Evaluation of Machine Learning Models for Received Signal Strength Based Visible Light Positioning with Obstacles

Jorik De Bruycker*
WaveCore, Dramco
KU Leuven
Ghent, Belgium
jorik.debruycker@kuleuven.be
*Corresponding author

Michiel De Wilde
INTEC, IDLab
Ghent University-imec
Ghent, Belgium
tom.dhaene@UGent.be

Federico Garbuglia
INTEC, IDLab
Ghent University-imec
Ghent, Belgium
Federico.Garbuglia@UGent.be

Ioana Nikova
INTEC, IDLab
Ghent University-imec
Ghent, Belgium
ioana.nikova@UGent.be

Ivo Couckuyt
INTEC, IDLab
Ghent University-imec
Ghent, Belgium
ivo.couckuyt@UGent.be

Tom Dhaene
INTEC, IDLab
Ghent University-imec
Ghent, Belgium
tom.dhaene@ugent.be

Nobby Stevens
WaveCore, Dramco
KU Leuven
Ghent, Belgium
nobby.stevens@kuleuven.be

***Abstract*—Received Signal Strength based Visible Light Positioning (RSS-VLP) is an attractive solution for indoor positioning as an enabling technology for future smart environments. Despite this significant importance, there is currently no research focus on the influence of obstacles in the environment. Obstacles could prevent the mobile unit from reaching certain locations, or altogether block the line of sight between the receiver and the light sources. Therefore, this work compares the performance of different machine learning models for visible light positioning using a simulated environment where obstacles can be added. The results shows that machine learning models require much more training data to achieve acceptable performance when shadowing occurs. Gaussian Processes show the best performance out of all evaluated models in environments both with and without obstacles. However, the optimal attainable performance degrades when shadowing is introduced.**

***Index Terms*—Visible Light Positioning (VLP), Received Signal Strength (RSS), Machine Learning (ML), Gaussian Processes (GP), Neural Networks (NN), Support Vector Machines (SVM).**

I. Introduction

Indoor positioning techniques have recently gained a lot of attention, thanks to their interesting application in IoT and ubiquitous connectivity [1], [2]. Since the use of GPS is not accurate enough for indoor positioning, the need for alternative approaches emerges. Many possibilities have been researched, but recently machine learning-based Visible Light Positioning (VLP) techniques have been shown to deliver promising results [3]–[5]. These techniques use the Received Signal Strength (RSS) of LED lights positioned in the environment to allow a receiver to determine its position. One of the advantages of this approach for indoor positioning is that it avoids the cost of dedicated equipment, as the LEDs can just replace the existing light bulbs in any room.

This research received funding from the Flemish Government via the AI Research Program and the Fonds Wetenschappelijk Onderzoek (FWO) program.

While these VLP-based techniques show promising performance, previous studies never investigated the influence of obstacles on the positioning accuracy. Therefore, this work presents a simulated environment with obstacles that can either pose a barrier to the receiver or cast shadows in the environment by interrupting line of sight. Adding these obstacles complicates the task of position estimation based on RSS.

The goal of this work is to gain insight into the influence of adding obstacles to the environment on the performance of various machine learning techniques for visible light positioning. These insights will help further research focus on the most promising models and give an estimation of the optimal performance that can be attained with data-efficient collection techniques that try to minimize the cost of gathering data for training the models.

The contents of this work are structured as follows: section II describes the models that will be compared in more detail. Next, section III describes the details of the simulated environment. Section IV discusses the results of the simulated experiments. Finally, section V summarizes the main findings of this work.

II. Models in comparison

The work compares the following models: Gaussian Processes (GPs), Deep Gaussian Processes (Deep GPs), Neural Networks (NNs), Extreme Gradient Boosting (XGBoost) and Support Vector Machines (SVMs). The inputs of each model are the different RSS values for 4 LEDs, while the outputs are the xy-coordinates. The following subsections provide a more detailed description of each model and elaborate on the implementation aspects.

A. Gaussian Processes

Gaussian Processes are a class of data-driven machine learning models [6]. GPs are a generalization of the Gaussian

probability distribution over variables to a distribution over functions. Where a Gaussian probability distribution is fully specified by a mean vector μ and a covariance matrix Σ, a GP is fully specified by two functions: a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x'})$.

For the experiments described below, the GP implementation of Trieste [7] was used. This implementation uses a mean function that is zero everywhere and the Matérn52 kernel as the covariance function.

