated Kinect v1 pixel is the weighted sum of all Kinect v2 depths falling in this pixel, multiplied by a depth factor. This way, we created a ground-truth depth image containing the Kinect v2 depth data but compatible with the format of the Kinect v1 depth image. We used a depth factor in which 1 m equals 5000 units, which allows us to save an image in uint16 format, retaining higher precision in the reliable camera measurement distance.

For the new loss component, we calculate the D_{RMSE} as an equally weighted average of the non-zero differences between the pixels from the neural network inference and the (synthesized) ground-truth depth image (4):

$$D_{\mathrm{RMSE}} = \sqrt{\frac{1}{n}\sum_{uv}(I_{uv} - G_{uv})^2}, \tag{4}$$

where n is the total count of non-zero pixels in the Kinect v2 converted image, D_{RMSE} is the root mean square error, I_{uv} represents an inference image pixel at the uv coordinates, and G_{uv} is its ground-truth counterpart.

At the beginning of the fine-tuning procedure, we learned only the last few classifier layers for the first few cycles and then all the layers. After every few cycles, we saved weights and ran the learning rate finder (FastAi) procedure to determine further fine-tuning. Figure 6 depicts the fine-tuning procedure as a block scheme.

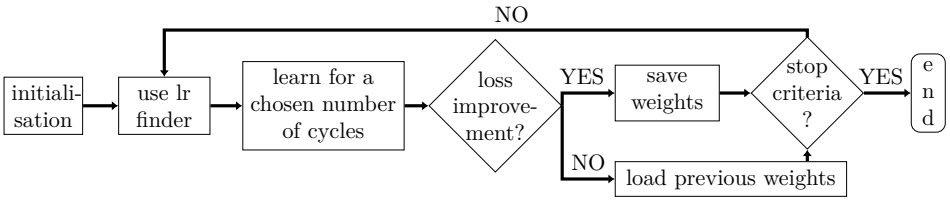

Figure 6. Block scheme of the fine-tuning procedure for the Monodepth model.

To augment data, we have chosen only affine (horizontal and vertical) transformations and rotation by 90°, as other techniques had a negative effect. Then, we used discriminative layer training: we learned the layers on the bottom of the model with lower learning rates than those on the top, following the geometrical progress for the layers in between. The FastAi library [48] used for optimization of the learning process implements, with few changes, the learning rate policy proposed in [49,50]. For each group, we increase the learning rate from $lr_{\mathrm{max}}/\mathrm{div}_{\mathrm{factor}}$ (default $\mathrm{div}_{\mathrm{factor}}$ equals 25) to lr_{max}, after which we perform a cosine annealing from lr_{max} to 0. One should pick the values before the minimum for the top layers ($lr_{max\_top\_layers}$) and at least one order of magnitude lower for the bottom layers ($lr_{max\_bottom\_layers}$). Figure 7 depicts the results of the learning rate search process (left) and the learning results of RGB-D-based fine tuning with the PUTKK dataset (right).

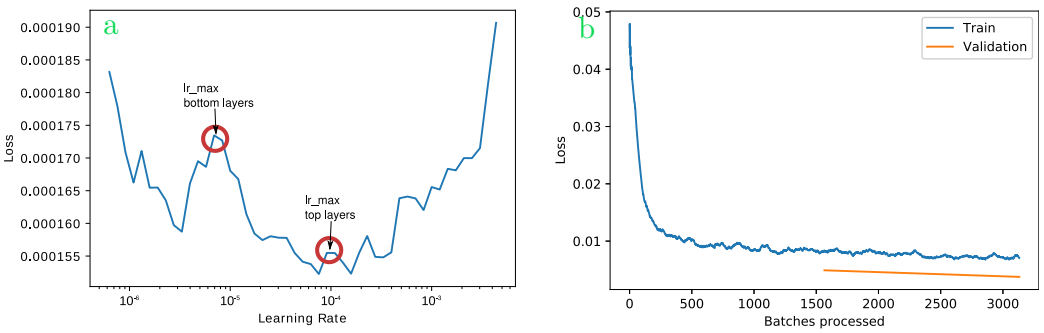

Figure 7. Search method for the best learning rate with FastAi (**a**) and learning results (**b**).

6. Experiments and Results

6.1. Selection of the Parameter Optimization Method

We have performed all experiments presented in this paper using the PUTKK dataset sequences. Investigating the role of the parameters, we leveraged our experience from [30], and we used only PSO to optimize the RANSAC parameters, as this approach yielded better results. However, for optimization of the AKAZE detector parameters, we tried both PSO and EA because the results from [30] did not show a clear winner. Therefore, we first optimized jointly all parameters using PSO, alternatively with the ATE or RPE fitness metric (Table 1).

Table 1. Visual odometry parameters that were optimized jointly using PSO.

Method	$d_{E,1}$	$\Gamma_{o,1}$	$d_{E,2}$	$\Gamma_{o,2}$	τ_A
PSO ATE	0.060	0.890	0.068	0.892	0.00001
PSO RPE	0.048	0.805	0.043	0.800	0.00001

Table 2 shows the accuracy of the *putkk_Dataset_5_Kin_1* trajectory estimation in terms of the RMSE ATE and RPE metrics for the parameter sets from Table 1.

