Towards Fast Visual Explanations of Local Path Planning with LIME and GAN

Amar Halilović[1], Senka Krivić[2]

[1]Institute of Artificial Intelligence, Ulm University
[2]Faculty of Electrical Engineering, University of Sarajevo

Abstract
As robots become a more significant part of humans' daily lives, bridging the gap between robot actions and human understanding of what robots do and how they make their decisions becomes challenging. We present an approach to local navigation explanation based on Local Interpretable Model-agnostic Explanations (LIME), a popular approach from the Explainable Artificial Intelligence (XAI) community for explaining individual predictions of black-box models. We show how LIME can be applied to a robot's local path planner. Moreover, we show how the General Adversarial Network (GAN) can be trained and used for fast explanation generation. We also analyze the quality and runtime of GAN explanations and present a tool for visualizing these explanations online as the robot navigates.

Keywords
Robotics, Path Planning, Explainable Artificial Intelligence, Explainability, Interpretability

1. Introduction

Robots in social environments raise the requirement for explainability of robot behavior [1]. As the tendency of robots' presence in society grows, this requirement becomes more pronounced. The introduction of the "Right to explanation" [2] in the European Union as a part of the General Data Protection Regulation (GDPR) [3] underlines the human right to explanation in the face of machines making decisions that affect humans. Current decision-making methods in robotics largely lack explainability and thus limit the faster adoption of robots in important tasks. A lack of explainability can also become a safety issue when robots behave unexpectedly, putting humans in highly sensitive environments at risk.

We address explainability in robotics by focusing on explainable robot navigation in social environments: Imagine a robot navigating in a known environment with the possibility of encountering humans and obstacles. Local path planners allow robots to follow a global path plan while dynamically reacting to unexpected occurrences. Some of the robot's decisions may require abrupt stops or changes of direction and path deviations, thus surprising people in the neighborhood or even scaring them. This can lead to trust loss, which needs to be mitigated [4]. One mitigation strategy is explanation. We want to mitigate trust loss by enabling robots to explain their navigational choices. Using Local Interpretable Model-agnostic Explanations (LIME) [5], an established method from Explainable Artificial Intelligence (XAI)

HI-AI@KDD, Human-Interpretable AI Workshop at the KDD 2024, 26th of August 2024, Barcelona, Spain
amar.halilovic@uni-ulm.de (A. Halilović); senka.krivic@etf.unsa.ba (S. Krivić)
0000-0002-2354-986X (A. Halilović); 0000-0001-8045-427X (S. Krivić)

[6], we demonstrate how a robot can generate visual explanations of its local decision-making in path planning and obstacle avoidance. To approach explanation generation in real-time, we train a Generative Adversarial Network (GAN) [7] model on a dataset produced by LIME. We demonstrate how the trained GAN model generates visual explanations of local path plans.

2. Technical background

2.1. Local Interpretable Model-agnostic Explanations (LIME)

LIME [5] is a model-agnostic local XAI technique that explains predictions of a black-box model by learning an interpretable model around the instance of interest. The instance of interest can be anything that is an input to an AI model, be it text, numerical data, or images. We focus on visual explanations and use an image (viz., the local costmap, see below) as the instance of interest. LIME for images[1] takes the input image and partitions it into segments – superpixels, thereby creating interpretable features. Then, it perturbs interpretable features, turning them off to generate perturbed samples (perturbations) in the neighborhood of the instance of interest. For every perturbation, LIME queries the black-box model and thereby generates a local data set of (perturbed) neighbor images and the respective black-box model's predictions. On this new dataset, LIME trains an interpretable model, viz., a weighted linear regression model. The explanation is obtained by interpreting the coefficients of the trained linear model: The importance of each segment in the image for the behavior of the black-box model is represented by one coefficient in the linear model. Depending on the sign of the coefficient, the interpretable feature (viz., the segment in the image) positively or negatively affects the black-box model's prediction. That said, applying LIME to explain local navigation visually, one needs to provide a suitable method for computing a segmentation of a local costmap (viz., the interpretable features). Moreover, the output of the local planner has to be interpreted as the prediction of some black-box model.