B. Deep Gaussian Processes

Deep Gaussian Processes [8] are constructed by combining multiple GPs in different layers to create one big model, similar to combining multiple neurons into a multilayer perceptron.

The experiments for deep GPs described below make use of the Trieste and GPflux libraries [7], [9]. Since Trieste currently does not support deep GPs with multiple outputs and manually building a 2-output deep GP leads to unstable results, the experiments below make use of a combination of 2 deep GPs: One predicts the x-position and the other one predict the y-position. The outputs of each individual model are then combined to form the planar position estimation. By doing this, however, some modelling capacity is lost, as the interaction between the x- and y-position cannot be exploited. The mean function that is used for the deep GP is the zero mean function, and the used kernel is the squared exponential kernel.

C. Neural Networks

Neural networks are one of the most ubiquitous machine learning models. They are composed of multiple neurons, which simply compute an output activation based on the activations of their inputs. The model is built by arranging many of these neurons in different layers, which learn different representations of the input data at different levels of abstraction [10].

The neural network in this work was built using the PyTorch library. The network has two hidden layers, each consisting of 100 units with the ReLu activation function. The output layer uses the sigmoid activation function.

D. XGBoost

XGBoost is an open-source system implementing tree boosting [11]. This library offers a class that implements an XGBoost regressor. In this work, the squared error objective function is used. The scikit-learn library is used to train the hyperparameters. The trained hyperparameters are α (L1 regularization on the weights), λ (L2 regularization on the weights), and γ (minimum loss required to split a leaf).

E. Support Vector Machines

The implementation used to create the SVM model is Epsilon support-vector regression from scikit-learn [12]. This implementation has two tunable hyperparameters: the regularization parameter C and ϵ. Both were optimized using the grid search implemented in scikit-learn.

III. Simulation details

As previously stated, all experiments were executed in a simulated environment. The top views of the simulated rooms are shown in Fig. 1. The accurate radiation patterns emitted by the LEDs were embedded using measurements of real LED lights. To handle shadows in the simulation, a line of sight is calculated between the LEDs and the floor. At each position where a support beam obstructs the line of sight between an LED and the floor, the received intensity of that LED is set to 0. Other effects caused by the obstacles such as non-line of sight contributions and diffraction effects are ignored in the simulation, as they will have a negligible impact on the RSS in real-life experiments. Fig. 2 shows the LED intensities across the floor of the rooms shown in Fig. 1 for one LED, which is indicated with a red dot.

IV. Simulated results

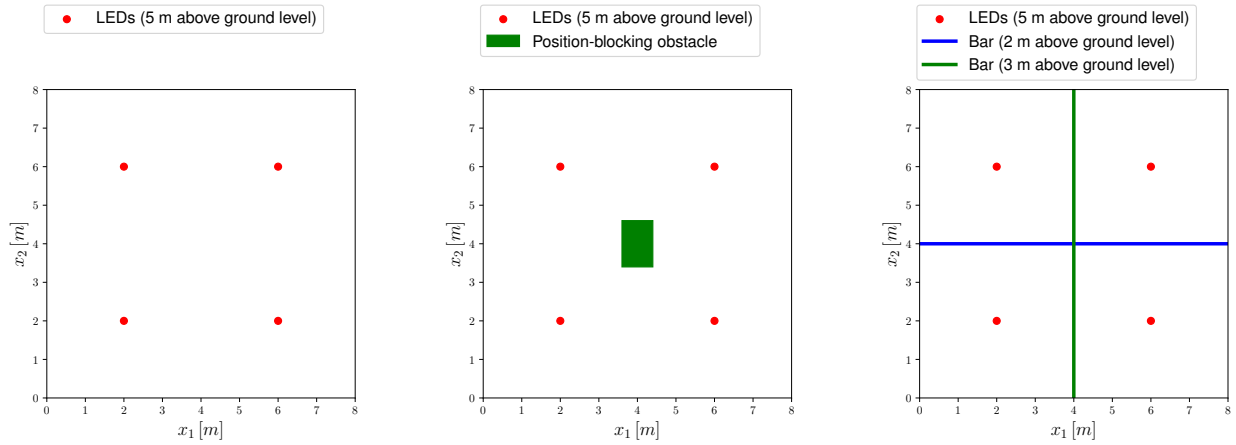
The objective of this work is to examine the impact of introducing obstacles on the performance of various machine learning models. Therefore, the performance of these models is investigated in a room without shadows, as a baseline, and compared to the performance in a room with obstacles present. To be able to make a representative comparison between the different models, all results presented below are averaged over 10 different runs, each using a different set of pseudo-random space-filling data points to train the models. The performance of each model is evaluated by evaluating the prediction error, defined by the planar euclidean distance between the estimated position and the ground truth position. This is executed on a 50×50 grid of test points for a total of 2500 points. To calculate the error at each point in this grid, the RSS is simulated at each point and used as input for the different models. Then, the distance between this point and the point predicted by the different models is computed.