Table 2. ATE and RPE values on the *putkk_Dataset_5_Kin_1* sequence for different parameter sets.

Method/Params	ATE RMSE [m]	Trans. RPE RMSE [m]	Rot. RPE RMSE [°]
PSO ATE	0.048	0.030	0.675
PSO RPE	0.056	0.029	0.643

Next, we alternatively used the PSO or EA algorithms, with the ATE or the RPE metric, to further optimize only the detector threshold using the RANSAC parameters obtained from the PSO optimization with ATE criteria, which we consider slightly better. Using PSO with either ATE or RPE, this process finished after 9 iterations, reaching, in both cases, the minimal value of τ_{Amin} = 0.00010, whereas with EA, the optimization took 20 iterations, and we obtained 0.000034 for EA with ATE and 0.000104 for EA with RPE. Table 3 shows the accuracy of the *putkk_Dataset_5_Kin_1* trajectory estimation in the sense of the RMSE ATE and RPE metrics for the parameter sets augmented by the AKAZE detector thresholds updated in the second stage of the optimization procedure.

Table 3. Comparison of ATE and RPE results on the *putkk_Dataset_5_Kin_1* sequence for parameter sets obtained with different optimization procedure variants of the AKAZE detector threshold.

		putkk_Dataset_5_Kin_1			
Error Metric		**PSO ATE**	**PSO RPE**	**EA ATE**	**EA RPE**
ATE RMSE	[m]	0.049	0.056	0.070	0.072
Trans. RPE RMSE	[m]	0.030	0.029	0.035	0.033
Rot. RPE RMSE	[°]	0.665	0.643	0.794	0.745

We obtained the best results (most accurate estimated trajectories) by applying the PSO optimization. As expected, the PSO ATE optimization resulted in a set of parameters that produced slightly better ATE RMS values, while the PSO RPE variant resulted in a trajectory showing slightly better RPE RMSE scores. As the ATE metric better reflects the global accuracy of the estimated trajectory, we prefer parameters resulting in better ATE RMSE. Therefore, the PSO ATE results (parameters) are used with the depth maps (completed with either inpainted depth values or depth values predicted by the learned neural model). For the classic methods from OpenCV, we only show results for the best inpainting radius.

6.2. Comparison of the Depth Completion Methods and Their Performance in VO

We started the investigation of the depth completion methods by comparing the conventional inpainting methods with the proposed deep learning depth estimation methods. We used *putkk_Dataset_5* sequences for choosing the best inpainting radius for the standard algorithms and for parameter optimization of the VO system using enhanced depth information. Parameters of the VO pipeline with enhanced depth frames were optimized using the approach described in Section 5.

We have used inpainted and learned depth maps to fill the no-depth areas in original Kinect v1 depth images. For visual assessment of the enhanced depth frames, see Figure 8. Depth images inpainted with the conventional algorithms (Telea or Navier–Stokes, Figure 8b,c) do not differ much. They do not restore most of the thin chair legs and are blurry in some areas. Objects in the depth images completed with depth values learned using Kinect v1 RGB frames for inference (Figure 8d) have smooth boundaries, present more plausible chair legs, and are overall more eye-pleasing, even though we could not reconstruct all of the visible details in the RGB images.

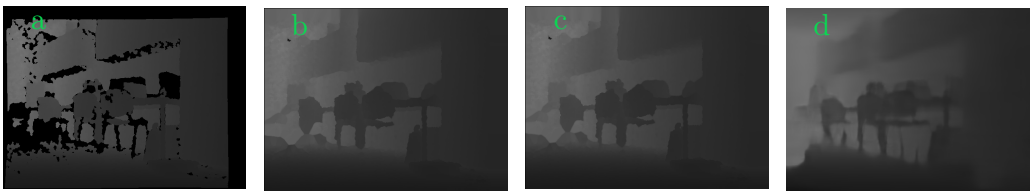

Figure 8. Depth maps: (**a**) Original, (**b**) Navier–Stokes (NS), (**c**) Telea, and (**d**) learned with RGB-D frames.

We have also compared the qualitatively different approaches to depth prediction using the Monodepth neural network model (Figure 9). This comparison aimed to determine if the original open-source model, with its pre-learned weights, can produce helpful scene depth estimates for the PUTKK scenes. Figure 9b shows a sample result, constituting evidence that the Monodepth model does not suit our purposes without further fine-tuning on the target dataset. In contrast, the depth frame estimated using the fine-tuned neural network and only Kinect v1 RGB images for inference (Figure 9c) is visually plausible. Therefore, we used only the fine-tuned Monodepth model in further research.

Figure 9. Kinect v1 depth maps: (**a**) Original, (**b**) estimated by the original Monodepth model, and (**c**) original depth image completed by learned depth with RGB inference.

While the visual comparison of frames with estimated scene depth encouraged us to use the Monodepth model in our VO system, the qualitative results needed to be verified by further quantitative results. To this end, we looked closely at the estimated depth values, directly comparing the simple fine-tuned model using only RGB images for inference and the more elaborated model that takes a pair of RGB and depth images from Kinect v1 as input. In Figure 10, we compare the estimated depth errors concerning a reference Kinect v2 depth map for these two Monodepth model variants. For visualization purposes, we have normalized (separately for each of the images) the estimated depth values to the $< 0, 1 >$ range and used a colormap to show the too-short depth estimates as those from 0 (blue) to 0.5, values matching the Kinect v2 reference depth as close to 0.5, and too long

estimates as those towards 1 (red). The regions without Kinect v2 depth information are black in both images.

