# An Approach to Location Estimation and Navigation Support Aided by Depth Estimation Map

**F. Geilson de L. Xavier** [*] **Suane Pires P. da Silva** [**,****]
**J. Jerovane da C. Nascimento** [***,****]
**Pedro Pedrosa R. Filho** [*,**,***,****]

[*] *Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), Universidade Federal do Ceará (UFC), Fortaleza/CE, (e-mail: geilsonxavier@gmail.com)*
[**] *Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI), Universidade Federal do Ceará (UFC), Fortaleza/CE, (e-mail: suanepires@lapisco.ifce.edu.br)*
[***] *Programa de Pós-Graduação em Ciência da Computação (PPGCC), Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), Fortaleza/CE, (e-mail: jerovane.nascimento@lapisco.ifce.edu.br)*
[****] *Laboratório de Processamento de Imagens e Simulação Computacional (LAPISCO), Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), Fortaleza/CE, (e-mail: pedrosarf@ifce.edu.br)*

**Abstract:** Depth sensing is essential to many robotic tasks, with mapping, localization and obstacle avoidance. In small robotic platforms, weight, volume and energy consumption limitations motivate the use of depth estimation using a monocular camera instead of depth sensors. This paper propose an approach for localization of autonomous mobile robots in an indoor environment using monocular vision aided by depth estimation maps from a single RGB input image applied to the concept of Transfer Learning linked to Convolutional Neural Networks (CNNs). The performance of classifiers in estimate the location was observed and compared using of a unique configuration of RGB-D images transformed into a mosaic image. The images were combined with the descriptive power of CNNs in the following scenarios: depth captured by the Kinect sensor and depth estimation generated by the AdaBins block. With the results, the performances of the classifiers were analyzed to evaluate the proposed approach.

*Keywords:* Robotic Systems, Deep Transfer Learning, Convolutional Neural Network, RGB-D Machine Learning, Topological Maps, Indoor Environment, Computer Vision.

## 1. INTRODUCTION

Mobile Robotics applications can be found in agricultural, industrial, military, domestic environments, among others. Because of the wide variety of uses of this technology, the localization task is one of the most searched topics regarding mobile robots (Li and Shi, 2018), since this is the means to an end, i.e. search and rescue, mining, delivery of people and cargo, all these activities depend on it.

But the localization task is not just the definition of absolute position in space (Elaraby et al., 2018). It is also extremely important that robotic platforms have the ability to explore depth cues of the environment.

The localization task must be performed by an appropriate system according to the operating environment. According to Zhang et al. (2016) and Wang et al. (2020) Image Recognition based on a computer vision system is one of the most recent methods to locate mobile robots in indoor environments. Its use is advantageous, as it requires less installation space, is robust to environmental conditions, captures RGB images, is a widespread technology, has efficient energy consumption and low cost (Ma and Karaman, 2018).

This work proposes the development and analysis of to location estimation and navigation support system for mobile robots by computer vision aided by depth estimation, using Convolutional Neural Networks (CNNs) allied to the concept of Transfer Learning, which allows this method to be applied as a feature extractor. The classifiers tested were: the Bayesian Classifier, k-Nearest Neighbor (kNN), Random Forest, Multi-layer Perceptron (MLP) and Support Vector Machine (SVM).

For this work, an indoor environment was mapped using the topological approach. The images were acquired by the Microsoft Kinect RGB-D sensor and the following data sets were built: RGB images; depth images (D); RGB-D

mosaic images; depth estimation images (De) and RGB-De mosaic images.

The strategy of this solution is based on location estimation by computer vision using the mosaic images from the combination of the RGB and depth estimation images with practical applications directly linked to mobile robots using the a monocular vision, as an alternative to the more robust sensors such as Light Detection and Ranging (LiDAR) and depth sensors.

The works proposed by da Silva et al. (2020), Islam and Park (2021) and Akai (2022) are similar to this work. In da Silva et al. (2020) the Kinect sensor is used to obtain the location of the mobile robot through RGB and depth mosaic images. The authors in Islam and Park (2021) propose a fine-tuned generative adversarial network to estimate the depth map effectively for a given single RGB image. Meanwhile in Akai (2022) it is presented a mobile robot localization method that uses CNNs to regress the depth from the camera images and performs comparison experiments using a manually created dataset using only a visual inertial odometry sensor, and the KITTI odometry dataset.

In Guo (2022) a turtlebot robot, equipped with odometer, RGB-D camera and laser, and odometer was used to test a cooperative positioning method based on deep laser and vision fusion aiming at the problem of robot positioning. However, in our approach, we use the Kinect sensor as a monocular vision device to obtain the location of the mobile robot through RGB images and depth estimation mosaic, eliminating the use of other sensors.

The focus of the proposed work is to use Transfer Learning to create depth maps from pre-trained models on popular indoor and outdoor Datasets. To evaluate the use of a monocular vision image map and depth estimation image map with mosaic image configuration to estimate robot location in an indoor environment using CNNs as a feature extractor, as also evaluated different machine learning methods.

The results showed that approach obtained 100% in Accuracy and F1-Score with depth estimation images from pre-trained model with outdoor Dataset, and 99.8% with indoor Dataset, thus confirming its efficiency and reliability.

The main contributions of this paper are:

- Efficient use of an RGB-De mosaic image;
- Use and evaluation of CNNs as feature extractors combined with classic machine learning methods;
- Indoor location estimation based on computer vision;
- Outstanding results with the Transfer Learning technique.

The rest of this article is structured as follows: Section 2 gives a brief description of the Transfer Learning technique, the machine learning methods and the method of monocular depth estimation used. Section 3 explains the methodology adopted, detailing the development of the proposed approach. Section 4 gives the results, and highlights the best performances. Finally, Section 5 presents the conclusion and future works.

## 2. OVERVIEW OF CNNS, MACHINE LEARNING METHODS AND DEPTH ESTIMATION METHOD

Originally started from LeCun et al. (1989), CNNs became known when Krizhevsky et al. (2012) used the algorithm in the ImageNet 2012 Large Scale Visual Recognition Challenge (ILSVRC-2012) achieving good results. The goal in ILSVRX-2012 was to solve classification tasks that consisted of 1000 classes and over 1 million samples (Krizhevsky et al., 2012; Deng et al., 2009).

From the results obtained by Krizhevsky et al. (2012), other CNNs architectures were developed in order to improve the results achieved. Among the architectures that emerged are VGG (Simonyan and Zisserman, 2015), MobileNet (Howard et al., 2017), Inception (Szegedy et al., 2015), Xception (Chollet, 2017), NasNet (Zoph and Le, 2017), ResNet (He et al., 2016), DenseNet (Huang et al., 2017) and Inception-Resnet (Szegedy et al., 2016).