A. Baseline without obstacles

First, the performance of the different models is investigated in a room without obstacles. This room has the layout as shown in Fig. 1a. To have a fair comparison between the different models, their performances can be compared at a point where they have stabilized. This point can be found by increasing the number of training samples until the performance reaches a plateau. A good metric to find this plateau is the P95 error, also known as the 95th percentile error, which indicates below what value 95% of all observed errors fall.

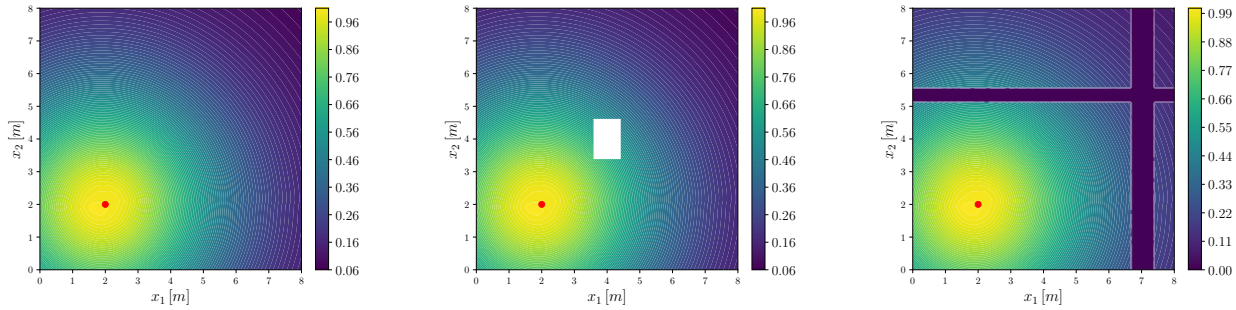
The results are shown in Fig. 3. For all models, excluding XGBoost and the neural network, training using 250 space-filling samples yields a P95 error that is nearly the same as when training uses only 150 space-filling samples. This means that a performance plateau is reached at 150 samples.

Fig. 3 already indicates GPs as the best performing option for position estimation, as it yields the lowest P95 error out of all models. To support this observation, Fig. 4 presents the cumulative distribution functions (CDFs) of the prediction errors of the different models, and Table I shows the exact performance metrics for the different models. The optimal values for each metric are indicated in bold. This information confirms the statement that GPs are indeed the best option for



(a) (b) (c)

Fig. 1: Layout of a square room of 8 by 8 m, with 4 LEDs positioned at a height of 5 m in the following cases: (a) without obstacles, (b) with a position-blocking obstacle (i.e. a pallet) indicated by the green area of 0.8 by 1.2 m that is unreachable by the receiver, (c) with two ceiling support beams (10 cm diameter) at a height of 2 and 3 m, casting a shadow on the floor.



(a) (b) (c)

Fig. 2: Normalized LED intensities across the rooms shown in Fig. 1 for the single LED, indicated by the red dot, in the following cases: (a) without obstacles, (b) with a position-blocking obstacle, no samples can be acquired in the white region, (c) with the ceiling support beams blocking line-of-sight, casting shadows on the floor.

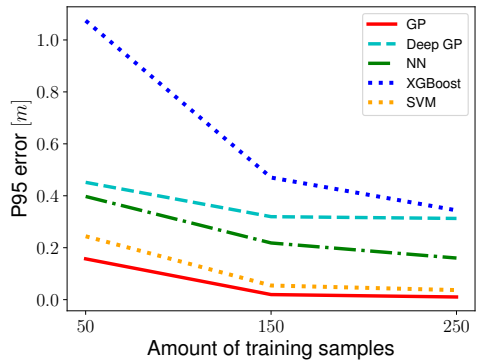


Fig. 3: Evolution of the P95 error for increasing amount of training samples in the room without obstacles.

