data

MDPI

Article

The Model and Training Algorithm of Compact Drone Autonomous Visual Navigation System

Viacheslav Moskalenko *, Alona Moskalenko *, Artem Korobov and Viktor Semashko

Department of Computer Science, Sumy State University, 40007 Sumy, Ukraine;
a.korobov@cs.sumdu.edu.ua (A.K.); viktor.s.5994@gmail.com (V.S.)
* Correspondence: v.moskalenko@cs.sumdu.edu.ua (V.M.); alenarizhova@gmail.com (A.M.)

Received: 4 November 2018; Accepted: 22 December 2018; Published: 28 December 2018

Abstract: Trainable visual navigation systems based on deep learning demonstrate potential for robustness of onboard camera parameters and challenging environment. However, a deep model requires substantial computational resources and large labelled training sets for successful training. Implementation of the autonomous navigation and training-based fast adaptation to the new environment for a compact drone is a complicated task. The article describes an original model and training algorithms adapted to the limited volume of labelled training set and constrained computational resource. This model consists of a convolutional neural network for visual feature extraction, extreme-learning machine for estimating the position displacement and boosted information-extreme classifier for obstacle prediction. To perform unsupervised training of the convolution filters with a growing sparse-coding neural gas algorithm, supervised learning algorithms to construct the decision rules with simulated annealing search algorithm used for finetuning are proposed. The use of complex criterion for parameter optimization of the feature extractor model is considered. The resulting approach performs better trajectory reconstruction than the well-known ORB-SLAM. In particular, for sequence 7 from the KITTI dataset, the translation error is reduced by nearly 65.6% under the frame rate 10 frame per second. Besides, testing on the independent TUM sequence shot outdoors produces a translation error not exceeding 6% and a rotation error not exceeding 3.68 degrees per 100 m. Testing was carried out on the Raspberry Pi 3+ single-board computer.

Keywords: navigation; visual odometry; convolutional neural network; neural gas; information criterion; extreme learning

1. Introduction

Autonomous navigation is crucial for search, rescue and remote inspection because teleoperation by video stream is extremely challenging when moving close to buildings or trees and in an indoor environment. Besides, the Global Positioning System (GPS) may not be reliable in cases of low satellite coverage and signal multipath interference [1,2]. Using a compact laser scanner can offer an alternative solution. However, such a solution is expensive and the laser has low frequency [3]. The vision-based solution seems particularly suitable to the autonomous navigation in terms of the weight, cost and information it can provide. With an onboard camera, the drone can both estimate the ego-motion and obtain the information on the environment simultaneously.

Visual geometric methods such as direct, semi-direct and feature-based visual odometry are well known [4,5]. However, geometric methods lack robustness for a number of very common effects, such as illumination changes, movement of dynamic objects, different camera calibrations, low-textured environments, noise and motion blur. Data-driven visual odometry using deep learning methods and low-cost cameras is the most promising approach.

Deep data-driven visual odometry has the potential for high learning capability and robustness of camera parameters and challenging environments [5,6]. However, deep models typically require large computational resources for training and inference mode. Besides that, deep models need large labelled datasets for successful training. Complexity reduction of the model can be achieved by narrowing the scope of application to a particular domain of use. However, even a tiny deep model requires significant computational resources and a large labelled training set, which leads to a significant overhead and slow adaptation to the new condition and environment.

The conventional convolutional neural network, comprised of a convolutional filter-based multilayer feature extractor and fully connected neural layers forming the decision rules, is an a very popular model choice for the analysis of image data [7,8]. This being said, conventional convolutional neural networks have a number of disadvantages, such as their inability to analyze the processes occurring in time, and high computational complexity of the learning algorithm based on backpropagation [9]. In addition, making an advance estimate of the required number of neurons in each convolutional layer is complicated. In contrast, a neural network training approach with unsupervised learning methods based on growing neural gas and their modifications offers better prospects [1,10]. Moreover, the convolutional feature extractor combined with sparse coding is characterized by its ability to operate on small datasets or significant target data imbalance in an unlabelled dataset, which can be dealt with by the fine-tuning of feature extractor using a criterion based on efficiency of decision rules [8]. End-to-end fine-tuning in hybrid models is often limited by many inconsistencies. In this case, fine-tuning can be effectively performed by any trajectory-based metaheuristic search algorithm. However, the training process speed depends on computation complexity of model.

The ego-motion estimation task is the most important part of visual navigation process and it can be formulated as a regression analysis for position displacement prediction. The support vector machine and extreme learning machine (ELM) are widely used as regression models in hybrid intelligent systems, especially with constraints on availability of computational resources and the size of labelled dataset. In this case, ELM is characterized by the most rapid training to obtain the least-squares solution of regression problem [11]. In order to eliminate overfitting, which occurs when the number of hidden layer nodes is large, it is relevant to investigate the incremental learning via successive addition of hidden nodes.

The information-extreme classifier of a high-level feature representation is one of the most promising approaches to implementing trainable obstacle avoidance capabilities in terms of computation efficiency and generalizing abilities. The main idea of information-extreme techniques is to transform input space of primary features into binary Hamming space with radial basis decision rules. This approach provides high computational efficiency because simple operation of comparison with thresholds and Hamming distance calculation based on logical XOR operation and non-zero bit counting are used. However, fast threshold optimization methods for feature encoding have not yet been proposed. In this case, feature inductions based on decision tree and boosting are two particularly promising algorithms which can speed up threshold optimization for binary feature encoding [12,13].

In order to implement autonomous navigation and identification of obstacles under the conditions of constrained computational resources and limited labelled training set, this paper proposes a multilayer convolutional sparse coding model for feature extraction from spatial-temporal visual data. In addition, the training algorithms based on unsupervised pretraining and supervised fine-tuning of feature extractor to maximize supervised learning efficiency of information-extreme classifier and extreme-learning regression model are also proposed.