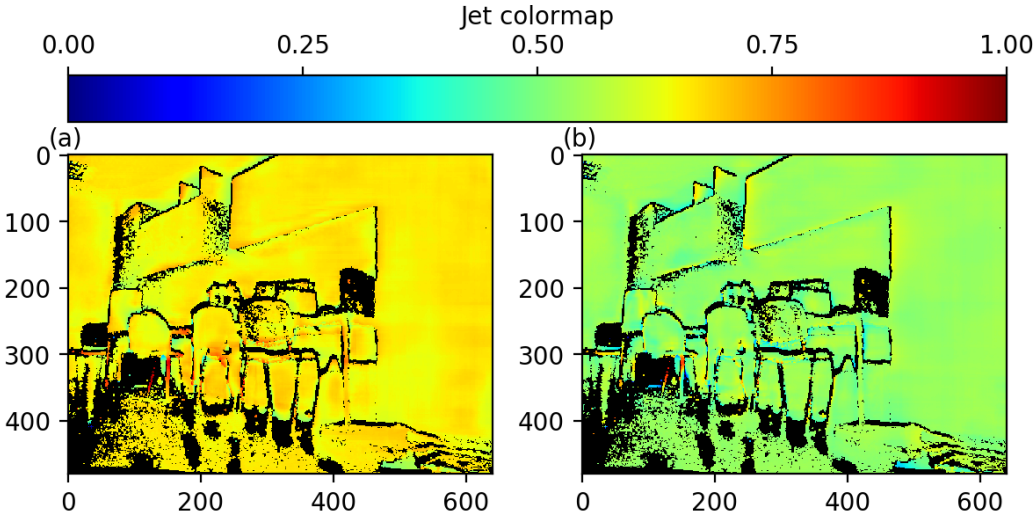

Figure 10. Colormap visualization of the difference between the estimated scene depth and the Kinect v2 ground-truth for the improved Monodepth model inference with Kinect v1 RGB frames only (**a**) and with both RGB and depth frames from Kinect v1 (**b**). See text for further explanation.

As one can see, the variant using only RGB images to infer the depth tends to overestimate the depth (Figure 10a). On the contrary, the version we propose in this paper, which jointly uses RGB and depth images from Kinect v1 for inference, has more correct depth values. Most notable differences are visible in the sensitive regions around the chair legs and edges.

These results are also validated by an experiment that shows the performance of different approaches to scene depth completion in an end-to-end manner, pairing the investigated methods with our VO pipeline and optimized parameters (Table 4). The variant of the learned depth used here is the one with a fine-tuned model and RGB-D inference. In particular, one can see the superiority of the variant with learned depth concerning the ATE RMSE value.

Table 4. Trajectory estimation results for Kinect v1 frames with no depth inpainting, Telea and Navier–Stokes inpainting (radius 3), and learned depth with optimized parameters for the *putkk_Dataset_5_Kin_1* sequence.

		putkk_Dataset_5_Kin_1			
Metric		**No Inpainting**	**NS**	**Telea**	**Learned RGB-D Inference**
ATE RMSE	[m]	0.198	0.201	0.204	0.049
Trans. RPE RMSE	[m]	0.012	0.011	0.012	0.010
Rot. RPE RMSE	[°]	0.174	0.187	0.183	0.665

To verify how this result generalizes to other sequences from the PUTKK dataset, we present results for three other sequences from this dataset in Table 5. The verification experiment shows that using enhanced depth information can sometimes worsen the results—which is the case for some versions using depth images inpainted by conventional methods. The RPE results are generally worse for the versions with modified images than for the original Kinect v1 frames. That might be due to specific scene characteristics differing from the view observed in the sequence used for fine-tuning the depth prediction model. Figure 11 shows the ATE plots (black lines represent the ground-truth trajectory,

blue lines are the estimated trajectories, and the red segments indicate the Euclidean errors) and translational RPE plots for *putkk_Dataset_1*.

Table 5. Trajectory estimation results with conventional and deep learning methods for three PUTKK sequences (not used for training or optimization of parameters).

		putkk_Dataset_1_Kin_1			
Metric		**No Inpainting**	**NS**	**Telea**	**Learned RGB-D Inference**
ATE RMSE	[m]	0.596	1.112	0.443	0.359
Trans. RPE RMSE	[m]	0.009	0.021	0.016	0.015
Rot. RPE RMSE	[°]	0.168	0.507	0.369	0.375
		putkk_Dataset_2_Kin_1			
ATE RMSE	[m]	0.677	1.148	0.616	0.502
Trans. RPE RMSE	[m]	0.010	0.030	0.033	0.020
Rot. RPE RMSE	[°]	0.243	0.713	0.694	0.446
		putkk_Dataset_3_Kin_1			
ATE RMSE	[m]	1.145	0.934	1.092	0.773
Trans. RPE RMSE	[m]	0.012	0.033	0.030	0.012
Rot. RPE RMSE	[°]	0.251	0.607	0.532	0.212

