2022/23

Small obstacle detection on road surfaces using RBM road reconstruction

Pattern Recognition Systems Laboratory – 2022/2023

Keresztes Beáta

Group: 30441

Keresztes Beata Pattern Recognition Systems Laboratory – 2022/2023
Group: 30441

Small obstacle detection on road surfaces using RBM road reconstruction

Small, unexpected objects on the road can pose a hazard and cause accidents when driving at high speed. Detecting them in time is difficult using sensors such as Lidar or Radar due to their limited detection range and poor resolution. A solution would be to use the neural network, known as Restricted Boltzmann Machine, which is able to reconstruct the texture of the road from randomly sampled patches of a color image. The difference (reconstruction error) between the original input image and the reconstructed road surface could be used for the segmentation of road anomalies. The results obtained could be more accurate than by using other segmentation algorithms or image processing techniques.

Introduction

The objective is to detect small, non-road items from 2D color images of roads. The approach consists of 2 phases: reconstructing the texture of the road surface, and generating an error heatmap which can be used as a segmentation mask to detect obstacles.

1. Preprocessing

In the preprocessing phase, the input images are resized and cropped with a mask of size 500x500 to extract the center region which contains the road area. Then the images are split into smaller patches of size 50x50, mean centered and normalized. When training these patches will be further processed, extracting randomly from each a smaller patch of size PxP to be processed faster. Generally, P with a size of 8 performed well enough for the majority of the cases.

2. Training the model

The model is trained and evaluated on the Lost and Found dataset, which contains over 2000 images in total, obtained from multiple video sequences. Each image has an associated segmentation mask for all classes of objects and one for distinguishing the road surface.

2.1. Introducing the RBM

The RBM is an energy-based model and it computes the free energy for the initial input to the visible layer and the reconstructed input, then it defines the loss as the absolute difference between these energy values. It assigns lower energy values to more relevant features and higher energy values for less important features.

Free energy is the energy that a single configuration would need to have the same probability as all configurations containing the input feature vector v.

Two passes are performed:

- **forward pass**: the input to the visible layer is the input image v
- **backward pass**: the input the hidden layer is the probability of obtaining v with the configuration of the input features

The objective is to obtain the optimal values for the hyperparameters (weight matrix, biases) so that the reconstruction error would be minimum. SGD is used to minimize the reconstruction error and obtain a more accurate reconstruction of the input, by updating the hyperparameters after visiting each instance.

The reconstruction error is computed as the difference between the pixel-wise mean-square error of the inputs to the visible layer (original image) and the output of the reconstructed image. The mean of the reconstruction errors of each training instance represents the training loss.

Another approach to train the RBMs is using Gibbs sampling and the Contrastive divergence algorithm. The learning process consists of 2 phases:

1. **Gibbs sampling**: first, the input (v) data is sent through the visible layer (forward pass), then the states of the neurons in the hidden layer are computed (h). Based on the obtained hidden layer values, it predicts the new input values for the visible layer (backward pass).
2. **Contrastive Divergence:** update weights based on the inputs to visible layers and the activation probabilities for the hidden layers.

To find the optimal weight W, which minimizes the cost function, J, we use SGD (Stochastic Gradient Descent) algorithm. For this we need to compute the derivative of J with respect to v and h (visible and hidden neuron's input), which in turn gives us 2 terms, a positive and negative gradient. Positive phase increases the probability of the training data, while the negative decreases it.

To adjust the weights we use the "Contrastive Divergence" algorithm, which builds an update matrix that allows to update incrementally the value of W.

W_new = W + learning_rate * CD

The RBM is trained through several forward and backward passes until it can extract the most important features/patterns.

2.2. Building the RBM

When building the RBM it is necessary to specify the nr of units in the visible layer, which is computed as PxPx3, where P is the size of the input patches used to preprocess the input images. The larger value we choose for P, the smoother the reconstruction result will be, but at a higher computational cost. A value of 8 for P has produced satisfying results. For the size of the hidden layer, H we select a small number, 20, because we would like to compress the representation of the input, encoding it using fewer values.

The RBM can be defined as follows:

$$RBM_{model} = (W, b_{hid}, b_{vis})$$

W represents the weights matrix, initialized with a random Gaussian distribution, and b_hid as well as b_vis represent the bias vectors associated to the hidden and the visible layers.