2. Data Description and Data Model Design

To prepare the input mathematical description of the intelligent information system, KITTI VO/SLAM benchmarks 07 and 09 from Kaggle were used. The datasets contain the image frame sequence captured by the moving video camera and the corresponding tri-axis ground truth movement data reported by GPS and LiDaR (dataset: http://www.cvlibs.net/datasets/kitti/raw_data.php?type=residential, https://vision.in.tum.de/data/datasets/visual-inertial-dataset; dataset license: Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License, Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License, Creative Commons 4.0 Attribution License CC BY 4.0) [8]. Since this dataset was recorded at a relatively low frame rate (10 frame per second) by driving in urban areas many dynamic objects and the speed of the driving was up to 90 km/h, some image pre-processing was necessary. This was achieved by applying the resizing, gray scaling and un-distorting functions from an open-source robotic operation system [9]. Whilst it required the camera calibration parameters, high accuracy was not needed. The training and test video sequences from the KITTI Vision dataset were used. The images were downscaled to 200×200 due to memory and computation resource limitations.

The main adopted methodological principles of data analysis model design are as follows:

- using more than 2 subsequent frames to determine the movement between the two last frames, which allows for better context detection and increase of the decision accuracy;
- hierarchical feature representation, which allows describing complex features with a smaller number of parameters;
- avoidance of subsampling for preservation of larger quantity of information (data shifts, deformation and scale), which is important for regression analysis of camera movement in time and space, especially in absence of end-to-end learning;
- low-computational-complexity decisions rules, which reduce the time needed for efficiency evaluation and feature extractor tuning.

The trainable model of the visual navigation should provide obstacle avoidance and control of UAV's (unmanned aerial vehicle) own position through odometry. The proposed model of visual navigation can be trained as an independent system or as an auxiliary parallel error corrector for geometric methods of visual odometry. In case of auxiliary option, both models must be synchronized.

Figure 1 depicts the schematic of the navigational system for a compact drone. A convolutional neural network which takes inputs in a form of a multichannel image formed by a series sampling of successive video frames in grayscale format is used to extract the feature representation of visual observations. A multilayered structure of Convolutional Neural Network (CNN) is used to form a high-level feature representation. Convolutional filters in the proposed CNN are subjected to unsupervised training layer by layer. An information-extreme classifier is then trained in supervised mode on the samples encoded by the corresponding high-level features. This classifier is then used to predict obstacles and for output of the corresponding response. The ELM-based regression model uses the visual features to predict the relative position change between two consecutive frames. From five consecutive observations, this component predicts position displacement along the x-axis and y-axis, and orientation (yaw) of the second frame with respect to the first frame.

The 4-layer architecture of the convolutional neural network is shown in Figure 2. The first layer of the CNN contains standard convolutional filters with different kernels: 5×5, 3×3 and 1×1. The parameter K_2 regulates the number of filters. To keep the size of feature maps formed by multiple-scale filters unchanged, the technique of padding by zeros is used [8]. Corresponding stride parameters of scanning a feature map with multiple-scale filters for the second and third layers are 3 and 2, respectively. The Orthogonal Matching Pursuit (OMP) algorithm and rectifier function,$y = \max(0, x)$, are used for calculating each multi-channel pixel of feature map [10]. To prevent information loss, the feature map can be doubled using the function $y = \{\max(0, x),\ \max(0, -x)\}$.

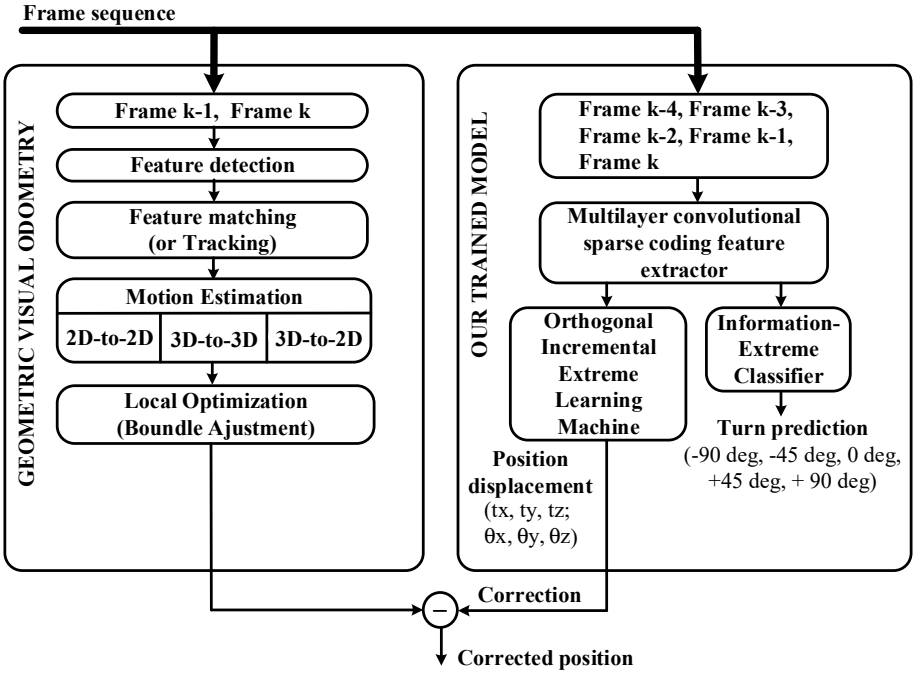


Figure 1. Model for classical odometry with the proposed model as a parallel error corrector.