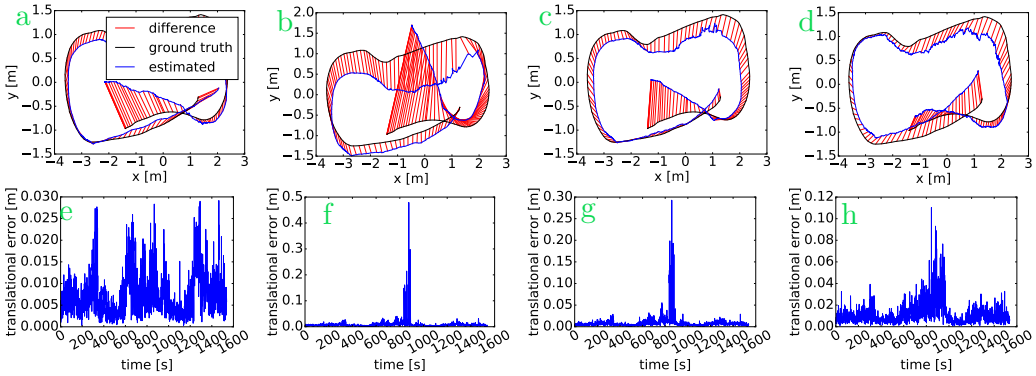

Figure 11. Trajectory estimation results for the *putkk_Dataset_1_Kin_1* sequence for our VO system working with: (**a**,**e**) no inpainting, (**b**,**f**) NS inpainting, (**c**,**g**) Telea inpainting, or (**d**,**h**) learned depth with RGB-D inference. First row: ATE error plots; second: translational RPE plots.

Figures 12 and 13 display the ATE and translational RPE plots for *putkk_Dataset_2_Kin_1* and *putkk_Dataset_3_Kin_1*, using the same graphical convention. Together with Figure 11, these figures enable qualitative assessment of how much the deep learning-based approach to Kinect v1 depth image completion outperforms the classic non-learning approaches across three different datasets.

For the experimental results shown in Table 5 and the following ATE and RPE plots, we have used RANSAC parameters found during PSO ATE optimization and have further optimized only the threshold τ_A, along with PSO ATE. We used the same sequence (*putkk_Dataset_5*) for this process and obtained $\tau_A = 0.001880$ for depth maps resulting from fusing Kinect v1 depth and RGB-only inference and $\tau_A = 0.000597$ for depth images completed with RGB-D inferences. Table 6 collects ATE and RPE results for this experiment.

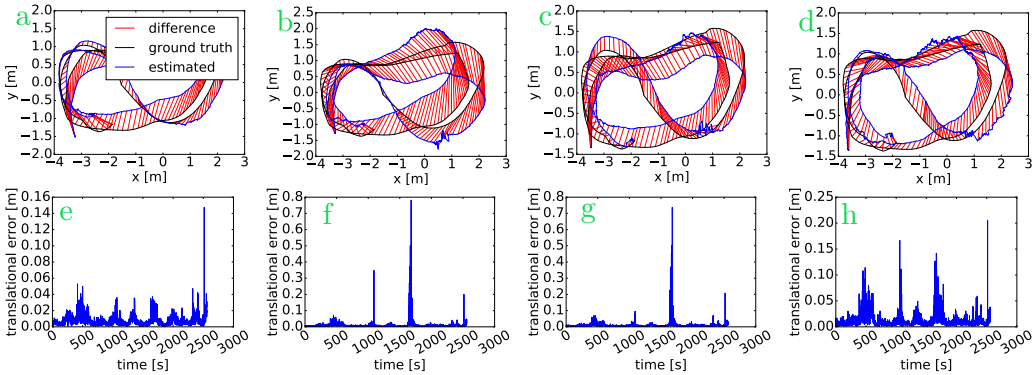

Figure 12. Trajectory estimation results for the *putkk_Dataset_2_Kin_1* sequence for our VO system working with: (**a**,**e**) no inpainting, (**b**,**f**) NS inpainting, (**c**,**g**) Telea inpainting, or (**d**,**h**) learned depth with RGB-D inference. First row: ATE error plots; second: translational RPE plots.

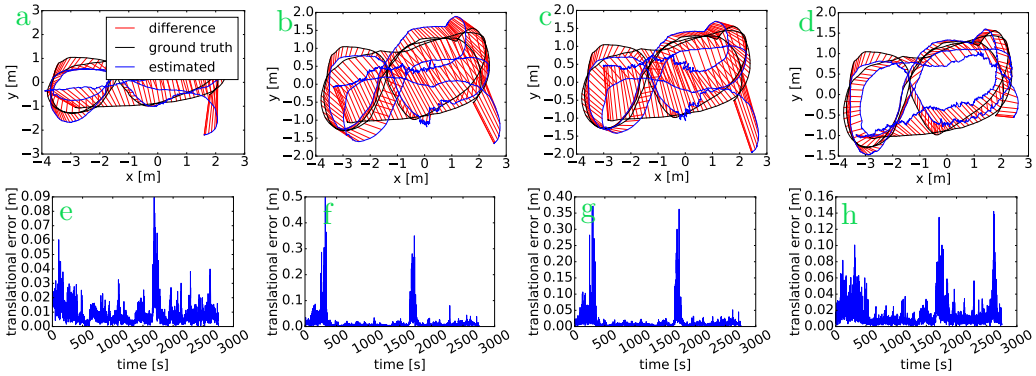

Figure 13. Trajectory estimation results for the *putkk_Dataset_3_Kin_1* sequence for our VO system working with: (**a**,**e**) no inpainting, (**b**,**f**) NS inpainting, (**c**,**g**) Telea inpainting, or (**d**,**h**) learned depth with RGB-D inference. First row: ATE error plots; second: translational RPE plots.