TABLE I: Performance metrics of the different models trained on 150 samples in the baseline room without obstacles.

Error [m]	GP	Deep GP	NN	XGBoost	SVM
Mean	**0.006045**	0.121078	0.099358	0.219910	0.014333
Median	**0.003247**	0.086946	0.087185	0.180971	0.006806
P50	**0.003245**	0.086918	0.087156	0.180933	0.006802
P95	**0.019532**	0.319513	0.218271	0.470128	0.054439
Max	**0.165373**	0.938490	0.406484	3.591142	0.331765

visible light positioning in the room without obstacles. Not only does the GP have the lowest P95 error, it also has by far the lowest maximum error, mean error, median error and P50 error. The explanation for this result is twofold. First, GPs are data-efficient models, which means that they are designed to work well using only few training samples. Second, GPs are particularly suited to approximate smooth functions. Looking at Fig. 2a, it is clear that the distribution of LED intensities in a room without shadows is very smooth. The position estimation method uses the inverse of this function, which will hence also be smooth. The only model that achieves similar performance to the GP is the SVM. Just like GPs, SVMs assume that the function they are modelling is smooth.

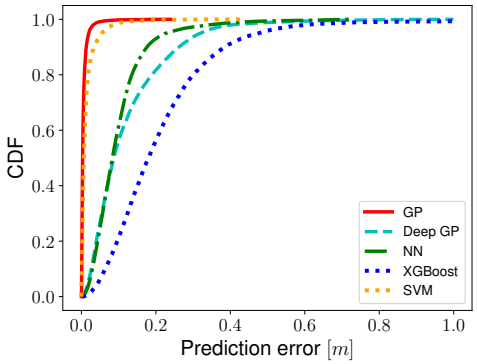


Fig. 4: CDFs of the prediction errors of the different models trained on 150 samples in the baseline room without obstacles.

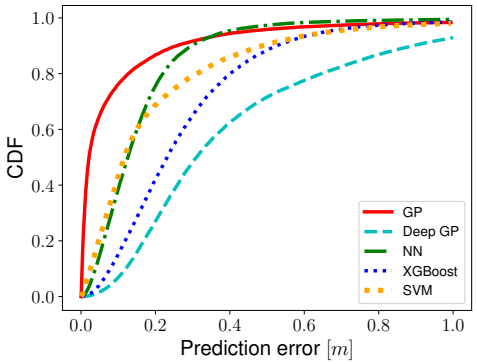


Fig. 5: CDFs of the prediction errors of the different models trained on 150 samples in the room with shadows.

TABLE II: Performance metrics of the different models trained on 150 samples in the room with a position-blocking obstacle.

Error [m]	GP	Deep GP	NN	XGBoost	SVM
Mean	**0.006283**	0.120784	0.093903	0.217222	0.014838
Median	**0.003382**	0.085778	0.084146	0.179969	0.006895
P50	**0.003381**	0.085718	0.084130	0.179914	0.006891
P95	**0.020034**	0.320919	0.197157	0.472902	0.057294
Max	**0.168516**	0.936891	0.401513	3.321792	0.331012

TABLE III: Performance metrics of the different models for 150 training samples in a room with shadows.

Error [m]	GP	Deep GP	NN	XGBoost	SVM
Mean	**0.115095**	0.429381	0.158973	0.283479	0.209209
Median	**0.021629**	0.307685	0.122845	0.231739	0.109007
P50	**0.021616**	0.307531	0.122811	0.231692	0.108953
P95	0.440135	1.103609	**0.387270**	0.652338	0.655778
Max	4.297818	2.957754	**1.596914**	2.826018	2.526133

B. Impact of obstacles

Now that the performance of the different models in the room without obstacles is evaluated, obstacles can be introduced. First, the room with the position-blocking obstacle as shown in Fig. 1b is investigated. This obstacle could be a pallet lying on the ground, hindering the receiver from accessing this area. The performance metrics, averaged over the same 10 runs of samples for the different models trained on 150 samples, are shown in Table II. Note that these 150 samples are spread across the reachable area. When comparing this to the results in Table I, it is clear that the introduction of a position-blocking obstacle barely affects the performance in any way.