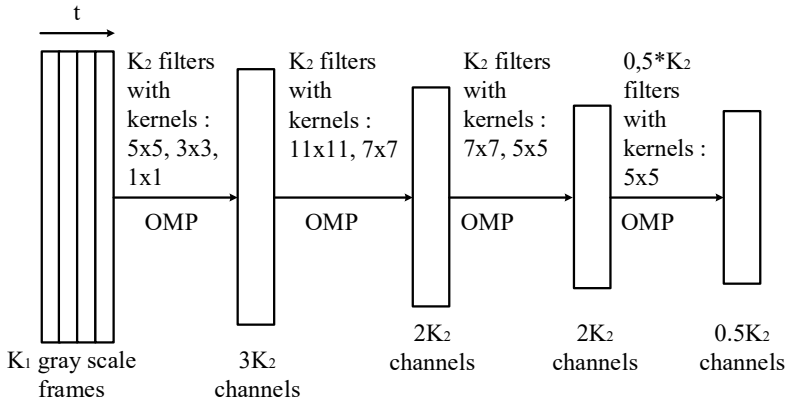


Figure 2. The architecture of the convolutional sparse coding model for visual feature extraction in the navigation system.

To train the regression model $y = f(x)$ represented by the single-hidden-layer feedforward network (SHFN), output variable $y_j \in R^M$ which corresponds to position displacement in terms of translation vector t and rotation vector θ of the camera $y = (y_1, y_2, \ldots, y_n)^T = (t, \theta)^T$, a set of $\{(x_j, y_j) | x_j \in R^N, y_j \in R^M, 1 \le j \le n\}$ training data consisting of visual features, is used, where $x_j = (x_{j1}, x_{j2}, \ldots, x_{jN})^T$. The SHFN with R additive hidden nodes and activation function φ(x) can be represented by:

$$\sum_{r=1}^{R} \beta_j \varphi(w_r^T x_j + b_r) = o_j, \; 1 \le j \le n \tag{1}$$

where $w_r = (w_{r1}, w_{r2}, \ldots, w_{rN})^T$ is the weight vector linking the input layer with the rth hidden node, b_r is the threshold of the rth hidden node, $\beta_r = (\beta_{r1}, \beta_{r2}, \ldots, \beta_{rM})$ is the weight vector linking the output layer with the rth hidden node, o_j is the output of the network with respect to the input vector x_j, and $\varphi(x)$ is the activation function that can be any bounded non-constant piecewise continuous functions.

The network with R hidden nodes can approximate these N samples with zero error when all the parameters are allowed to be adjusted freely, i.e., there exist β_r, w_r and b_r. The above n equations can be compactly rewritten as the matrix equation $H\beta = Y$, where

$$H = \begin{bmatrix} \varphi(w_1^T x_1 + b_1) & \dots & \varphi(w_R^T x_1 + b_R) \\ \dots & \dots & \dots \\ \varphi(w_1^T x_n + b_1) & \dots & \varphi(w_R^T x_n + b_R) \end{bmatrix}_{n \times R} \tag{2}$$

$$\beta = \begin{bmatrix} \beta_1^M \\ \dots \\ \beta_R^M \end{bmatrix}_{R \times M} \tag{3}$$

$$Y = \begin{bmatrix} y_1^M \\ \dots \\ y_n^M \end{bmatrix}_{n \times M} \tag{4}$$

Here, H is called the hidden layer output matrix of the SHFN.

The information-extreme classifier under inference mode makes decision on belonging of input datapoint x with appropriate binary representation b to one class of turn from set $\{X_z^o | z = \overline{1, Z}\}$, according to the maximum value of membership function $\mu_z(b)$ through the expression $\underset{z}{\text{argmax}}\{\mu_z(b)\}$. In this case, membership function $\mu_z(b)$, the optimal container of which has support vector b_z^* and radius d_z^*, is derived from:

$$\mu_z(b) = \exp\left(-\sum_{i=1}^{N_2} b_i \oplus b_{z,i}^* / d_z^*\right). \tag{5}$$

Binary encoding of datapoint x_j is carried out by concatenation of decision paths from ensemble of trees T_1, ... , T_k. Datapoint x_j is classified in the leaf node by trees. Each decision node receives a unique identifier. If a test is satisfied in a node, then the corresponding bit is asserted. Finally, the encodings for each tree are combined by concatenation (or more generally by hashing the features ID onto a smaller-dimensional space) [143,134].

3. Training Algorithm Design

The main adopted methodological principles of machine learning algorithm design are as follows:

- uniting the competitive learning principles with sparse coding to reduce the training data sample size requirements and improved convergence during the feature extractor pretraining;
- training decision rules on the basis of fast randomized least-squares solvers (Gram–Schmidt process, and Moore–Penrose pseudoinverse) or decision trees-inspired techniques (random forest, random subspace, and adaptive boosting);
- metaheuristic optimization for harmonization of hybrid data analysis model components and improved convergence to the global model efficiency criterion optimum;
- complex criterion of the model efficiency is formed as a multiplicative aggregation of partial efficiency criteria aimed at the optimal feature extractor configuration which simultaneously improves all partial criteria.

The aim of the machine learning process of the navigation system is to determine the optimal vector of parameter g, which maximize the complex criterion as follow:

$$J = \frac{\overline{E}}{E_{\max}} \cdot \frac{\varepsilon_{\min}}{\varepsilon} \cdot \frac{C_{\min}}{C}, \tag{6}$$

$$g^* = \underset{G}{\text{argmax}}\{J(g)\}\,, \tag{7}$$

where $\overline{E}$—learning effectiveness of recognition of obstacle averaged by the set of classes learning criterion; ε—regression mean square error for the change of camera coordinates in space; C—computational complexity criterion of feature extraction algorithms; $E_{\max}$, $\varepsilon_{\min}$, and $C_{\min}$—the maximum possible value of the classifier training efficiency informational criterion, the minimum allowable values of regression model error and computational complexity of the system's algorithms criterion, respectively; G—admissible domain of parameters affecting feature extraction and decision-making.