### *2.1 CNNs as a Feature Extractor*

Choosing a database to train a Convolutional Neural Network (CNN) satisfactorily can become a very problematic task depending on the desired application (Karpathy et al., 2014; Molchanov et al., 2016). However, this impasse can be resolved using the concept of Transfer Learning (Karpathy et al., 2014; Molchanov et al., 2016; Alexandre, 2014). It ensures that knowledge gained during a problem solving task can be reused to solve a similar problem (Pan and Yang, 2010; Weiss et al., 2016). Using Tansfer Learning does not require large scale training data (Pan and Yang, 2010) and the training time to achieve equivalent accuracy can be reduced (Tang et al., 2019).

To take advantage of the power of a pre-trained CNN one must follow two steps: i) one must eliminate the last fully connected layer of the model and then ii) one must resize the input of the discarded layer to a one-dimensional vector (Karpathy et al., 2014; Molchanov et al., 2016; Alexandre, 2014). In this way, the pre-trained model is used as a feature extractor and is no longer considered a classifier, in accordance with what was used in our proposal. The transfer learning concept is detailed in da Nobrega et al. (2018), who used the technique to perform the classification of pulmonary nodules.

### *2.2 Machine Learning Methods*

The classification task can be performed on the deep features returned by the CNNs. This section summarizes the five techniques of machine learning used in this paper.

The Bayesian Classifier is a probabilistic and supervised technique. It is used to categorize samples by the proportion of each sample that belongs to a given class (Theodoridis and Koutroumbas, 2008). For this, it marks the samples based on the a posteriori probability value that is calculated from conditional densities and a priori probability (Theodoridis and Koutroumbas, 2008).

K-Nearest Neighbor (kNN) is a supervised machine learning technique. It describes the category to which a sample belongs according to the characteristics of the k nearest

neighbors from the training set. The variable k is the number of nearest neighbors used to find out which category the proposed sample belongs to.

The Random Forest algorithm is based on the Decision Tree Method. It introduces supervised learning and aims to create decision trees from a collection of characteristics chosen arbitrarily from the original set. Learning is done with bagging, a meta-algorithm that improves model selection and regression according to the model's stabilization and accuracy (Breiman, 2004).

Multilayer Perceptron (MLP) is a multilayer neural network structure. It is a perceptron composition for dealing with nonlinearly separable questions (Haykin, 2010). The impulses propagate to subsequent layers from the input layer, that initially receives the feature vector. Adjusted by weights, the pulses can be transmitted through the neuronal links (Haykin, 2010).

Based on the Theory of Statistical Learning developed by Vapnik (2000), the Support Vector Machine (SVM) has as main objective to identify surface categories that boost the length between them. To do so, he must demarcate the space models and their entries proportionally to a large feature vector.

### *2.3 Monocular Depth Estimation Method*

Monocular depth estimation has been considered by many CNN methods as a regression of a dense depth map from a single RGB image (Alhashim and Wonka, 2018; Fu et al., 2018; Hao et al., 2018; Hu et al., 2019; Huynh et al., 2020; Lee et al., 2019; Xu et al., 2018).

The current architectures do not perform enough global analysis of the output values. A drawback of convolutional layers is that they only process global information once the tensors reach a very low spatial resolution at or near the bottleneck. For this reason, this paper uses a transformer based architecture block call AdaBins whose architecture is detailed in Bhat et al. (2021). This Block divides the depth range into bins whose center value is estimated adaptively per image. The final depth values are estimated as linear combinations of the bin centers (Bhat et al., 2021).

## 3. METHODOLOGY

Figure 1 shows the steps of the methodology of our approach. The first step is Image Acquisition, in which the Kinect sensor using the open-source libfreenect software available in the OpenKinect community captures the RGB and depth-of-environment images (Sarbolandi et al., 2015).

The community makes this library available for Kinect versions v1 and v2 (`github.com/OpenKinect/libfreenect`). In this paper, we used version 1 of this library. The next step is the preprocessing step. Here, initialy, are estimate two high quality dense depth map from a each RGB input image via the AdaBins block (Bhat et al., 2020). The depths maps are genereted using AdaBins block pretreined with two popular indoor and outdoor datasets, NYU (Silberman et al., 2012) and KITTI (Geiger et al., 2013) respectively. Then all images are scaled to the size of 320 x 240 pixels and concatenated to give the mosaic RGB-D and RGB-De image dataset, where the tiled images have a size of 640 x 480 pixels. Copies of the full-sized depth original and estimated images are made for the composition of the depth image dataset. The following step is Feature Extraction, which returns the feature vectors that will serve as input to the Machine Learning techniques, which perform the classification task; thus, completing the last step of the process.

All datasets genereted (Depth, RGB-D, Depth estimated and RGB-De) have 50 samples per class, totaling 1500 samples for each dataset. Figure 2 shows examples of samples from each class and which physical room they are in. Figure 3 show a example of sample for the depth estimation genereted by AdaBins Block.

The environment for the navigation of the robot was the second floor of a university campus building. The indoor environment, which is a controlled environment, has characteristics that assist in the localization and navigation of the robot.

Figure 4 shows the topological map of the environment, which is made by six rooms, which are referred to as R1 to R6. Each room is divided into one or two nodes, totaling in nine nodes, labeled with letters A through I; each node represents the position of the robot in the room. The nodes are divided into classes, which are the possible directions performed by the robot, and are labeled from C1 to C30. These classes consist of the classes of our approach. If the robot identifies the class, it can define the direction it should go, and node and room in which it is.

Table 1 illustrates some of the routes that the mobile robot can follow in the environment. Five distinct commands for the locomotion task were given for the C20 to C9 route. The robot begins at C20 (node F). First a command to turn 90° to the right is sent to it, sequentially, two "go forward" commands are given. Then, another command is sent to the robot to turn 90° to the left, and another command to go forward is received by the robot, and thus it reaches its destination, which corresponds to C9 (node C).

Table 1. Example of routes that can be performed by the mobile robot based on the topological map and using the available commands. Commands: GF - Go Forward; TR90 - Turn Right 90°; TL90 - Turn Left 90°; TR180 - Turn Right 180°; TL180 - Turn Left 180°.

| Route | Commands |
|---|---|
| C1 to C7 | GF, TL90 |
| C7 to C16 | TL90, GF, GF, TR90, GF, TR90, GF, TL180 |
| C16 to C20 | TR180, GF, TL90 |
| C20 to C9 | TR90, GF, GF, TL90, GF |
| C9 to C25 | GF, GF |
| C25 to C30 | TR180, GF, TL90, GF, TR90 |
| C30 to C6 | TR90, GF, TL90, GF, TR90, GF, GF, TL180 |
| C6 to C1 | GF, TR180 |

The following CNN architectures were considered for the feature extraction step: VGG16, VGG19, MobileNet, ResNet50, InceptionV3, Xception, InceptionResNetV2, DenseNet121, DenseNet169, DenseNet201, and NASNet-Mobile. Table 6 presents the number of attributes extracted by each architecture.