Now it is established that position-blocking obstacles do not affect the performance of the studied machine learning models, shadow-casting obstacles are investigated. To get a good view of the influence of adding shadows, the CDFs of the prediction errors of the different models, when trained using 150 samples in the room shown in Fig. 1c are shown in Fig. 5. This figure was created by averaging over the same 10 runs of 150 space-filling samples used to create Fig. 4. When comparing these results to Fig. 4, it is clear that the performance of all models has deteriorated significantly compared to the baseline without shadowing. Table III shows the performance metrics for the different models. These metrics again confirm that all models perform worse when shadows are introduced.

Unlike the first simulations, where there were no shadows, the GP now does not have the lowest P95 error and maximum error. Instead, the neural network now performs best for these two metrics, as it has a P95 error of 38.73 cm and a maximum error of 1.60 m. P95 errors of more than 20 cm are however not acceptable for real-life use cases, so none of the models perform well enough when trained using only 150 samples.

Since the model performance is not yet satisfactory for 150 space-filling samples when there are shadows, collecting more samples will be necessary to achieve acceptable performance. Fig. 6 shows how the P95 error evolves when increasing the number of training samples[1]. It can be observed that the performance of the GP levels out at 550 training samples and that adding more samples does not improve performance anymore. Remarkably, this means that by introducing shadows, the number of training samples needed to reach optimal performance has more than tripled. It is interesting to note that the P95 error slightly increases again when the GP is trained using 650 samples. A possible reason for this is that with this large number of samples, some data points might be too close together. When this occurs, GPs can have difficulties fitting to the data [13].

We now turn our focus to the performance of the different models when trained using 550 space-filling samples, as shown in Fig. 7. Again, it is clear that the GP outperforms the other models. This information can also be deduced from Table IV. The GP performs best for all metrics, except for the maximum error, which is lowest for the Neural Network. The GP is the only model that achieves a P95 error below 20 cms, rendering it the only usable option for accurate visible light positioning in the presence of sharp shadows when using 550 training samples. It should be noted that Fig. 6 indicates that increasing the number of training samples further than 550 will decrease the P95 error for some models. However, doing this is certainly not data-efficient, given that the GP already leads to acceptable performance with 550 training samples.

[1]These results are averaged out over 10 runs. However, for 550 and 650 samples, one of the runs of the GP had to be discarded, as the P95 error was more than 10 times as large as for the other runs. This is likely due to some training samples lying too close to each other, causing the GP failing to fit to the data [13].

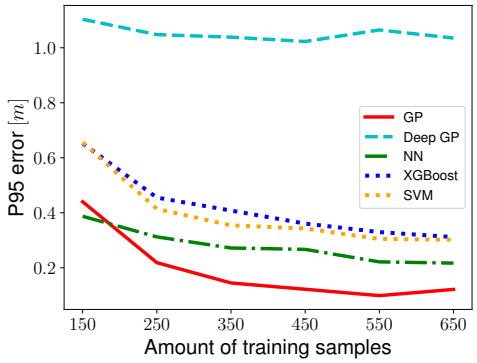


Fig. 6: Evolution of the P95 error for increasing amount of training samples in the room with shadows.

TABLE IV: Performance metrics of the different models for 550 training samples in the room with shadows

Error [m]	GP	Deep GP	NN	XGBoost	SVM
Mean	**0.030026**	0.420668	0.095400	0.142986	0.075077
Median	**0.004229**	0.305035	0.080137	0.114477	0.025778
P50	**0.004226**	0.304872	0.080107	0.114435	0.025763
P95	**0.098736**	1.064793	0.221514	0.329723	0.304864
Max	4.032876	2.427313	**0.613520**	3.932276	0.918414

C. Evaluation of model robustness

The results of previous simulations indicate that position-blocking obstacles do not significantly influence the model performance, but objects interrupting line of sight can substantially complicate accurate position estimation. Therefore, it is interesting to see which model is most robust to the introduction of shadows. To this end, this section compares the performance of each model individually.

1) Gaussian Processes: The previous sections showed that GPs outperform the other models in all scenarios. Fig. 8 compares the performance of the GP in the different cases. The plot shows that the GP did indeed reach a performance plateau at 550 samples in the case there are shadows, as the performance for 650 samples is nearly identical. From the figure, it is also clear that the GP cannot attain the same performance in a room with shadows as in a room without shadows. The reason for this is that GPs are not suited for modelling discontinuities, such as the ones encountered at the intersection between light and shadowy regions.