Table 6. Trajectory optimization results for no inpainting and two variants of learned depth with optimized parameters.

		putkk_Dataset_5_Kin_1		
Metric		**No Inpainting**	**Learned RGB Inference**	**Learned RGB-D Inference**
ATE RMSE	[m]	0.198	0.176	0.049
Trans. RPE RMSE	[m]	0.012	0.015	0.012
Rot. RPE RMSE	[°]	0.174	0.289	0.212

The joint use of RGB and depth images for inference allows us to obtain better results than those we can achieve with our simple VO for the unmodified depth images from Kinect v1 and depth images completed with depth values from the simple fine-tuned model using only RGB frames for inference. We have also used other sequences from the PUTKK dataset (same as before) to confirm this. Table 7 collects the results for the first three sequences.

The verification step shows that using enhanced depth information based on RGB-D inferences allows our VO pipeline to achieve better ATE results than when the original depth images are applied. However, RPE results are a bit worse here. The reasons for this can come from the VO parameter optimization goal, which stresses ATE, but also from the overfitting of the neural network model to the particular RGB-D sequence used for fine-tuning.

Table 7. Trajectory estimation results with different deep learning methods for three PUTKK sequences (not used for training or optimization of parameters).

		putkk_Dataset_1_Kin_1		
Metric		**No Inpainting**	**Learned RGB Inference**	**Learned RGB-D Inference**
ATE RMSE	[m]	0.596	0.701	0.359
Trans. RPE RMSE	[m]	0.009	0.022	0.015
Rot. RPE RMSE	[°]	0.168	0.535	0.375
		putkk _Dataset_2_Kin_1		
ATE RMSE	[m]	0.677	1.122	0.502
Trans. RPE RMSE	[m]	0.010	0.027	0.020
Rot. RPE RMSE	[°]	0.243	0.593	0.446
		putkk_Dataset_3_Kin_1		
ATE RMSE	[m]	1.145	0.886	0.773
Trans. RPE RMSE	[m]	0.012	0.027	0.019
Rot. RPE RMSE	[°]	0.251	0.591	0.407

The use of RGB-only inference for Kinect v1 depth inpainting does not allow for achieving the same performance as its RGB-D-based counterpart, but we can still see improvement in some areas. Nonetheless, we recommend the proposed RGB-D-based neural network model, as it uses whole information from Kinect v1 and the correlations between the RGB and depth images in the same sensor frame.

To further verify the generalization of our method, we also show Figures 14 and 15 with corresponding ATE plots and RPE plots for the *putkk_Dataset_2* and *putkk_Dataset_3* sequences, respectively. One can see from the ATE plots that the application of our depth completion method with RGB-D-based inference from the fine-tuned neural network outperforms the simpler variant with RGB-only inference.

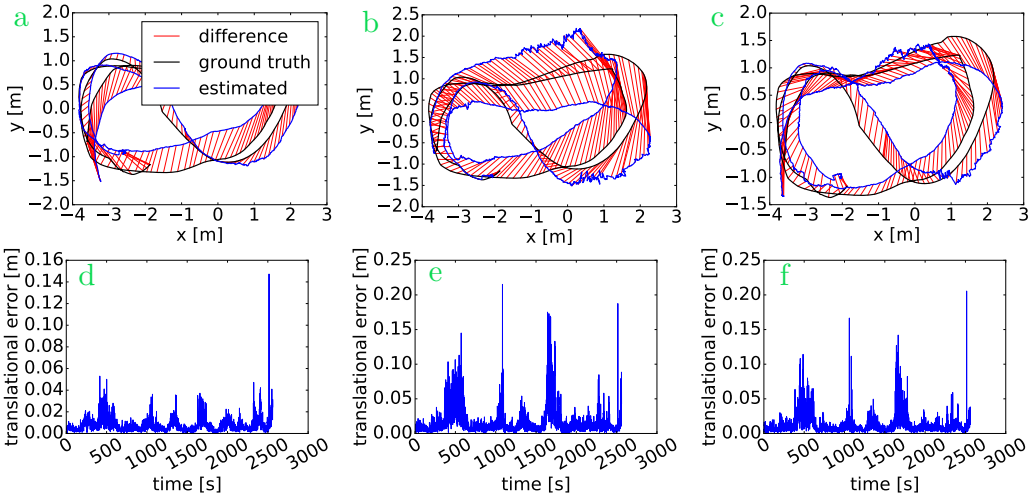

Figure 14. Trajectory estimation results for Kinect v1 frames on the *putkk_Dataset_2_Kin_1* sequence for VO system working with (**a**,**d**) no inpainting, (**b**,**e**) learned depth completion with RGB inference, and (**c**,**f**) learned depth completion with RGB-D inference. The first row presents ATE error plots. The second includes translational RPE plots.