It is proposed to carry out unsupervised training of feature extractor using the growing sparse coding neural gas technique, based on the principles of growing neural gas and sparse coding. In this case, the dataset for training convolutional filters is formed by cropping the 3-dimensional patches of input images or feature maps. The shape of each patch is equivalent to corresponding filter shape. These patches reshape to 1-dimensional vectors, arriving at the input of the algorithm of growing sparse coding neural gas, and the basic stages of which are given below:

1. the counter of learning vectors t is initialized to be 0;
2. two initial nodes (neurons) w_a and w_b are assigned by random choice out of the training dataset. Nodes w_a and w_b are connected by the edge, the age of which is zero. These nodes are considered non-fixed;
3. the following vector x, set to the unit length (L2-normalization), is selected;
4. each basis vector $w, k = \overline{1, M}$ is set to the unit length (L2-normalization);
5. the measure of similarity of input vector x to basis vectors $w_{s_k} \in W$ is calculated for sorting

$$-\left(w_{s_0}^T x\right)^2 \leq \ldots \leq -\left(w_{s_k}^T x\right)^2 \leq \ldots \leq -\left(w_{s_{M-1}}^T x\right)^2, \tag{8}$$

   where w_{s0} is the nearest node and w_{s1} is the second nearest node by proximity node;
6. the age of all incidents w_{s0} is increased by unity;
7. if w_{s0} is fixed, go to step 9, otherwise, go to step 10;
8. if $\left(w_{s0}^T x\right)^2 \geq v$, go to step 12. Otherwise, a new non-fixed neuron w_r is added to the point which coincides with the input vector $w_r = x$; besides, a new edge that connects w_r and w_{s0} is added, and then go to step 13;
9. node w_{s0} and its topological neighbors (the nodes connected with it by the edge) are displaced in the direction to input vector x according to the Oja's rule [13] by:

$$\Delta w_{s0} = \varepsilon_b \eta_t w_{s0}^T x (x - w_{s0}^T x w_{s0}), \tag{9}$$

$$\Delta w_{sn} = \varepsilon_n \eta_t w_{sn}^T x (x - w_{sn}^T x w_{sn}), \tag{10}$$

$$0 < \varepsilon_b \ll 1,\ 0 < \varepsilon_n \ll \varepsilon_b, \tag{11}$$

$$\eta_t := \eta_0 \left(\eta_{final} / \eta_0\right)^{t/t_{\max}}, \tag{12}$$

   where Δw_{s0}, and Δw_{sn} are the vector of correction of weight coefficients of the neuron-winner and its topological neighbors, respectively; ε_b and ε_n are the constants of the updated force of weight coefficients of the neuron-winner and its topological neighbors, respectively; η_0, η_t, and η_{final} are the initial, current and final value of learning rate, respectively;
10. if $\left(w_{s0}^T x\right)^2 \geq v$, the neuron w_{s0} is labelled as fixed;
11. if w_{s0} and w_{s1} are connected by the edge, its age is nulled, otherwise, a new edge with the zero age is formed between w_{s0} and w_{s1};
12. all edges in the graph with the age of more than $a_{\max}$ are removed. In the case when some nodes do not have any incident edges (become isolated), these nodes are also removed;
13. if $t < t_{\max}$, proceed to step 15, otherwise, increment the step counter t by 1 described as $t := t + 1$ and proceed to step 3;

14. if all neurons are fixed, the algorithm implementation stops, otherwise, proceed to step 3 and a new learning epoch begins (repetition of the training dataset).

The information-extreme classifier that evaluates belonging of j^{th} datapoint x_j with N_1 features to one of the Z classes performs feature encoding using boosted trees and decision rules constructed in radial basis of binary Hamming space. In this case, there is the training set, $D = \{x_j,\ y_j \mid j = \overline{1,n}\}$, where n is the size of dataset and y_j is the label of j^{th} pixel, which correspond to one class from set of classes $\{X_z^o | z = \overline{1,Z}\}$. The training of boosted information-extreme classifier is performed according to the following steps:

1. Initialize weight $w_j = 1/n$.
2. Initialize step counter $k = 1$.
3. Bootstrap D_k from D using probability distribution $P(X = x_j) = w_j$.
4. Train decision tree T_k on D_k using entropy criterion to measure the quality of split.
5. Perform binary encoding of datapoint x_j from D using concatenation of results from $T_1, \ldots, T_k$ trees. In this case, each decision tree produces a binary code where each non-zero bit corresponds to a node of decision path from root to leaf node [12,14]. The output of this step is a binary matrix $\{b_{z,s,i} | i = \overline{1,N_2};\ s = \overline{1,n_z}; z = \overline{1,Z}\}$, where N_2 is the number of induced binary features and n_z is the number of samples corresponding to class X_z^o. Hence, the equality $n = \sum\limits_z n_z$ condition is met.
6. Build information-extreme decision rules in radial basis of binary Hamming space and compute optimal information criterion:

$$E_z^* = \max_{\{d\}} E_z(d) \tag{13}$$

where $\{d\} = \{0,\ 1,\ \ldots,\ \left(\sum\limits_i b_{z,i} \oplus b_{c,i} - 1\right)\}$ is a set of concentric radiuses with center b_z (support vector) of data distribution in class X_z^o, which is computed using the rule:

$$b_{z,i} = \begin{cases} 1, \text{if} \frac{1}{n_z} \sum\limits_{s=1}^{n_z} b_{z,s,i} > \frac{1}{Z} \sum\limits_{c=1}^{Z} \frac{1}{n_c} \sum\limits_{s=1}^{n_c} b_{c,s,i}; \\ 0, \text{otherwise.} \end{cases} \tag{14}$$

where E_z—training efficiency criterion of decision rule for class X_z^o, which is computed as the normed modification of the Kullback's information measure [13]:

$$E_z = \frac{1 - (\alpha_z + \beta_z)}{\log_2(2 + \varsigma) - \log_2 \varsigma} \cdot \log_2 \left[\frac{2 - (\alpha_z + \beta_z) + \varsigma}{(\alpha_z + \beta_z) + \varsigma} \right], \tag{15}$$

where α_z, and β_z are the false-positive and false-negative rates of classification of input vectors as belonging to the class X_z^o; ς is any small non-negative number, introduced to avoid uncertainty when being divided by zero.