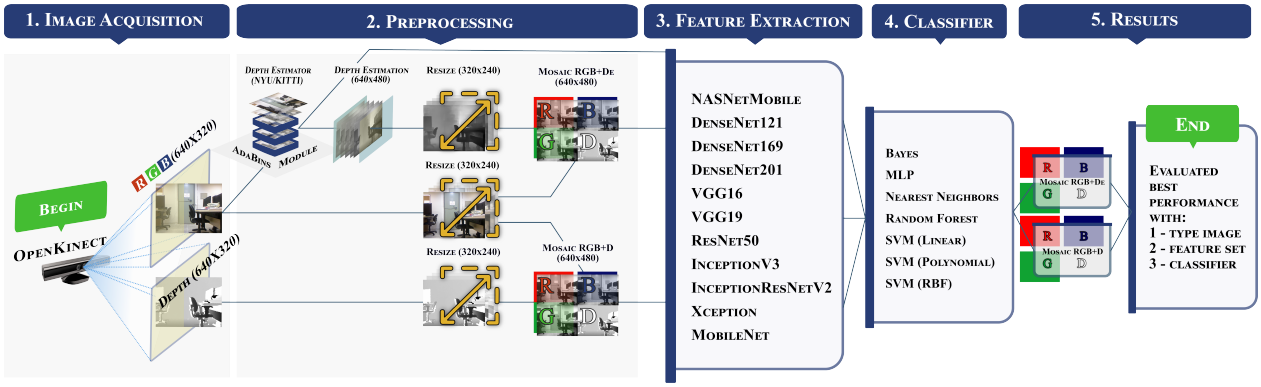


Figure 1. Flowchart of the adopted methodology in 5 steps: (1) Image Acquisition using the Kinect Sensor; (2) Preprocessing images to create the datasets: RGB-D and RGB-De; (3) Feature extraction with eleven CNN architectures; (4) Classification with five methods of machine learning; (5) Evaluation of the results in all datasets.

Figure 2. Examples of samples of the dataset for each class.

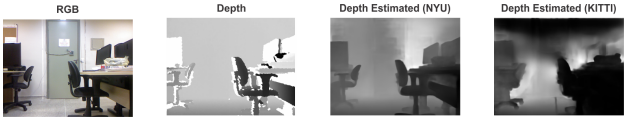


Figure 3. Example of sample of depth estimation for a dataset class.

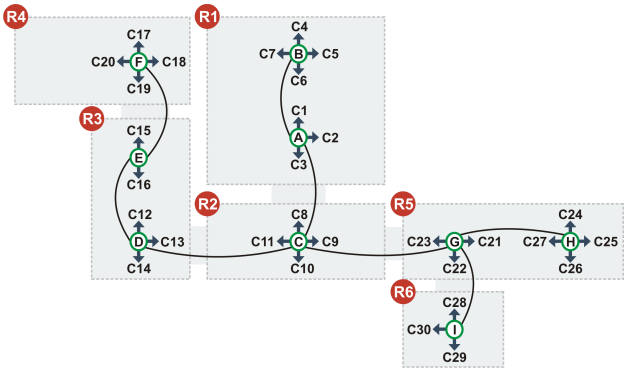


Figure 4. Topological map of the environment.

In the classification processes, the Bayesian classifier operated with the Gaussian probability density function. The hyperparameter $k$ from the kNN classifier was chosen through a grid search, testing the odd values from 1 to 11. The MLP carried out its training using the Levenberg-Marquardt method with neurons ranging from 2 to 1000 in the hidden layer. SVM used linear and RBF kernels, and the hyperparameters $C$ and $\gamma$ with $2^{-5}, 2^{-4}, 2^{-3}, ..., 2^{15}$ and $2^{-15}, 2^{-14}, ..., 2^{5}$ values, respectively. The MLP and SVM hyperparameters were determined through cross-validation with 10-folds. The random search for Random Forest was used to find: the number of features for the best split (1,2,3,...,or 10); the maximum tree depth to be used (6 or none); the minimum number of samples required to split an internal node (1,2,3,...,or 10); the minimum number of samples per leaf to be used (1,2,3,...,or 10); whether bootstrap samples should be used when building trees or not, and which function should be used to measure the quality of a split (gini or entropy). Furthermore, the Random Forest training process used 3000 estimators.

The following measures were used to evaluate the performance of the classifiers: the micro averaged Accuracy and F1-Score. F1-Score was used because both metrics Precision and Recall are essential for the system since false positives and false negatives are not desirable.

## 4. RESULTS AND DISCUSSION

In this section, we discuss the results obtained by combining the extraction of features using CNNs and the machine learning methods of both datasets. The images were randomly divided into 10 groups, 20% for testing and the rest for training, in each of the 10 iterations.

The hardware infrastructure used in the solution was an Intel Core i7 7th Gen with 16Gb of memory and NVIDIA GeForce GTX 1050 Ti 4Gb with GPU, running Windows 10 Professional 64-bit operating system. The part of the system that actually performs the image processing was developed in Python language and uses the TensorFlow, Keras libraries, and SciKit Learning with the use of feature extractors and classification with neural networks and CNNs.

Tables 2 and 3 present the mean values and standard deviations of Accuracy and F1-Score, respectively, for image datasets obtained from Kinect sensor. The Tables 4 and 5, for the AdaBins block pre-trained weights with the NYU dataset (indoor Dataset) and the Tables 7 and 8, for the AdaBins block pre-trained weights with the KITTI dataset (outdoor Dataset), respectively. The best values are highlighted in green in the all Tables.