2.2. Generative Adversarial Networks (GANs)

GANs were introduced by Ian Goodfellow et al. [7] as a deep learning framework for the estimation of generative models. Estimation is done by an adversarial process where a generative model, Generator (G), and a discriminative model, Discriminator (D), are trained concurrently. G generates new samples by learning the training data distribution, while D estimates whether the provided sample is from the training data or is produced by G. Mehdi and Osindero [8] introduced conditional GAN (cGAN), where G and D are conditioned on some information. Isola and colleagues [9] show how cGAN can be used for image-to-image translation by conditioning on images. G is trained to learn translation between input and output images and fool D, while D learns to classify output images as real (coming from the training dataset) or fake (generated by G). In our work, we employ their *pix2pix* cGAN architecture[2] to achieve a fast explanation generation of local navigation decisions. D is trained by minimizing the negative log-likelihood of identifying real and fake images conditioned on input images, while G is trained using the

[1]https://github.com/marcotcr/lime
[2]shorturl.at/giFUX

adversarial loss of D (whether it fools the discriminator or not) and L1 loss (mean absolute per pixel difference between real and fake images) which are combined into a composite loss function. We condition G and D on local costmaps (see Fig. 1b,1f,1j) as inputs and explanation images (LIME outputs) (see Fig. 1c,1g,1k) as outputs. Both input and output images include (besides obstacle information) the robot's location, the local plan, and the global plan.

3. Experiment I: Explanations with LIME

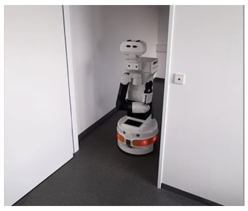
3.1. Technical Set-Up

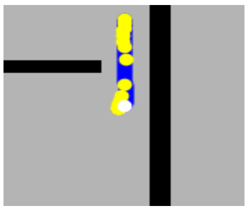
Our set-up is situated in the context of the ROS navigation stack [10]. A global path planner has generated a global path plan for the robot to navigate to a specified goal position. For path following and obstacle avoidance, a local path planner takes the local costmap and the global path as input and outputs a local path (in terms of a velocity vector) for the robot to execute. For LIME to be applicable, the black-box behavior needs to be deterministic. Therefore, we do not employ sampling-based planners, such as DWA or RRT, but instead, employ the TEB planner [11]. To use LIME for generating visual explanations of local path plans in terms of obstacles in the local neighborhood of the robot, we use the local costmap as an instance of interest and the TEB planner as the black box that takes that costmap as input and outputs some local path. LIME first segments the local costmap into obstacles as interpretable features. The SLIC [12] segmentation algorithm is used to get obstacle segments. As the second step, LIME obtains perturbations of the segmented costmap by turning off segments by replacing them with free space. The perturbed local costmap, together with the global plan, the robot's footprint, and its current velocities, form the input to the TEB, which then outputs a local plan for the perturbation at hand. The deviation of the so-calculated local plan from the global plan is taken as a target for the interpretable model and is calculated as a sum of the minimal point-to-point L2 differences between the local and the global plan.

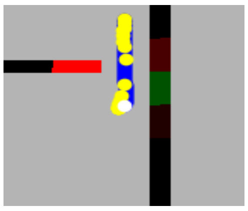
We get an explanation image for each local navigation decision by coloring segments based on their LIME coefficients. The sign of the coefficient dictates the color: Positive-weighted segments are colored green, and negative-weighted segments are colored red. A green-colored segment contributed positively to the deviation; that is, green indicates *"without that segment, the local plan would deviate less from the global plan"*. Conversely, a red-colored segment indicates *"without that segment, the local plan would deviate more from the global plan"*. Color intensity is set proportional to the coefficient with intensities in the range $[0, 255]$ in RGB color space.