A conventional way to increase the learning efficiency is a reduction of the multi-class classification problem to the series of the two-class one by the principle "one-against-all". In this case, to avoid the problems of class imbalance, due to the majority of negative datapoints in datasets, a synthetic class is an alternative to class X_z^o. The synthetic class is represented by n_z datapoints of the remaining classes, which are the closest to support vector b_z, where n_z is the volume of the training dataset of class X_z^o.

7. Test obtained information-extreme rules on dataset D and compute the error rate for each sample from D. Under the inference mode, decision on belonging of datapoint b to one class from set $\{X_z^o | z = \overline{1,Z}\}$ is made according to the maximum value of membership function $\mu_z(b)$ through the expression $\underset{z}{\text{argmax}}\{\mu_z(b)\}$. In this case, membership function $\mu_z(b)$ of binary representation

b of input datapoint x to class X_z^o, the optimal container of which has support vector b_z^* and radius d_z^*, is derived from Equation (5).

8. Update $\{w_j\}$ proportional to errors on datapoint x_j.
9. If $|E_k^* - E_{k-1}^*| < \varepsilon$, abort the loop.
10. k = k + 1.
11. If $k \leq K$, where K is the maximum number of trees, go to step 2.

Orthogonal incremental ELM is proposed to be used as a regression model. It avoids redundant nodes and obtains the least-squares solution of equation $H\beta = Y$ by incorporating the Gram–Schmidt orthogonalization method into well-known incremental ELM. rigorous proofs in theory of convergence of orthogonal incremental extreme learning are given by Ying [11]. The training of orthogonal incremental ELM is performed according to the following steps:

1. Set the maximum number of iterations $L_{\max}$ and expected learning accuracy E_0.
2. For $L = 1, \ldots, L_{\max}$ do
3. Increase by one the number of hidden nodes: $r = r + 1$.
4. Randomly generate one hidden node and calculate its output vector h_r.
5. If $r = 1$ then $v_r = h_r$ else

$$v_r = h_r - \frac{\langle v_1,\ h_r \rangle}{\langle v_1,\ v_1 \rangle} v_1 - \frac{\langle v_2,\ h_r \rangle}{\langle v_2,\ v_2 \rangle} v_2 - \ldots - \frac{\langle v_{r-1},\ h_r \rangle}{\langle v_{r-1},\ v_{r-1} \rangle} v_{r-1} \tag{16}$$

6. If $||v_r|| \geq \varepsilon$ calculate the output weight for the new hidden node:

$$\beta_r = v_r^T E / (v_r^T v_r) \tag{17}$$

and calculate the new residual error:

$$E = E - v_r \beta_r \tag{18}$$

else $r = r - 1$.

7. If $||E|| \geq E_0$, abort the loop.

In order to approximate the global optimum of complex criterion during training of decision rules and feature exctactor fine-tuning, the use of simulated annealing metaheuristic algorithm is proposed [13]. The efficiency of the simulated annealing algorithm depends on the implementation of the *create_neighbor_solution* procedure, forming a new solution s_i on the ith iteration of the algorithm. Figure 3 shows a pseudocode of the simulated annealing algorithm, which is implemented by the *epochs_max* iterations, each of which function f is calculated by passing a labelled training dataset through the model of the system of detection and calculation of a complex criterion (6).

An analysis of the pseudocode in Figure 3 shows that current solution $s_{current}$, in relation to which new best solutions s_{best} are sought for, is updated in case of providing a new solution of the criterion increase (1), or randomly from the Gibbs distribution [13]. In this case, an initial search point that is formed by the *create_initial_solution* procedure can be either randomly generated or a result of the preliminarily training by another algorithm. To generate new solutions in the *create_neighbor_solution* procedure, it is proposed to use the simplest non-adaptive algorithm, which can be represented as:

$$s_{current} = s_{current} + uniform\_random(-1, 1) \cdot step\_size, \tag{19}$$

where *uniform_random* is the function of generation of a random number from the uniformed distribution from the assigned range; *step_size* is the size of a range of the search for new solutions, neighboring to $s_{current}$.

$s_{current} \leftarrow create\_initial\_solution()$
$s_{best} \leftarrow s_{current}$
$T \leftarrow T_0$
$c \leftarrow \varepsilon,\ 0 < \varepsilon < 1$
$for(i = 1\ to\ epochs\_\max)$
  $s_i \leftarrow create\_neighbor\_solution(s_{current})$
  $if\ f(s_i) \geq f(s_{current})$
    $s_{current} \leftarrow s_i$
    $if\ f(s_i) \geq f(s_{best})$
      $s_{best} \leftarrow s_i$
    $end\ if$
  $elseif\ \exp\left(\frac{f(s_{current}) - f(s_i)}{T}\right) > uniform\_random(0,1)$
    $s_{current} \leftarrow s_i$
  $end\ if$
  $T \leftarrow c \times T$
$end\ for$
$return(s_{best})$

Figure 3. Pseudocode of metaheuristic simulated annealing algorithm.

4. General Implementation and Setup Remarks

Autonomous navigation implementation is foreseen on 1 or 2 Raspbery Pi 3+ single-board 64-bit computers. They are equipped with CPU Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz and have 1GB of RAM. The official Raspbian operating system is 32-bit only and various other 64-bit have numerous problems with input–output device drivers, and hence, Debian Stretch is compiled for 64-bit SoC Broadcom BCM2837 Cortex-53 chip.

Training and inference regimes were implemented using Opencv 3.4, Tensorflow HRT with contributed modules and Robot Operating System Melodic Morenia (for arm64) with ORB-SLAM libraries. The libraries were optimised using single instruction, multiple data (SIMD) technologies (NEON-instruction). Jemalloc was installed to manage a given memory space. Image preprocessing was done with Opencv. Orthogonal Matching Pursuit, decision trees, ELM and growing sparse coding neural gas were implemented on Tensorflow framework for both training and inference.