Table 2. Accuracy reached by CNN and Classifiers with RGB-D images obtained from Kinect sensor

| RGB-D | | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 90.943±0.951 | 99.358±0.561 | 98.784±0.538 | 97.441±1.865 | 99.629±0.486 | 4.597±1.191 | 99.629±0.486 |
| VGG19 | 92.069±1.639 | 99.211±0.635 | 98.782±0.803 | 98.634±0.957 | 99.324±0.712 | 15.505±3.476 | 99.324±0.712 |
| MobileNet | 96.789±0.508 | 99.873±0.253 | 98.431±0.854 | 99.193±0.768 | 99.746±0.506 | 99.746±0.506 | 99.873±0.253 |
| InceptionV3 | 96.207±1.205 | 97.726±0.795 | 98.487±0.878 | 99.041±0.608 | 98.899±0.981 | 9.569±5.327 | 98.899±0.981 |
| Xception | 95.942±1.438 | 99.352±0.588 | 98.270±1.007 | 98.623±0.933 | 99.316±0.630 | 17.474±8.369 | 99.316±0.630 |
| ResNet50 | 83.280±1.943 | 92.362±2.005 | 96.098±2.067 | 85.882±19.318 | 96.661±1.481 | 8.385±4.033 | 96.779±1.583 |
| InceptionResNetV2 | 93.400±2.494 | 97.996±0.449 | 97.684±1.413 | 98.484±1.213 | 98.635±0.893 | 12.566±2.087 | 98.635±0.893 |
| NASNetMobile | 90.628±0.886 | 96.283±1.325 | 96.030±0.944 | 94.456±2.283 | 97.973±0.531 | 96.620±1.230 | 97.973±0.531 |
| DenseNet121 | 87.624±1.673 | 99.463±0.270 | 99.327±0.608 | 99.183±0.828 | 99.437±0.848 | 14.330±7.396 | 99.437±0.848 |
| DenseNet169 | 95.574±0.738 | 99.594±0.332 | 99.581±0.572 | 99.449±0.535 | 99.722±0.340 | 6.967±3.109 | 99.722±0.340 |
| DenseNet201 | 93.721±1.490 | 99.456±0.517 | 99.580±0.557 | 99.061±0.529 | 99.470±0.513 | 8.321±7.803 | 99.737±0.322 |

Table 3. F1-Score reached by CNN and Classifiers with RGB-D images obtained from Kinect sensor

| RGB-D | | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 88.137±0.916 | 99.275±0.644 | 98.726±0.584 | 97.165±2.004 | 99.579±0.568 | 0.807±0.552 | 99.579±0.568 |
| VGG19 | 90.894±1.725 | 99.190±0.659 | 98.818±0.793 | 98.675±0.953 | 99.306±0.737 | 8.866±2.863 | 99.306±0.737 |
| MobileNet | 96.984±0.473 | 99.888±0.223 | 98.496±0.771 | 99.154±0.812 | 99.771±0.457 | 99.771±0.457 | 99.888±0.223 |
| InceptionV3 | 96.058±1.340 | 97.590±0.781 | 98.352±0.943 | 98.936±0.624 | 98.788±0.978 | 3.963±3.251 | 98.788±0.978 |
| Xception | 95.458±1.759 | 99.314±0.601 | 98.154±1.105 | 98.533±1.119 | 99.246±0.685 | 9.524±7.171 | 99.246±0.685 |
| ResNet50 | 81.546±2.297 | 91.985±2.173 | 95.841±2.206 | 85.215±20.573 | 96.528±1.570 | 2.386±2.276 | 96.658±1.649 |
| InceptionResNetV2 | 92.842±2.711 | 97.912±0.445 | 97.479±1.605 | 98.337±1.386 | 98.501±0.960 | 10.386±1.784 | 98.501±0.960 |
| NASNetMobile | 89.585±1.361 | 95.914±1.528 | 95.666±1.101 | 94.023±2.538 | 97.857±0.534 | 96.160±1.495 | 97.834±0.574 |
| DenseNet121 | 84.637±1.725 | 99.479±0.264 | 99.268±0.651 | 99.123±0.922 | 99.355±0.987 | 8.362±6.364 | 99.355±0.987 |
| DenseNet169 | 94.123±0.881 | 99.582±0.341 | 99.584±0.558 | 99.392±0.599 | 99.682±0.390 | 2.197±2.264 | 99.682±0.390 |
| DenseNet201 | 92.227±2.189 | 99.461±0.503 | 99.605±0.522 | 99.093±0.500 | 99.497±0.496 | 3.040±4.841 | 99.766±0.285 |

Tables 2 and 3 shows that the highest accuracy and F1-score were achieved by combining the MobileNet architecture with the kNN and SVM (RBF) Classifiers, with 99.873% and 99.888% respectively for RGB-D mosaic images. The worst result was obtained by the combination of the VGG16 architecture with the SVM Polynomial Classifier obtained only an accuracy of 4.597% and F1-score of 0.807%.

For the RGB-De mosaic images (depth estimation pre-trained with NYU Dataset), the highest accuracy was achieved by combining the VGG16 architecture with the kNN Classifier, with 99.872% as show in Table 4. The combination of the kNN Classifier with the MobileNet and DenseNet201 Architectures also stands out in the Table 4. They present an accuracy greater than 99.8%. Already, in Table 5 the highest F1-Score were achieved by combining the VGG16 and DenseNet201 Architectures with the kNN Classifier, with 99.874%. But, the combination with MobileNet Architecture also stands out with F1-score greater than 99.8%. The worst results of accuracy and F1-Score were obtained by the DenseNet121 Architecture combining with SVM Polynomial classifier, with 4.243% and 0.993% in Tables 4 and 5 respectively.

For the RGB-De mosaic images (depth estimation pre-trained with NYU Dataset), the highest accuracy and F1-Score were achieved in 4 different combinations in the Tables 7 and 8, all with a value of 100%. The worst results of accuracy and F1-Score were obtained by the combination of the VGG16 architecture with the SVM Polynomial Classifier obtained only an accuracy of 6.729% and 3.794%, respectively.

These results show that the RGB-De mosaic images performed well using the depth estimation approach when compared to the RGB-D mosaic images approach (depth sensor). For this approach, we can highlight the performance of the kNN Classifier combined with the VGG16 and MobileNet Architectures. These combinations present results superior to 99.8% for the RGB-De images generated from the obtained depth estimates. In general, the use of pre-trained weights indoors or outdoors to generate depth estimation showed no loss of performance in the classifiers. However, the depth estimation generated when the model was pre-trained with an outdoor Dataset bring less detail to objects closer to the camera, making the classification more general. This can be seen by the 100% accuracy in some combinations of classifiers and architectures.

Table 6 shows the extraction times of each CNN architecture used. For the mosaic images RGB-D, RGB-De (NYU) and RGB-De (KITTI), the MobileNet architecture had the shortest extraction time of 8.895 ms, 8.710 ms and 11.871 ms while in contrast, the InceptionResNetV2 architecture had the longest time of 80.095 ms, 81.290 ms and 91.352 ms.