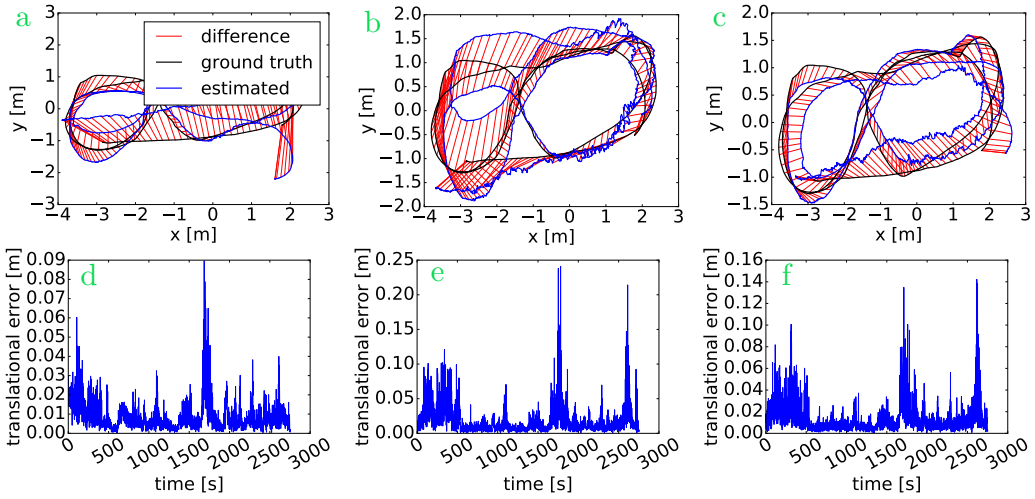

Figure 15. Trajectory estimation results for Kinect v1 frames on the *putkk_Dataset_3_Kin_1* sequence for VO system working with (**a**,**d**) no inpainting, (**b**,**e**) learned depth completion with RGB inference, and (**c**,**f**) learned depth completion with RGB-D inference. The first row presents ATE error plots. The second includes translational RPE plots.

6.3. Computation Efficiency

Depth completion is an additional step in the Kinect sensor data processing in the proposed VO pipeline. Therefore, it is interesting to assess its impact on the processing time and the achieved frame rate of the VO system. Table 8 provides execution times for the depth completion procedure of the three methods: Telea [45], Navier–Stokes-based [46], and our neural network in the RGB-D mode for the *putkk_Dataset_5_Kin_1* sequence. The values are the average over all frames of this sequence. These results were obtained on a computer with an i7-6820HQ CPU at 2.7 GHz and an NVIDIA Quadro M2000M GPU under Linux. It is notable that the conventional methods running on a CPU cannot compete with the CNN inference on a GPU in terms of the time to complete the task. The additional time for depth completion is also small compared to the frame processing time in the VO pipeline, which runs at about 15 fps (frames per second) when raw Kinect v1 depth is used (no inpainting).

Table 8. Execution time results for different depth completion methods on the *putkk_Dataset_5_Kin_1* sequence.

Depth Completion Time	NS	Telea	Learned RGB-D Inference
Mean value [ms]	221.0	220.9	6.5
Standard deviation [ms]	33.3	33.3	1.9

7. Conclusions

We have presented a novel approach for enhancing RGB-D visual odometry through depth completion. This application of RGB-D depth completion has received limited attention compared to other applications such as 3D reconstruction or augmented reality. We demonstrated the effectiveness of using depth maps from the Kinect v2 to train a model for completing depth maps from the Kinect v1, showcasing the potential of cross-device training to improve the navigation capabilities of affordable indoor mobile robots. Our experiments show that deep learning allows for better results in the VO task than with the classic inpainting approaches that do not explore the RGB image context. Training the learned model jointly with the depth and RGB images and using a better, more complete depth image as ground truth for the loss function enables the model to infuse dependencies between the appearance of the objects and the no-depth areas in the Kinect v1 depth maps. While using only inference data can be beneficial, it is a further fusion of those data and

the original one that outperforms other approaches because it allows for achieving better ATE metric scores than the original data. The similar use of RGB-inference data is not as beneficial but is still worth developing in the case of limited-depth information.

Furthermore, we employed population-based optimization for parameter adaptation, a technique not commonly used in RGB-D-based localization systems. That technique allowed us to fine-tune our model parameters more effectively than conventional methods, leading to better adaptation to different scenarios.

Our experimental results highlight the improvements in depth map quality and visual odometry accuracy achieved by the deep learning-based approach. An issue in the proposed approach that needs further investigation is that the deep learning model overfits (to a given environment) and may not be suitable for different ones. Also, the keypoint detector parameters found using PSO/EA with the enhanced data differ from those obtained with original depth information, allowing for more keypoint detection. Such parameters also do not generalize well across different scenes. Despite the open issues of generalization, by addressing the specific challenges of missing and noisy depth data in Kinect-like inputs, we contribute to the field of visual odometry, opening new possibilities for more reliable and precise navigation in practical robotic applications based on budget sensors.

Author Contributions: Conceptualization, A.K. and P.S.; methodology, P.S.; software, A.K.; validation, A.K.; formal analysis, P.S.; investigation, A.K.; resources, P.S.; data curation, A.K.; writing—original draft preparation, A.K. and P.S.; writing—review and editing, P.S. and A.K.; visualization, A.K.; supervision, P.S.; project administration, P.S.; funding acquisition, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Poznań University of Technology, internal grant number 0214/SBAD/0248. The APC was funded by Poznań University of Technology.