For the purposes of testing the error corrector regime, dataset pre-processing and subsequent processing based on geometric odometry ORB-SLAM with a constraint on the maximum keypoint number limited to 100 was performed. The results were saved in a text file. After this, the proposed model was trained on a half of the KITTI dataset sequence. It is worth mentioning that for classifier training purposes, manually labelled turn samples taken from various sequences of KITTI dataset were used. A classifier in this case plays a regularizing role. The last step is the model launched in the inference mode to generate the final result taking into account ORB-SLAM results.

5. Simulation Results and Discussion

Parameters K_1 and K_2 values affect both the informative nature of the feature representation and the degree of computational complexity. For the purposes of this paper, computational complexity is measured by the quantity of "Mul" and "Add" operations performed at the time of convolutional operations with an image or a feature map. The computational complexity for the network architecture in Figure 2 can be calculated as:

$$S = K_2(2706472K_1 + 4438784K_2) \tag{20}$$

The optimal configuration of the convolutional extractor or the classifier and the regression model may differ, as they are responsible for different tasks. A complex criterion (6) offers an acceptable compromise between accuracy of the decision rules and the computational complexity of the visual feature extractor.

The set of recognition classes $\{X_z^o\}$ describes the characteristic obstacles and the corresponding response commands, and has a power $Z = 5$. The first class of recognition X_1^o characterizes the forward movement without turning. Classes X_2^o and X_3^o correspond to the left turn of 45 and 90 degrees, respectively. Classes X_4^o and X_5^o correspond to the right turn of 45 and 90 degrees, respectively. The size of the corresponding training and test sets for each class is $n_z = 30$. The regression model was trained on a half of the total length of the KITTI dataset sequence. KITTI-0.7 and KITTI-0.9 were used. The length of KITTI-0.7 sequence is equal to 1100 and the length of KITTI-0.9 sequence is equal to 1590.

It is proposed first to train the model using the unsupervised pretrained feature extractor via growing sparse coding neural gas without fine-tuning. In this case, three fixed values of the training dataset reconstruction parameter ν were used during training. This parameter directly affects the number of feature map channel K_2.

In order to improve the results of machine learning of the visual navigation system, informativeness of feature description was increased by fine-tuning of the unsupervised trained convolutional filters parameters, which are depicted in Figure 2. In this case, the following parameters of simulated annealing algorithm were used: c = 0.98, T_0 = 10, *epochs_max* = 3000, and *step_size* = 0.001. Each fine-tuning step involved a re-training of a regression model and classifier. A simulation was performed for the three fixed values of parameters K_1 and K_2 (Table 1) to evaluate the tendency of change in average values of the partial and complex criteria during the growth of such parameters affect the size of the feature extractor (Figure 2). The optimal parameter values were determined for the KITTI-07 open dataset.

Table 1. Dependence of partial and complex criteria on feature extractor hyperparameters K_1 and K_2.

v	K_1	K_2	$\overline{E}/E_{\max}$	$\varepsilon_{\min}/\varepsilon$	$C_{\min}/C$	J
0.6	3	18	0.083	0.112	1.000	0.009296
0.6	5	18	0.101	0.188	0.827	0.015703
0.6	7	18	0.098	0.200	0.705	0.013818
0.7	3	21	0.28	0.688	0.297	0.057214
0.7	5	21	0.29	0.756	0.264	0.057879
0.7	7	21	0.29	0.775	0.238	0.053491
0.8	3	46	0.39	0.968	0.082	0.030957
0.8	5	46	0.55	1.000	0.077	0.04235
0.8	7	46	0.51	1.000	0.072	0.03672

The analysis of Table 1 shows that increases in parameter values K_1 and K_2 tend to increase both the reliability and computational complexity (20) of the decision rules of the classifier and the regression model. At the same time, the increase in parameter K_1 has a negligible effect on the overall classifier efficiency due to the corresponding decrease in the learning efficiency with a substantial growth of the feature space, while the regression error is equally sensitive to the values of parameters K_1 and K_2.

However, considering that, as K_1 and K_2 grow, the reliability of decision rules increases at a slower rate than computational complexity, and the use of complex criterion J offers a pragmatic compromise solution. The optimal parameter values are $K_1^* = 5$ and $K_2^* = 21$.

The average value of the information criterion of functional efficiency for the optimal configuration of the feature extractor is $\overline{E} = 0.29$. This corresponds to 95.2% accuracy for the training set, and 94% for the test dataset. Figure 4 depicts the change in average information efficiency criterion (4) as a function of the number of iterations of the simulated annealing algorithm.

The analysis of Figure 4 shows that after the 1000th iteration, the rate of increase of the information criterion (15) reduces, and after the 2500th iteration, the information criterion remains virtually unchanged. This indicates that the further increase in the information criterion can be obtained only by expanding the informative nature of the features by increasing values of K_1 and K_2 or improving the architecture of the feature extractor (Figure 2).

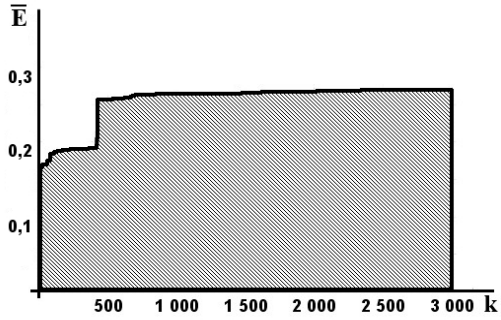


Figure 4. A graph of the change of the average information efficiency criterion (15) in dependence of the number of iterations of the simulated annealing algorithm.