In order to compare other classification methods with the kNN the function call time was calculated because this distance-based classification method does not require mathematical training and only needs memory to store the data.

Tables 9, 10 and 11 presents the training times of each classifier, considering all databases. The classifier that achieved the shortest training time for the RGB-D and RGB-De (NYU and KITTI) mosaic images was the kNN, with values of 0.024s, 0.024s and 0.028s respectively, with the attributes returned by the VGG19 and VGG16 architectures.

## 5. CONCLUSION

In this article, is presented a approach for location estimation and consequently support for mobile robot navigation. The work consists of a computer vision system using Deep Transfer Learning, by combining the concept of Transfer Learning with CNN, and monocular vision for depth estimate as from image acquisition. The results of the proposed approach are compared with results generated from depth images obtained by a Kinect sensor. The localization of the robot was performed in an indoor environment and the images acquired from the sensor are organized to give rise to new images, creating an RGB-D mosaic image database.

The depth estimation applied in the proposed Deep Transfer Learning approach using the RGB-De mosaic images

Table 4. Accuracy reached by CNN and Classifiers with estimated RGB-De images taken from the AdaBins block pre-trained weights with the NYU dataset.

| | RGB-De | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 93.079±0.791 | 99.872±0.254 | 99.081±0.781 | 99.621±0.507 | 99.609±0.320 | 9.834±2.580 | 99.749±0.307 |
| VGG19 | 84.194±1.348 | 99.356±0.698 | 99.365±0.559 | 98.689±1.121 | 99.488±0.477 | 7.047±3.511 | 99.488±0.477 |
| MobileNet | 97.460±1.288 | 99.870±0.258 | 99.744±0.313 | 99.326±0.434 | 99.473±0.480 | 99.617±0.507 | 99.744±0.313 |
| InceptionV3 | 97.312±0.666 | 98.790±0.293 | 98.501±0.785 | 97.138±2.547 | 98.905±0.549 | 33.881±3.253 | 98.905±0.549 |
| Xception | 97.085±0.553 | 99.174±1.046 | 99.328±0.449 | 99.052±1.020 | 99.315±0.777 | 89.977±1.706 | 99.174±1.046 |
| ResNet50 | 78.396±1.072 | 97.789±1.228 | 98.160±0.925 | 96.224±2.091 | 98.817±0.452 | 4.701±3.216 | 98.817±0.452 |
| InceptionResNetV2 | 94.571±0.944 | 98.586±1.050 | 97.619±1.008 | 96.530±4.396 | 98.696±0.765 | 9.767±5.754 | 98.696±0.765 |
| NASNetMobile | 94.099±0.814 | 97.343±0.772 | 97.733±0.288 | 97.440±1.140 | 98.306±0.707 | 96.823±1.211 | 98.574±0.743 |
| DenseNet121 | 96.387±1.115 | 99.716±0.567 | 99.734±0.326 | 99.037±1.191 | 99.577±0.565 | 4.243±1.138 | 99.577±0.565 |
| DenseNet169 | 97.195±0.972 | 99.611±0.525 | 98.945±1.156 | 99.606±0.528 | 99.482±0.492 | 8.893±4.378 | 99.466±0.496 |
| DenseNet201 | 97.457±1.792 | 99.859±0.281 | 99.731±0.329 | 99.463±0.658 | 99.736±0.324 | 5.719±3.119 | 99.736±0.324 |

Table 5. F1-Score reached by CNN and Classifiers with estimated RGB-De images taken from the AdaBins block pre-trained weights with the NYU dataset.

| | RGB-De | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 91.304±0.707 | 99.874±0.250 | 99.007±0.851 | 99.552±0.600 | 99.584±0.340 | 3.402±2.388 | 99.718±0.344 |
| VGG19 | 82.290±1.735 | 99.331±0.734 | 99.369±0.542 | 98.666±1.160 | 99.489±0.465 | 1.991±2.600 | 99.489±0.465 |
| MobileNet | 97.482±1.283 | 99.865±0.269 | 99.753±0.304 | 99.314±0.398 | 99.437±0.502 | 99.606±0.520 | 99.730±0.329 |
| InceptionV3 | 97.267±0.687 | 98.764±0.346 | 98.472±0.781 | 97.160±2.464 | 98.855±0.594 | 29.675±2.176 | 98.855±0.594 |
| Xception | 96.953±0.565 | 99.212±0.988 | 99.314±0.477 | 98.946±1.197 | 99.337±0.752 | 88.378±1.668 | 99.212±0.988 |
| ResNet50 | 76.467±1.291 | 97.673±1.242 | 98.112±0.910 | 96.146±1.886 | 98.765±0.491 | 1.583±2.765 | 98.765±0.491 |
| InceptionResNetV2 | 94.334±1.032 | 98.373±1.268 | 97.393±1.214 | 96.213±4.736 | 98.570±0.939 | 4.817±3.820 | 98.583±0.949 |
| NASNetMobile | 93.866±0.651 | 97.188±0.852 | 97.536±0.428 | 97.387±1.163 | 98.136±0.781 | 96.773±1.236 | 98.441±0.797 |
| DenseNet121 | 96.133±1.170 | 99.621±0.756 | 99.688±0.381 | 98.924±1.320 | 99.521±0.672 | 0.993±0.928 | 99.521±0.672 |
| DenseNet169 | 96.698±1.121 | 99.625±0.503 | 98.989±1.141 | 99.595±0.538 | 99.474±0.501 | 3.145±2.347 | 99.474±0.501 |
| DenseNet201 | 97.557±1.633 | 99.874±0.250 | 99.749±0.307 | 99.479±0.636 | 99.717±0.347 | 1.630±2.365 | 99.717±0.347 |

Table 6. Number of attributes and extration time returned by each CNN architecture

| CNN Architecture | Number of attributes | Extraction Time (ms) | | |
|---|---|---|---|---|
| | | RGB-D | RGB-De (NYU) | RGB-De (KITTI) |
| VGG16 | 512 | 21.256±31.036 | 21.252±31.732 | 22.099±48.433 |
| VGG19 | 512 | 25.770±11.220 | 25.817±11.940 | 26.768±28.478 |
| MobileNet | 1024 | 8.895±34.690 | 8.710±30.262 | 11.871±109.150 |
| InceptionV3 | 2048 | 36.081±69.628 | 36.656±74.127 | 40.619±140.990 |
| Xception | 2048 | 32.868±25.068 | 32.995±25.965 | 35.109±58.785 |
| ResNet50 | 2048 | 26.233±27.794 | 25.928±28.885 | 28.324±57.001 |
| InceptionResNetV2 | 1536 | 80.095±106.805 | 81.290±121.896 | 91.352±283.922 |
| NASNetMobile | 1056 | 42.908±140.504 | 45.862±162.584 | 68.680±308.278 |
| DenseNet121 | 1024 | 38.621±139.208 | 40.873±166.983 | 48.318±290.044 |
| DenseNet169 | 1664 | 56.857±180.714 | 58.986±201.686 | 68.588±355.374 |
| DenseNet201 | 1920 | 75.657±231.761 | 76.068±260.249 | 89.623±461.305 |