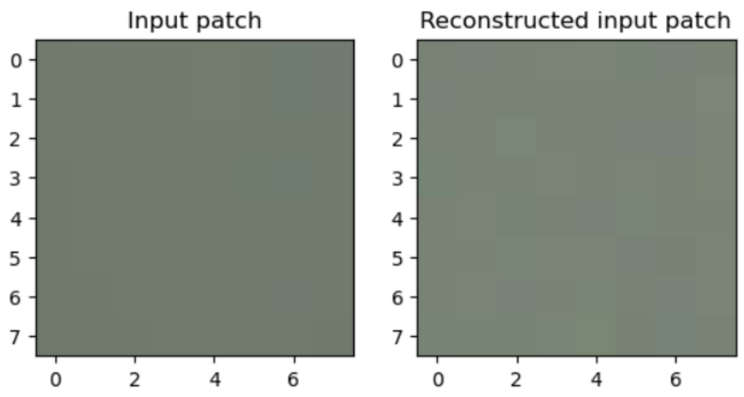
The formula for obtaining the reconstruction vector can be written as:

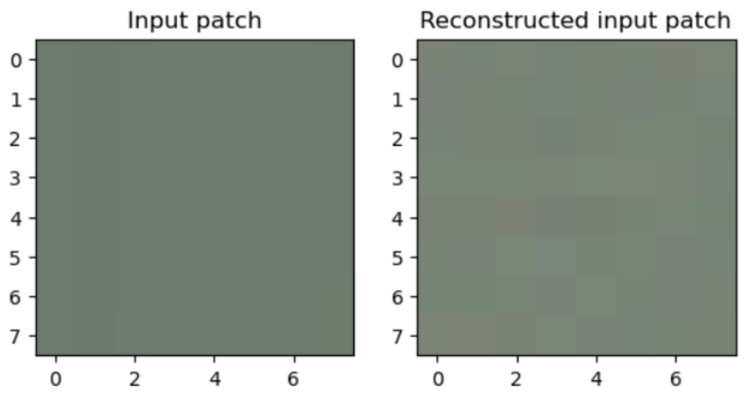
$$x_i' = Sigmoid(x_i \cdot W + b_{hid}) \cdot W^T + b_{vis}$$

The input to the visible layer represent the list of input patches. The output of the visible layer will be the probability of obtaining that input configuration and it is forwarded to the hidden layer which uses this probability to re-obtain the original input configuration.

The RBM is trained using Stochastic Gradient Descent, and the cost function which it tries to minimize is the reconstruction error, calculated as the mean squared error between the input and the reconstructed patch.