For a visual assessment of the effectiveness of the machine learning of the navigation system, a ground truth trajectory constructed using annotation data from the KITTI dataset can be compared with a reconstructed trajectory obtained using a trained model. The regression model produces yaw and translation along the x-axis and y-axis. To obtain the rotation matrix R from the rotation vector θ, the Rodrigues Transformation was used [9]. To reconstruct the movement trajectory, the coordinate t_{pos}[i] and orientation R_{pos}[i] of camera for ith frame relative to the start position were calculated as:

$$R_{pos}[i] = R \cdot R_{pos}[i-1] \tag{21}$$

$$t_{pos}[i] = t_{pos}[i-1] + t \cdot R_{pos}[i] \tag{22}$$

Figure 5a shows the result of using feature-based odometry from the ORB-SLAM system [5,6]. The maximum number of keypoints is limited to 100 to provide the required frame rate in case of onboard system without graphics processing unit (GPU) acceleration.

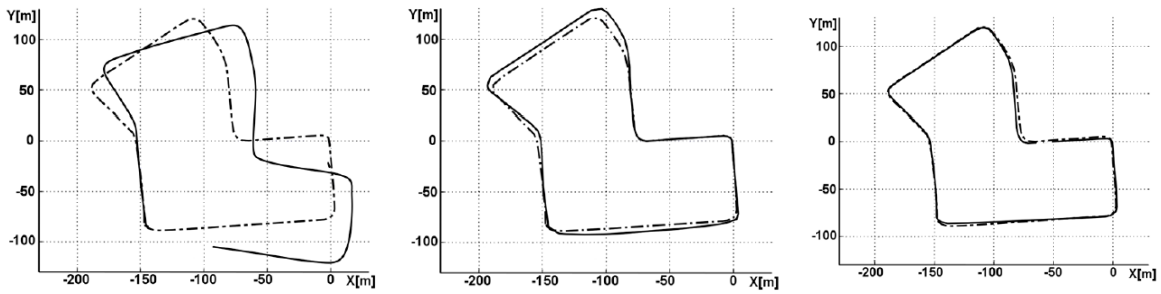


Figure 5. Ground truth (dotted line) and reconstructed (solid line) trajectories for KITTI-07: (**a**) feature-based monocular odometry; (**b**) proposed model; (**c**) parallel error correction for feature-based monocular odometry using the proposed model.

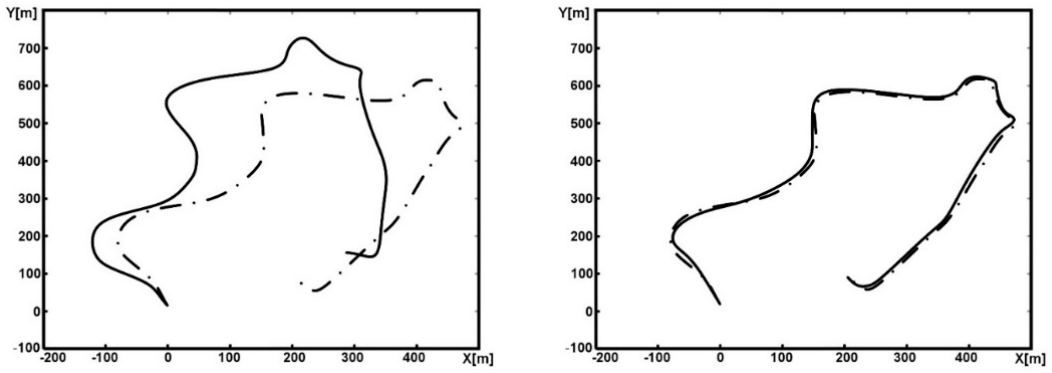
The analysis of Figure 5a shows that the reconstructed trajectory has a significant error. Figure 5b shows the result of using proposed model trained and tested on the same datasets. In this case, the error of reconstruction is much smaller. Figure 5b shows the result of using feature-based odometry from ORB-SLAM with parallel correction based on the proposed model. As can be seen, the accuracy of the reconstruction is high.

Table 2 shows that the proposed approach performs a better trajectory reconstruction than the well-known ORB-SLAM. In particular, for sequence 7 from the KITTI dataset, the translation error of length is reduced by nearly 65.6% under the same frame rate (10 frames per second). Testing was carried out on the single-board Raspberry Pi 3+ Computer.

Table 2. Mean results in the KITTI dataset.

Sequence	Proposed Model		ORB-SLAM with 100 Keypoints Limitation	
	***t*, %**	***θ*, degree per 100 m**	***t*, %**	***θ*, degree per 100 m**
KITTI-07	1.57	2.60	2.24	7.56
KITTI-09	1.49	2.12	1.64	6.89

Figure 6a shows that the trained model on the KITTI-07 dataset has poor performance when tested on another dataset (KITTI-09). Figure 6b shows that re-training of the model on the new dataset considerably increases the accuracy of the trajectory reconstruction from the dataset.



(**a**) (**b**)

Figure 6. Ground truth (dotted line) and reconstructed (solid line) trajectories for KITTI-09: (**a**) trained model on KITTI-07; (**b**) after re-training on KITTI-09.

Thus, the results obtained on open datasets testify to the suitability of the model for practical use. To avoid overfitting, the complexity of model is optimized. However, the limited capacity of the optimized model requires its re-training when changing the domain area of use.

For an independent testing of trained model TUM set sequences shot outdoors were used. ORB-SLAM was tested on the same sequences with a limit of a 100 keypoints for comparison (Table 3).

Table 3. Mean results in the TUM dataset.

Sequence	Proposed Model		ORB-SLAM with 100 Keypoints Limitation	
	***t*, %**	***θ*, degree per 100 m**	***t*, %**	***θ*, degree per 100 m**
TUM-outdoors 5	4.31	3.60	6.24	7.56
TUM-outdoors 6	5.42	3.68	7.64	8.11
TUM-outdoors 7	4.33	3.59	6.44	7.93
TUM-outdoors 8	5.01	3.61	7.12	7.99

Analysis of Table 3 demonstrates that ORB-SLAM translation error for TUM sequences exceeds the translation error of the proposed model more than 1.4 times. The ORB-SLAM rotation error exceeds the corresponding rotation error of the proposed model more than 2 times. Accuracy of the proposed model, however, tested on the independent sequence is lower than on the KITTI dataset, which was used in training, since at the beginning of TUM sequences indoor video footage, very different from the training data, was used.