Table 7. Accuracy reached by CNN and Classifiers with RGB-De images taken from the AdaBins block pre-trained weights with the KITTI dataset.

| | RGB-De | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 91.597±1.558 | 100.000±0.000 | 99.747±0.309 | 99.737±0.321 | 100.000±0.000 | 6.729±2.430 | 100.000±0.000 |
| VGG19 | 89.260±1.431 | 99.209±0.972 | 98.815±1.213 | 98.802±1.003 | 99.577±0.845 | 14.633±6.406 | 99.442±0.820 |
| MobileNet | 98.365±0.842 | 99.876±0.246 | 99.317±0.434 | 99.593±0.545 | 99.863±0.273 | 99.454±0.274 | 99.863±0.273 |
| InceptionV3 | 97.161±0.888 | 99.029±1.066 | 98.516±0.706 | 97.164±2.024 | 99.181±0.799 | 59.929±2.953 | 99.321±0.599 |
| Xception | 97.214±0.825 | 99.349±0.570 | 99.339±0.411 | 99.474±0.488 | 99.604±0.324 | 98.025±1.141 | 99.604±0.324 |
| ResNet50 | 83.686±2.996 | 97.340±1.315 | 95.626±1.372 | 95.432±2.659 | 97.901±1.054 | 10.809±8.592 | 97.896±0.821 |
| InceptionResNetV2 | 95.861±0.796 | 98.535±0.475 | 98.945±0.689 | 99.201±0.517 | 99.074±0.687 | 61.871±2.533 | 99.074±0.687 |
| NASNetMobile | 92.964±3.520 | 97.981±0.780 | 98.215±0.990 | 95.391±4.993 | 99.039±0.717 | 98.778±0.912 | 99.039±0.717 |
| DenseNet121 | 95.707±0.915 | 99.585±0.545 | 99.327±0.740 | 99.585±0.545 | 99.720±0.342 | 13.151±6.568 | 99.720±0.342 |
| DenseNet169 | 98.941±0.515 | 99.708±0.583 | 99.592±0.334 | 99.446±0.539 | 99.738±0.321 | 13.441±2.801 | 100.000±0.000 |
| DenseNet201 | 96.497±0.658 | 99.595±0.332 | 99.364±0.569 | 99.330±0.043 | 99.722±0.340 | 11.269±5.554 | 99.722±0.340 |

Table 8. F1-Score reached by CNN and Classifiers with RGB-De images taken from the AdaBins block pre-trained weights with the KITTI dataset.

| | RGB-De | | | | | | |
|---|---|---|---|---|---|---|---|
| Architecture \ Classifier | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 89.628±1.821 | 100.000±0.000 | 99.762±0.291 | 99.744±0.314 | 100.000±0.000 | 3.794±2.374 | 100.000±0.000 |
| VGG19 | 87.553±1.681 | 99.154±1.040 | 98.718±1.341 | 98.642±1.162 | 99.534±0.931 | 6.717±3.840 | 99.386±0.903 |
| MobileNet | 98.211±0.924 | 99.878±0.242 | 99.200±0.530 | 99.570±0.566 | 99.830±0.338 | 99.365±0.320 | 99.830±0.338 |
| InceptionV3 | 97.050±0.969 | 98.974±1.137 | 98.479±0.800 | 97.146±2.038 | 99.173±0.800 | 58.474±2.888 | 99.307±0.606 |
| Xception | 97.019±0.864 | 99.385±0.537 | 99.337±0.416 | 99.506±0.474 | 99.628±0.305 | 98.017±1.143 | 99.628±0.305 |
| ResNet50 | 82.218±3.065 | 97.091±1.445 | 95.410±1.393 | 95.019±3.158 | 97.715±1.149 | 4.415±5.474 | 97.712±0.924 |
| InceptionResNetV2 | 95.667±0.888 | 98.458±0.485 | 98.936±0.669 | 99.112±0.631 | 99.014±0.789 | 64.089±2.722 | 99.014±0.789 |
| NASNetMobile | 92.667±3.630 | 97.965±0.675 | 98.113±1.025 | 95.283±5.080 | 98.986±0.734 | 98.756±0.877 | 98.976±0.748 |
| DenseNet121 | 95.062±1.260 | 99.547±0.569 | 99.282±0.798 | 99.548±0.568 | 99.694±0.387 | 6.867±5.338 | 99.694±0.387 |
| DenseNet169 | 98.893±0.575 | 99.712±0.575 | 99.618±0.314 | 99.457±0.545 | 99.753±0.304 | 6.746±2.665 | 100.000±0.000 |
| DenseNet201 | 94.984±1.182 | 99.515±0.412 | 99.269±0.626 | 99.192±0.171 | 99.718±0.356 | 6.990±5.994 | 99.718±0.356 |

proved to be an efficient method for the task of mobile robot location estimation and navigation support, achieving 100% in Accuracy and F1-Score with depth estimation images from pre-trained model with outdoor Dataset, and 99.8% with indoor Dataset. The processing times performed were also suitable for a computer vision system, with values of 8.710ms and 0.024s, and 11.871ms and 0.028s, for the extraction and training times for RGB-De (NYU) and RGB-De (KITTI), respectively.