2) Deep Gaussian Processes: For deep GPs, the performance comparison is depicted in Fig. 9. Naturally, the performance in the presence of shadows is much worse than in the absence of shadows. But more remarkably, increasing the number of training samples does not significantly improve the performance. As already explained in the implementation section about deep GPs, this is likely due to the fact that in this work, the xy-coordinates are modelled independently. The performance of deep GPs should improve with a stable multi-output implementation when available.

3) Neural Networks: Fig. 10 shows that the performance of the neural network is affected by the introduction of shadows, but that the influence is rather limited. Moreover, for increasing amounts of samples, the neural network is able to achieve comparable performance with shadows as without shadows. This was not the case for the GP and deep GP.

4) XGBoost: Comparing the different scenarios for XGBoost in Fig. 11, it is remarkable that this is the only model where the performance exceeds the baseline when trained using more samples. This means that XGBoost is the most robust to the introduction of obstacles. This does not mean that XGBoost is the best model for position estimation, however, since this model performs worse than most previously discussed models in all tested cases.

5) Support Vector Machines: As can be seen in Fig. 12, the results for SVMs are comparable to the results for GPs: a performance plateau is reached at 550 samples when shadows are present. This means that beyond the point of 550 samples, adding more samples will not improve performance. Also, the performance reached when shadows are present is worse than the performance baseline without shadowing. SVMs are less robust to the introduction of shadows than GPs, as they do not achieve acceptable performance when shadows are present.

V. Conclusion

This work compared the performances of various machine learning models for RSS-based visible light positioning. It was demonstrated that the introduction of position-blocking obstacles barely impacted the model performance. However, it is shown that shadowing by the environment significantly affects the models' capabilities in estimating positions accurately. Moreover, when shadowing is present, much more training data is required to achieve adequate position estimation performance. In the absence of shadow-casting obstacles, GPs and SVMs demonstrate superior performance compared to other models. When shadow-casting obstacles are introduced into the room, GPs emerge as the top-performing model.

Further research is required to verify the simulated results in real environments. Moreover, further focus on the performance of GPs in these environments is interesting as they do not only achieve the best performance, but GPs are also the most data-efficient solution. In particular, the behavior of these models at the intersection between light and shadowy regions should be examined. These edges cause discontinuities in the RSS values, which are difficult to model by GPs, and will hence impact the model performance detrimentally. Additionally, the performance of deep GPs should be reassessed once a stable implementation supporting 2-output models is available.

References

[1] P. S. Farahsari, A. Farahzadi, J. Rezazadeh, and A. Bagheri, "A Survey on Indoor Positioning Systems for IoT-Based Applications," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7680–7699, May 2022, conference Name: IEEE Internet of Things Journal.

[2] F. Zafari, A. Gkelias, and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019, conference Name: IEEE Communications Surveys & Tutorials.

[3] W. Raes, N. Knudde, J. De Bruycker, T. Dhaene, and N. Stevens, "Experimental evaluation of machine learning methods for robust received signal strength-based visible light positioning," *Sensors*, vol. 20, no. 21, 2020.

[4] N. Knudde, W. Raes, J. De Bruycker, T. Dhaene, and N. Stevens, "Data-efficient gaussian process regression for accurate visible light positioning," *IEEE Communications Letters*, vol. 24, no. 8, pp. 1705–1709, 2020, publisher: IEEE.

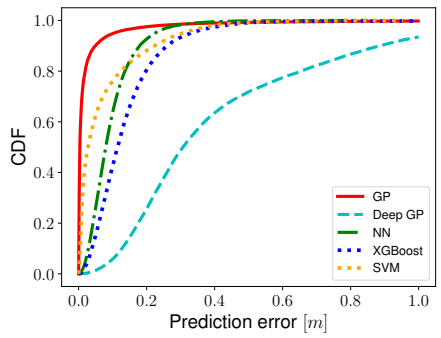


Fig. 7: CDFs of the prediction errors trained on 550 samples with shadowing.

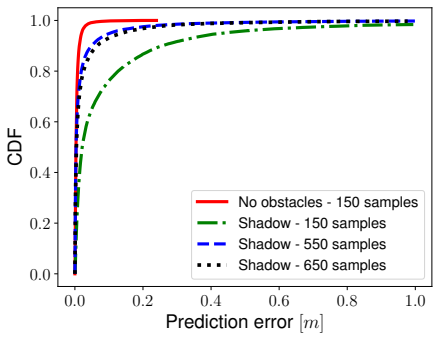


Fig. 8: CDFs for GP in the different cases for varying training set size.