Data Availability Statement: Open source code is available on GitHub: https://github.com/VVilk/RGBDVisualOdometryParticleSwarmoptimisation, accessed on 30 May 2024. The PUTKK dataset used is available on the project's web page (http://lrm.put.poznan.pl/putkk/, accessed on 30 May 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fraundorfer, F.; Scaramuzza, D. Visual Odometry: Part I the first 30 years and fundamentals. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92. [CrossRef]
2. Fraundorfer, F.; Scaramuzza, D. Visual Odometry: Part II - Matching, Robustness, and Applications. *IEEE Robot. Autom. Mag.* **2012**, *19*, 78–90. [CrossRef]
3. De La Puente, P.; Bajones, M.; Reuther, C.; Wolf, D.; Fischinger, D.; Vincze, M. Robot Navigation in Domestic Environments: Experiences Using RGB-D Sensors in Real Homes. *J. Intell. Robot. Syst.* **2019**, *94*, 455–470. [CrossRef]
4. Halmetschlager-Funek, G.; Suchi, M.; Kampel, M.; Vincze, M. An Empirical Evaluation of Ten Depth Cameras: Bias, Precision, Lateral Noise, Different Lighting Conditions and Materials, and Multiple Sensor Setups in Indoor Environments. *IEEE Robot. Autom. Mag.* **2019**, *26*, 67–77. [CrossRef]
5. Atapour-Abarghouei, A.; Breckon, T.P. Dealing with Missing Depth: Recent Advances in Depth Image Completion and Estimation. In *RGB-D Image Analysis and Processing*; Springer: Cham, Switzerland, 2019; pp. 15–50.
6. Richardt, C.; Stoll, C.; Dodgson, N.A.; Seidel, H.P.; Theobalt, C. Coherent Spatiotemporal Filtering, Upsampling and Rendering of RGBZ Videos. *Comput. Graph. Forum* **2012**, *31*, 247–256. [CrossRef]
7. Chen, C.; Cai, J.; Zheng, J.; Cham, T.J.; Shi, G. Kinect Depth Recovery Using a Color-Guided, Region-Adaptive, and Depth-Selective Framework. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 1–19. [CrossRef]
8. Atapour-Abarghouei, A.; de La Garanderie, G.P.; Breckon, T.P. Back to Butterworth—A Fourier basis for 3D surface relief hole filling within RGB-D imagery. In Proceedings of the 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 2813–2818.
9. Eigen, D.; Puhrsch, C.; Fergus, R. Depth map prediction from a single image using a multi-scale deep network. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2366–2374.
10. Zhang, Y.; Funkhouser, T. Deep Depth Completion of a Single RGB-D Image. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 175–185.

11. Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; Huang, T.S. Generative Image Inpainting with Contextual Attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 5505–5514.
12. Khan, F.; Salahuddin, S.; Javidnia, H. Deep Learning-Based Monocular Depth Estimation Methods—A State-of-the-Art Review. *Sensors* **2020**, *20*, 2272. [CrossRef] [PubMed]
13. Masoumian, A.; Rashwan, H.A.; Cristiano, J.; Asif, M.S.; Puig, D. Monocular Depth Estimation Using Deep Learning: A Review. *Sensors* **2022**, *22*, 5353. [CrossRef]
14. Godard, C.; Aodha, O.M.; Firman, M.; Brostow, G. Digging Into Self-Supervised Monocular Depth Estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3827–3837.
15. Xian, C.; Zhang, D.; Dai, C.; Wang, C.C.L. Fast Generation of High-Fidelity RGB-D Images by Deep Learning With Adaptive Convolution. *IEEE Trans. Autom. Sci. Eng.* **2021**, *18*, 1328–1340. [CrossRef]
16. Senushkin, D.; Romanov, M.; Belikov, I.; Konushin, A.; Patakin, N. Decoder Modulation for Indoor Depth Completion. *arXiv* **2021**, arXiv:2005.08607.
17. Wang, J.; Huang, Q. Depth Map Super-Resolution Reconstruction Based on Multi-Channel Progressive Attention Fusion Network. *Appl. Sci.* **2023**, *13*, 8270. [CrossRef]
18. Wu, C.Y.; Wang, J.; Hall, M.; Neumann, U.; Su, S. Toward Practical Monocular Indoor Depth Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 11–17 October 2021; pp. 3804–3814.
19. Cheng, A.; Yang, Z.; Zhu, H.; Mao, K. GAM-Depth: Self-Supervised Indoor Depth Estimation Leveraging a Gradient-Aware Mask and Semantic Constraints. *arXiv* **2024**, arXiv:2402.14354.
20. Castro, A.R.; Grassi, V., Jr.; Ponti, M.A. Deep Depth Completion of Low-cost Sensor Indoor RGB-D using Euclidean Distance-based Weighted Loss and Edge-aware Refinement. In Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022), Vienna, Austria, 6–8 February 2022; pp. 204–212.
21. Kostusiak, A. Frame-to-Frame Visual Odometry: The Importance of Local Transformations. In Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017, Polanica Zdroj, Poland, 22–24 May 2017; Volume 578, pp. 357–366.
22. Kostusiak, A. Improving RGB-D Visual Odometry with Depth Learned from a Better Sensor's Output. In Proceedings of the Progress in Polish Artificial Intelligence Research 4, Lodz, Poland, 24–26 April 2023; pp. 429–434.
23. Tateno, K.; Tombari, F.; Laina, I.; Navab, N. CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6565–6574.
24. Wang, S.; Clark, R.; Wen, H.; Trigoni, N. DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2024; pp. 2043–2050.
25. Tang, K.; Yuan, J.; Sun, Q.; Zhang, X.; Gao, H. An Improved ORB-SLAM2 with Refined Depth Estimation. In Proceedings of the IEEE International Conference on Real-time Computing and Robotics (RCAR), Irkutsk, Russia, 4–9 August 2019; pp. 885–889.
26. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
27. Belter, D.; Nowicki, M.; Skrzypczyński, P. On the Performance of Pose-Based RGB-D Visual Navigation Systems. In Proceedings of the Computer Vision ACCV 2014, Singapore, 1–5 November 2014; pp. 407–423. [CrossRef]
28. Endres, F.; Hess, J.; Engelhard, N.; Sturm, J.; Cremers, D.; Burgard, W. An evaluation of the RGB-D SLAM system. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St. Paul, MN, USA, 14–19 May 2012; pp. 1691–1696. [CrossRef]
29. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D Mapping with an RGB-D Camera. *IEEE Trans. Robot.* **2014**, *30*, 177–187. [CrossRef]
30. Kostusiak, A.; Skrzypczynski, P. On the Efficiency of Population-Based Optimization in Finding Best Parameters for RGB-D Visual Odometry. *J. Autom. Mob. Robot. Intell. Syst.* **2019**, *13*, 5–14. [CrossRef]
31. Sehgal, A.; Singandhupe, A.; La, H.M.; Tavakkoli, A.; Louis, S.J. Lidar-Monocular Visual Odometry with Genetic Algorithm for Parameter Optimization. In Proceedings of the International Symposium on Visual Computing, Lake Tahoe, NV, USA, 7–9 October 2019.
32. Wang, J.; Fujimoto, Y. High Accuracy Real-Time 6D SLAM with Feature Extraction Using a Neural Network. *IEEJ J. Ind. Appl.* **2021**, *10*, 512–519. [CrossRef]
33. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
34. Ayoppan, A. A Genetic Algorithm with Online Learning Approach for Improving Loop Closure Detection of a Visual SLAM. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 159–166. [CrossRef]
35. Han, D.; Li, Y.; Song, T.; Liu, Z. Multi-Objective Optimization of Loop Closure Detection Parameters for Indoor 2D Simultaneous Localization and Mapping. *Sensors* **2020**, *20*, 1906. [CrossRef]

36. Zhou, L.; Wang, M.; Zhang, X.; Qin, P.; He, B. Adaptive SLAM Methodology Based on Simulated Annealing Particle Swarm Optimization for AUV Navigation. *Electronics* **2023**, *12*, 2372. [CrossRef]
37. Alcantarilla, P.; Nuevo, J. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In Proceedings of the British Machine Vision Conference, Bristol, UK, 9–13 September 2013; BMVA Press: Durham, UK, 2013. [CrossRef]
38. Kostusiak, A. The Comparison of Keypoint Detectors and Descriptors for Registration of RGB-D Data. In Proceedings of the Challenges in Automation, Robotics and Measurement Techniques, Warsaw, Poland, 2–4 March 2016; Springer: Cham, Switzerland, 2016; pp. 609–622.
39. Umeyama, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 376–380. [CrossRef]
40. Kraft, M.; Nowicki, M.; Schmidt, A.; Fularz, M.; Skrzypczynski, P. Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect. *Mach. Vis. Appl.* **2017**, *28*, 61–74. [CrossRef]
41. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95 Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
42. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A Benchmark for the Evaluation of RGB-D SLAM Systems. In Proceedings of the International Conference on Intelligent Robot Systems (IROS), Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 573–580. [CrossRef]
43. Xinchao, Z. A perturbed particle swarm algorithm for numerical optimization. *Appl. Soft Comput.* **2010**, *10*, 119–124. [CrossRef]
44. Annunziato, M.; Pizzuti, S. Adaptive parameterization of evolutionary algorithms driven by reproduction and competition. In Proceedings of the European Symposium on Intelligent Techniques (ESIT 2000), Aachen, Germany, 14–15 September 2000; pp. 325–329.
45. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graph. Tools* **2004**, *9*, 23–34. [CrossRef]
46. Bertalmio, M.; Bertozzi, A.; Sapiro, G. Navier-Stokes, fluid dynamics, and image and video inpainting. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1. [CrossRef]
47. Feng, C.; Zhang, C.; Chen, Z.; Hu, W.; Ge, L. Real-time Monocular Depth Estimation on Embedded Systems. *arXiv* **2024**, arXiv:2308.10569.
48. Howard, J.; Gugger, S. FastAi: A Layered API for Deep Learning. *Information* **2020**, *11*, 108. [CrossRef]
49. Smith, L.N. Cyclical Learning Rates for Training Neural Networks. *arXiv* **2017**, arXiv:1506.01186.
50. Smith, L.N. A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay. *arXiv* **2018**, arXiv:1803.09820.