For future work, outdoor environments and other indoor environments will be tested with this approach. In addition, we can evaluate others depth estimation methods such as M4Depth, present in Fonder et al. (2021), or FastDepth, present in Wofk et al. (2019). We can apply other machine learning techniques such as the Optimum-Path Forest (OPF), presented in Nunes et al. (2014), and still utilize other CNN architectures, such as ResNeXt-50 (Xie et al., 2016), and CapsNet (Sabour et al., 2017). Another important aspect involves evaluating simpler classification methods by applying the raw images directly, and in this

Table 9. Training time(s) and function call time(s) for kNN of the Classifiers with RGB-D images obtained from Kinect sensor

| RGB-D | | | | | | | |
|---|---|---|---|---|---|---|---|
| Classifier / Architecture | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 0.027±0.006 | 0.024±0.001 | 5.919±1.349 | 42.061±9.755 | 1.286±0.064 | 2.372±0.053 | 1.350±0.053 |
| VGG19 | 0.051±0.023 | 0.024±0.000 | 6.956±1.747 | 54.409±14.176 | 1.302±0.074 | 2.389±0.025 | 1.447±0.040 |
| MobileNet | 0.043±0.013 | 0.053±0.001 | 9.063±2.557 | 110.412±41.555 | 3.111±0.055 | 2.939±0.034 | 3.256±0.066 |
| InceptionV3 | 0.098±0.018 | 0.178±0.110 | 11.953±2.348 | 67.547±24.147 | 6.068±0.103 | 8.909±0.266 | 6.438±0.112 |
| Xception | 0.087±0.011 | 0.163±0.071 | 14.665±1.735 | 126.040±51.244 | 5.683±0.079 | 9.223±0.279 | 5.794±0.101 |
| ResNet50 | 0.123±0.017 | 0.121±0.004 | 51.246±1.704 | 299.568±83.043 | 4.326±0.101 | 8.926±0.240 | 4.530±0.109 |
| InceptionResNetV2 | 0.075±0.018 | 0.090±0.004 | 42.111±3.417 | 145.097±54.793 | 4.329±0.083 | 6.799±0.207 | 4.515±0.073 |
| NASNetMobile | 0.090±0.024 | 0.051±0.003 | 35.841±1.852 | 137.584±35.169 | 3.009±0.100 | 2.871±0.084 | 4.168±0.094 |
| DenseNet121 | 0.038±0.000 | 0.056±0.004 | 9.864±2.132 | 124.647±49.385 | 2.778±0.058 | 4.566±0.140 | 3.129±0.049 |
| DenseNet169 | 0.080±0.027 | 0.088±0.002 | 13.492±2.729 | 147.620±52.917 | 4.385±0.113 | 7.171±0.211 | 4.689±0.098 |
| DenseNet201 | 0.079±0.014 | 0.109±0.002 | 31.235±1.370 | 186.674±66.198 | 4.985±0.084 | 8.323±0.231 | 5.356±0.080 |

Table 10. Training time(s) and function call time(s) for kNN of the Classifiers with RGB-De images taken from the AdaBins block pre-trained weights with the NYU dataset.

| RGB-De | | | | | | | |
|---|---|---|---|---|---|---|---|
| Classifier / Architecture | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 0.049±0.017 | 0.024±0.002 | 26.114±2.707 | 62.683±18.757 | 1.305±0.149 | 2.627±0.228 | 2.257±0.045 |
| VGG19 | 0.113±0.080 | 0.025±0.001 | 24.468±9.099 | 55.731±23.876 | 1.411±0.038 | 2.866±0.085 | 2.421±0.062 |
| MobileNet | 0.056±0.008 | 0.072±0.016 | 16.663±6.779 | 70.738±24.376 | 3.062±0.039 | 3.070±0.055 | 4.566±0.066 |
| InceptionV3 | 0.069±0.011 | 0.129±0.033 | 11.424±2.465 | 290.229±111.480 | 6.442±0.115 | 8.956±0.222 | 6.964±0.138 |
| Xception | 0.091±0.020 | 0.148±0.041 | 23.374±2.374 | 131.792±50.627 | 5.377±0.091 | 8.426±0.111 | 7.679±0.137 |
| ResNet50 | 0.107±0.029 | 0.145±0.005 | 16.438±2.878 | 410.348±107.988 | 4.817±0.103 | 9.948±0.269 | 5.200±0.074 |
| InceptionResNetV2 | 0.063±0.005 | 0.082±0.004 | 6.977±1.744 | 166.297±58.708 | 4.007±0.070 | 6.793±0.225 | 5.028±0.129 |
| NASNetMobile | 0.059±0.014 | 0.055±0.002 | 9.965±2.093 | 116.031±40.760 | 3.015±0.071 | 3.278±0.107 | 4.574±0.123 |
| DenseNet121 | 0.044±0.006 | 0.056±0.003 | 13.339±2.684 | 103.971±39.986 | 2.384±0.049 | 4.429±0.111 | 2.668±0.050 |
| DenseNet169 | 0.058±0.005 | 0.101±0.012 | 10.693±2.624 | 80.411±26.885 | 4.045±0.070 | 7.116±0.182 | 5.638±0.084 |
| DenseNet201 | 0.079±0.011 | 0.154±0.090 | 33.409±1.510 | 97.632±32.946 | 4.600±0.082 | 8.478±0.250 | 5.341±0.153 |

Table 11. Training time(s) and function call time(s) for kNN of the Classifiers with RGB-De images taken from the AdaBins block pre-trained weights with the KITTI dataset.

| RGB-De | | | | | | | |
|---|---|---|---|---|---|---|---|
| Classifier / Architecture | Bayes | kNN | RF | MLP | SVM (Linear) | SVM (Poly.) | SVM (RBF) |
| VGG16 | 0.047±0.019 | 0.028±0.002 | 36.526±1.529 | 50.299±19.809 | 1.411±0.086 | 2.805±0.122 | 1.565±0.030 |
| VGG19 | 0.041±0.013 | 0.031±0.004 | 27.102±2.609 | 66.370±20.671 | 1.377±0.108 | 2.671±0.056 | 1.666±0.110 |
| MobileNet | 0.048±0.007 | 0.076±0.037 | 14.015±3.005 | 57.264±24.453 | 3.264±0.074 | 3.116±0.068 | 4.017±0.070 |
| InceptionV3 | 0.113±0.024 | 0.210±0.083 | 46.377±2.200 | 198.043±81.017 | 6.977±0.124 | 9.622±0.257 | 8.813±0.215 |
| Xception | 0.099±0.014 | 0.125±0.005 | 30.111±2.605 | 118.055±54.341 | 5.103±0.159 | 5.516±0.171 | 5.523±0.123 |
| ResNet50 | 0.109±0.009 | 0.162±0.062 | 12.591±2.571 | 355.899±44.826 | 4.277±0.093 | 10.137±0.221 | 5.344±0.101 |
| InceptionResNetV2 | 0.101±0.047 | 0.128±0.055 | 17.968±3.033 | 185.668±73.678 | 4.481±0.064 | 6.972±0.145 | 4.724±0.058 |
| NASNetMobile | 0.067±0.019 | 0.055±0.001 | 12.781±2.007 | 151.102±35.603 | 2.969±0.062 | 2.916±0.070 | 3.323±0.056 |
| DenseNet121 | 0.045±0.000 | 0.054±0.002 | 51.697±1.842 | 155.797±62.362 | 2.726±0.072 | 5.045±0.131 | 3.718±0.107 |
| DenseNet169 | 0.081±0.026 | 0.109±0.021 | 34.326±2.357 | 32.265±10.791 | 4.513±0.128 | 8.174±0.244 | 6.965±0.174 |
| DenseNet201 | 0.069±0.015 | 0.117±0.013 | 27.906±2.056 | 150.570±47.949 | 5.111±0.084 | 9.263±0.300 | 5.373±0.139 |

context we can mention the Nearest Neighbors (NN), Minimum Distance Classifier (MDC), and the Linear and Quadratic Classifier (CQ and LMQ).