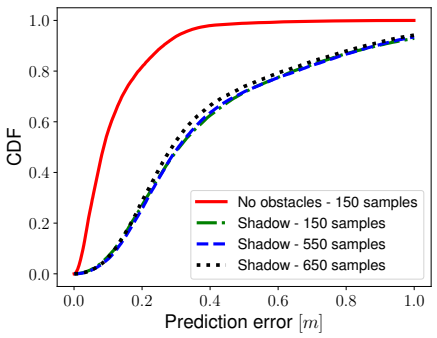


Fig. 9: CDFs for deep GP in the different cases for varying training set size.

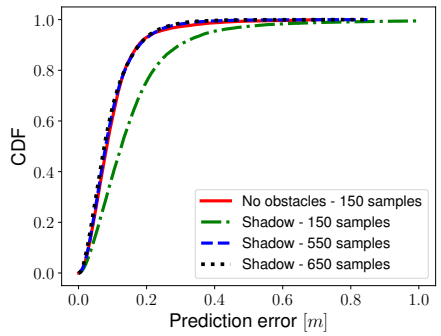


Fig. 10: CDFs for the NN in the different cases for varying training set size.

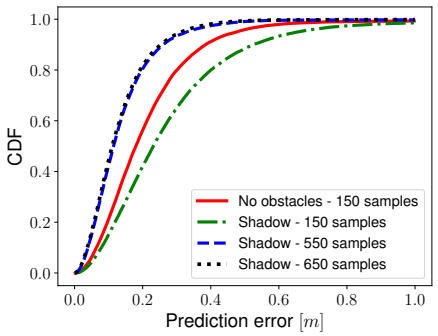


Fig. 11: CDFs of XGBoost in the different cases for varying training set size.

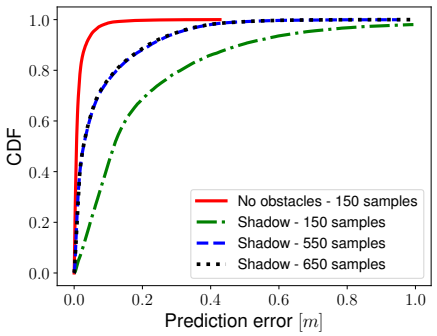


Fig. 12: CDFs of the SVM in the different cases for varying training set size.

[5] F. Garbuglia, W. Raes, J. De Bruycker, N. Stevens, D. Deschrijver, and T. Dhaene, "Bayesian Active Learning for Received Signal Strength-Based Visible Light Positioning," *IEEE Photonics Journal*, vol. 14, no. 6, pp. 1–8, Dec. 2022, conference Name: IEEE Photonics Journal.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006, oCLC: ocm61285753.

[7] V. Picheny, J. Berkeley, H. B. Moss, H. Stojic, U. Granta, S. W. Ober, A. Artemev, K. Ghani, A. Goodall, A. Paleyes, S. Vakili, S. Pascual-Diaz, S. Markou, J. Qing, N. R. B. S. Loka, and I. Couckuyt, "Trieste: Efficiently exploring the depths of black-box functions with tensorflow," 2023. [Online]. Available: https://arxiv.org/abs/2302.08436

[8] A. Damianou and N. D. Lawrence, "Deep Gaussian Processes," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2013, pp. 207–215, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v31/damianou13a.html

[9] V. Dutordoir, H. Salimbeni, E. Hambro, J. McLeod, F. Leibfried, A. Artemev, M. van der Wilk, M. P. Deisenroth, J. Hensman, and S. John, "Gpflux: A library for deep gaussian processes," *arXiv:2104.05674*, 2021. [Online]. Available: https://arxiv.org/abs/2104.05674

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, number: 7553 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14539

[11] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794, arXiv:1603.02754 [cs]. [Online]. Available: http://arxiv.org/abs/1603.02754

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] S. Basak, S. Petit, J. Bect, and E. Vazquez, "Numerical issues in maximum likelihood parameter estimation for Gaussian process interpolation," in *Machine Learning, Optimization, and Data Science*, 2022, vol. 13164, pp. 116–131, arXiv:2101.09747 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2101.09747