$$\underset{W,b_{hid},b_{vis}}{\arg\min} \sum_i (x_i' - x_i)^2$$



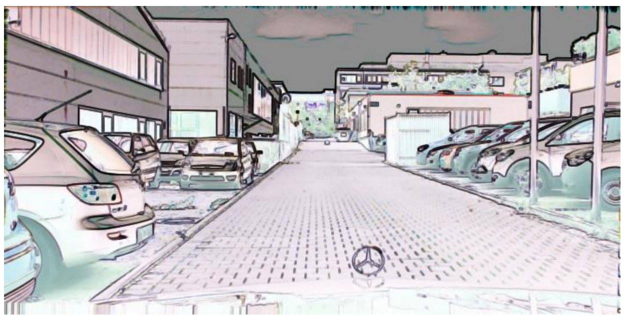


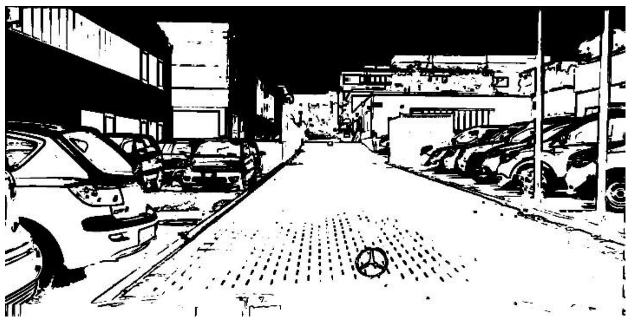
3. Generating the heatmaps

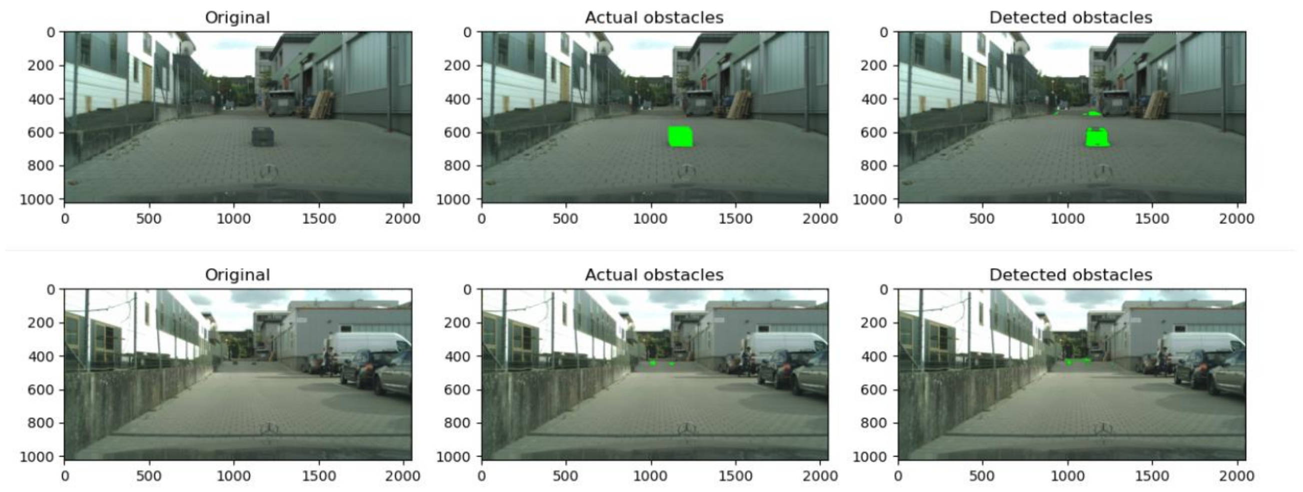
The reconstructed patches will be stitched together to obtain an image with the original input size. Then we calculate the absolute difference between the pixel values.

$$\Delta_{x_i} = |x_i' - x_i|$$

The obtained error map will be inverted, and we apply a binary thresholding on it to eliminate noises and keep only the most relevant pixels which could indicate obstacles.







4. Testing the model

The performance of the model is evaluating using the following metrics:

- **Mean-squared Error (MSE)**: squared difference between the pixels of the original and reconstructed image.
- **Structural Similarity Index (SSIM)**: looks for similarities between the pixel density values checks the image quality degradation, considering parameters such as luminance, contrast and structural information between the 2 images. A SSIM index of 1 represents a perfect match.
- **Pixel-wise Detection Rate (PDR)**: ratio between the correctly detected pixels as obstacles (CDP) and the actual number of pixels belonging to obstacles (AP).

$$PDR = CDP/AP$$

- **Pixel-wise False-Positive Rate (PFPR):** ratio between the number of incorrectly detected pixels as obstacles (IDP) and the actual number of pixels belonging to obstacles (AP).

$$PFPR = IDP/AP$$

5. Results

The model was trained on an nVidia GeForce RTX-3060, 6GB GPU, i7-11800H 2.3GHz machine, in a Cuda environment. Due to the resource limitations and the large processing time, the model was trained and tested on a small subset of the dataset, 32 instances.

Multiple models have been trained with different parameter configurations; the results are recorded below:

Model	MSE	SSIM	PDR	PFPR
A	0.0049	0.8425	0.4880	7.5494
A2	0.0039	0.8741	0.4624	1.5660
B	0.1673	0.0670	0.3916	570.9214
B2	0.1878	0.0352	0.3918	570.0306
C	0.0023	0.9450	0.4916	3.0089
D	0.2979	-0.1965	0.0015	579.5578

Model configurations

Each model has been trained for max 100 epochs, with error limit 0.001 and patience 3, meaning that if the loss hasn't changed after 3 consecutive epochs, then the training is stopped.

P = size of the input patches
H = nr of units in the hidden layer
k = nr of iterations during Gibbs sampling. (k = 1 usually performs well)

Model	Loss function	Train loss	(P, H, k)	Learning rate
A	Mean-squared-error	0.0813	(8, 20, 1)	0.01
A2	Mean-squared-error	0.1965	(16, 20, 1)	0.01
B	Energy loss	0.7064	(8, 20, 1)	0.001
B2	Energy loss	0.4540	(16, 20, 1)	0.001
C	Contrastive divergence	2.4947	(8, 20, 2)	0.01
D	Cross-entropy-loss	0.7227	(8, 20, 1)	0.01

Observations

- Model A had the highest PFPR value, which means it detected the most false positives.
- Model C had the highest SSIM and lowest MSE value which means it could approximate the original input image the best.
- Model D had the smallest MSE and SSIM value which indicates poor performance, similarly to the low true positives rate and the large false positive rate.

Limitations

The anomaly detection system can detect road segments that differ considerably from the learned road texture, based on color information. However, it cannot give information about the dimension or volume of the detected object, therefore it cannot distinguish between a flat object, a newspaper or a brick on the road.

Bibliography

1. Real-time small obstacle detection on highways using compressive RBM road reconstruction, Clement Creusot and Asim Munawar, 2015
2. A Practical Guide to Training Restricted Boltzmann Machines, Geoffrey Hinton, Department of Computer Science, University of Toronto
3. https://wiki.pathmind.com/restricted-boltzmann-machine
4. https://rubikscode.net/2018/10/01/introduction-to-restricted-boltzmann-machines