Thus, the proposed model and training algorithm of the autonomous robot navigation system were tested on open datasets. Testing on the independent TUM sequence shot outdoors produces a translation error not exceeding 6% and a rotation error not exceeding 3.68 degrees per 100 m. Accuracy can be further improved by additional training on the samples of such sequences.

6. Conclusions

The scientific novelty of the results lies in the following:

- The model of autonomous navigation system for a compact drone is proposed. The model consists of a 4-layer convolutional sparse coding network with series of successive frames as an input and multi-scale convolutional filters, ELM for estimating position displacement and an information-extreme classifier for predicting turns;
- a complex criterion for evaluating effectiveness of the model which takes into account computation complexity of feature extractor and accuracy of decision rules;
- the training algorithm consists of 3 parts. The first part is an unsupervised training of feature extractor with series of successive frames as an input. The second part is a supervised training of information-extreme classifier for turn prediction with fine-tuning of feature extractor using a metaheuristic search algorithm. The last part is a supervised training of ELM for position displacement prediction or correction using the feature extractor.

The significance of the results for the unmanned aviation sector lies principally in the formulation of a modern scientific and methodological basis for the design of trainable autonomous navigation systems for compact drones, which operate under computational resource and information constraints. Simulation results validate the high efficiency of the resulting decision rules for coordinate determination and obstacle recognition based on the video footage. Our approach performed a better trajectory reconstruction than the well-known ORB-SLAM. In particular, for sequence 7 from the KITTI dataset, the translation error is reduced by nearly 65.6% under the same frame rate. Besides, testing on the independent TUM sequence shot outdoors produces a translation error not exceeding 6% and a rotation error not exceeding 3.68 degrees per 100 m. Testing was carried out on the single-board Raspberry Pi 3+ Computer.

Author Contributions: conceptualization and methodology, V.M.; writing of review and editing, A.M.; software and validation, A.K.; visualization, V.S.

Funding: The research was concluded in the Intellectual Systems Laboratory of Computer Science Department at Sumy State University with the financial support of the Ministry of Education and Science of Ukraine in the form of a President's of Ukraine grant for competitive projects F75/144-2018 of the State Fund for Fundamental Research.

Acknowledgments: The authors thank the organizers of the DSMP'2018 conference for the opportunity to publish the article, as well as reviewers for the relevant comments that helped to better present the paper's material.

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

References

1. Moskalenko, V.; Moskalenko, A.; Korobov, A.; Boiko, O.; Martynenko, S.; Borovenskyi, O. Model and Training Methods of Autonomous Navigation System for Compact Drones. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; pp. 503–508.
2. Suwandi, B.; Kitasuka, T.; Aritsugi, M. Low-cost IMU and GPS fusion strategy for apron vehicle positioning. In Proceedings of the 2017 IEEE Region 10 Conference (TENCON), Penang, Malaysia, 5–8 November 2017; pp. 449–454.
3. Wang, S.; Deng, Z.; Yin, G. An Accurate GPS-IMU/DR Data Fusion Method for Driverless Car Based on a Set of Predictive Models and Grid Constraints. *Sensors* **2016**, *16*, 280. [CrossRef] [PubMed]
4. Mary, B.A.; Gerhard, P.H. Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter. *Sensors* **2017**, *17*, 2164. [CrossRef]
5. Folkesson, J.; Leederkerken, J.; Williams, R.; Patrikalakis, A.; Leonard, J.A. A Feature Based Navigation System for an Autonomous Underwater Robot. In *Springer Tracts in Advanced Robotics, Proceedings of the Sixth Edition of Field and Service Robotics, Chamonix, France, 9–12 July 2007*; Springer: Berlin, Germany, 2008; pp. 105–114. [CrossRef]

6. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
7. Dorian, G.-L.; Marta, S.; Juan, D.T.; Montiel, J.M.M. Real-time monocular object SLAM. *Robot. Auton. Syst.* **2016**, *75*, 435–449. [CrossRef]
8. Ayoul, T.; Buckley, T.; Crevier, F. UAV Navigation above Roads Using Convolutional Neural Networks. Available online: http://cs231n.stanford.edu/reports/2017/pdfs/553.pdf (accessed on 24 December 2018).
9. Mohanty, V. DeepVO: A Deep Learning Approach for Monocular Visual Odometry. Available online: https://arxiv.org/pdf/1611.06069.pdf (accessed on 24 December 2018).
10. Labusch, K.; Barth, E.; Martinetz, T. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing* **2009**, *72*, 1547–1555. [CrossRef]
11. Zou, W.; Xia, Y.; Li, H. Fault Diagnosis of Tennessee-Eastman Process Using Orthogonal Incremental Extreme Learning Machine Based on Driving Amount. *IEEE Trans. Cybern.* **2018**, *48*, 3403–3410. [CrossRef] [PubMed]
12. Moskalenko, V.V.; Korobov, A.G. Information-extreme algorithm of the system for recognition of objects on the terrain with optimization parameter feature extraction. *Radio Electron. Comput. Sci. Control* **2017**, *2*, 38–45. [CrossRef]
13. Moskalenko, V.; Dovbysh, S.; Naumenko, I.; Moskalenko, A.; Korobov, A. Improving the effectiveness of training the on-board object detection system for a compact unmanned aerial vehicle. *Eastern-Eur. J. Enterp. Technol.* **2018**, *4*, 19–26. [CrossRef]
14. Vens, C.; Costa, F. Random Forest Based Feature Induction. In Proceedings of the IEEE 11th International Conference on Data Mining, VA, Canada, 11–14 December 2011; pp. 744–753. [CrossRef]