## REFERENCES

Akai, N. (2022). Mobile robot localization considering uncertainty of depth regression from camera images. *IEEE Robotics and Automation Letters*, 7(2), 1431–1438. doi:10.1109/LRA.2021.3140062.

Alexandre, L.A. (2014). 3d object recognition using convolutional neural networks with transfer learning between input channels. In *IAS*.

Alhashim, I. and Wonka, P. (2018). High quality monocular depth estimation via transfer learning. *ArXiv*, abs/1812.11941.

Bhat, S., Alhashim, I., and Wonka, P. (2021). Adabins: Depth estimation using adaptive bins. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4008–4017.

Bhat, S.F., Alhashim, I., and Wonka, P. (2020). Adabins: Depth estimation using adaptive bins. *CoRR*, abs/2011.14141.

Breiman, L. (2004). Random forests. *Machine Learning*, 45, 5–32.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1800–1807.

da Nobrega, R.V.M., Filho, P.P.R., Rodrigues, M.B., da Silva, S.P.P., Júnior, C.M.J.M.D., and de Albuquerque, V.H.C. (2018). Lung nodule malignancy classification in chest computed tomography images using transfer learning and convolutional neural networks. *Neural Computing and Applications*, 1–18.

da Silva, S.P.P., Almeida, J.S., Ohata, E.F., Rodrigues, J.J., de Albuquerque, V.H.C., and Filho, P.P.R. (2020). Monocular vision aided depth map from rgb images to estimate of localization and support to navigation of mobile robots. *IEEE Sensors Journal*, 20, 12040–12048.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*.

Elaraby, A.F., Hamdy, A., and Rehan, M.M. (2018). A kinect-based 3d object detection and recognition system with enhanced depth estimation algorithm. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 247–252.

Fonder, M., Ernst, D., and Droogenbroeck, M.V. (2021). M4depth: A motion-based approach for monocular depth estimation on video sequences. *CoRR*, abs/2105.09847.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2002–2011.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11), 1231–1237. doi:10.1177/0278364913491297. URL `https://doi.org/10.1177/0278364913491297`.

Guo, J. (2022). Robot localization and scene modeling based on rgb-d sensor. In J. Macintyre, J. Zhao, and X. Ma (eds.), *The 2021 International Conference on Machine Learning and Big Data Analytics for IoT Security and Privacy*, 753–760. Springer International Publishing, Cham.

Hao, Z., Li, Y., You, S., and Lu, F. (2018). Detail preserving depth estimation from a single image using attention guided networks. *2018 International Conference on 3D Vision (3DV)*, 304–313.

Haykin, S. (2010). Neural networks and learning machines.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861.

Hu, J., Ozay, M., Zhang, Y., and Okatani, T. (2019). Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1043–1051.

Huang, G., Liu, Z., and Weinberger, K.Q. (2017). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

Huynh, L., Nguyen-Ha, P., Matas, J., Rahtu, E., and Heikkila, J. (2020). Guiding monocular depth estimation using depth-attention volume. *ArXiv*, abs/2004.02760.

Islam, N.U. and Park, J. (2021). Depth estimation from a single rgb image using fine-tuned generative adversarial network. *IEEE Access*, 9, 32781–32794. doi:10.1109/ACCESS.2021.3060435.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1725–1732.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 84 – 90.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., and Jackel, L.D. (1989). Handwritten digit recognition with a back-propagation network. In *NIPS*.

Lee, J.H., Han, M.K., Ko, D.W., and Suh, I.H. (2019). From big to small: Multi-scale local planar guidance for monocular depth estimation. *ArXiv*, abs/1907.10326.

Li, Y. and Shi, C. (2018). Localization and navigation for indoor mobile robot based on ros. *2018 Chinese Automation Congress (CAC)*, 1135–1139.

Ma, F. and Karaman, S. (2018). Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1–8.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2016). Pruning convolutional neural networks for resource efficient transfer learning. *ArXiv*, abs/1611.06440.

Nunes, T., Coelho, A., Lima, C., Papa, J., and Albuquerque, V. (2014). Eeg signal classification for epilepsy diagnosis via optimum path forest - a systematic assessment. *Neurocomputing*, –. doi:10.1016/j.neucom.2014.01.020.

Pan, S.J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22, 1345–1359.

Sabour, S., Frosst, N., and Hinton, G.E. (2017). Dynamic routing between capsules. *CoRR*, abs/1710.09829.

Sarbolandi, H., Lefloch, D., and Kolb, A. (2015). Kinect range sensing: Structured-light versus time-of-flight kinect. *CoRR*, abs/1505.05459. URL `http://arxiv.org/abs/1505.05459`.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826.

Tang, H., Scaife, A.M.M., and Leahy, J.P. (2019). Transfer learning for radio galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 488, 3358–3375.

Theodoridis, S. and Koutroumbas, K.D. (2008). Pattern recognition, fourth edition.

Vapnik, V.N. (2000). The nature of statistical learning theory. In *Statistics for Engineering and Information Science*.

Wang, C., Dong, S., Zhao, X., Papanastasiou, G., Zhang, H., and Yang, G. (2020). Saliencygan: Deep learning semisupervised salient object detection in the fog of iot. *IEEE Transactions on Industrial Informatics*, 16, 2667–2676.

Weiss, K.R., Khoshgoftaar, T.M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3, 1–40.

Wofk, D., Ma, F., Yang, T., Karaman, S., and Sze, V. (2019). Fastdepth: Fast monocular depth estimation on embedded systems. *CoRR*, abs/1903.03273.

Xie, S., Girshick, R.B., Dollár, P., Tu, Z., and He, K. (2016). Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431.

Xu, D., Wang, W., Tang, H., Liu, H., Sebe, N., and Ricci, E. (2018). Structured attention guided convolutional neural fields for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3917–3925.

Zhang, D., Han, J., Li, C., Wang, J., and Li, X. (2016). Detection of co-salient objects by looking deep and wide. *International Journal of Computer Vision*, 120, 215–232.

Zoph, B. and Le, Q.V. (2017). Neural architecture search with reinforcement learning. *ArXiv*, abs/1611.01578.