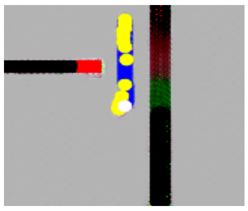
3.2. Qualitative results

Figures 1a, 1e, and 1i show three characteristic local navigation cases (C1, C2, and C3) in our lab. The robot (a TIAGo from PAL robotics) tries to follow the global plan that leads it through the doorway. Figure 1b, 1f, and 1j show the local costmaps for three local navigation cases with black obstacles, white robot's location, and grey free space. In C1, the local plan (yellow dots) mostly coincides with the global plan (blue dots), while in C2, the starting and ending points of the local plan could not be connected into a joint trajectory. In C3, the local plan deviates

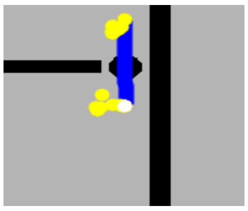

(a) **C1**: robot

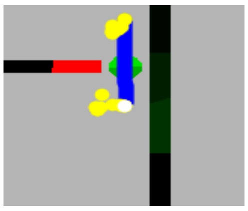

(b) **C1**: costmap

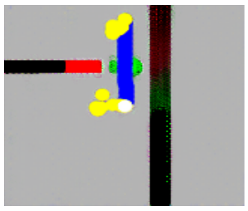

(c) **C1**: LIME expl.


(d) **C1**: GAN expl.

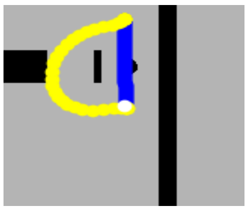
(e) **C2**: robot

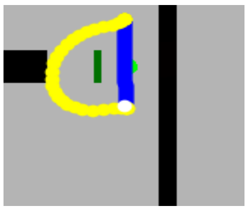

(f) **C2**: costmap


(g) **C2**: LIME expl.


(h) **C2**: GAN expl.

(i) **C3**: robot


(j) **C3**: costmap


(k) **C3**: LIME expl.

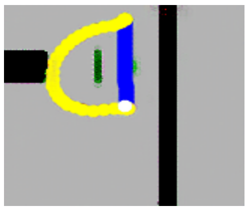

(l) **C3**: GAN expl.

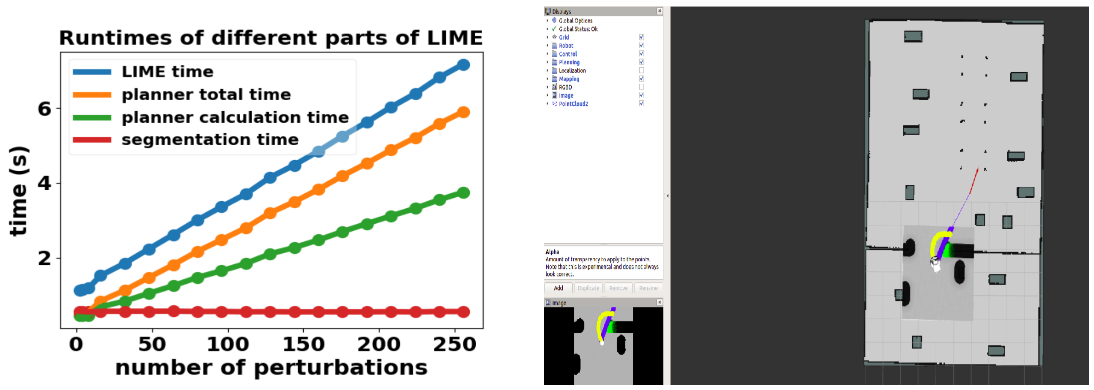
Figure 1: C1: A free doorway allows TIAGo to follow the initial trajectory and move through the doorway. Because TIAGo is too close to the right wall, it has to adjust its position and proceed through the doorway. **C2**: The same doorway is blocked with a chair, so TIAGo cannot progress through the doorway. It stops and rotates in place to try to go left. **C3**: A table, box, and wall form two doorways where the right doorway through which TIAGo should go is suddenly blocked by the trash can. The robot must deviate from the initial trajectory, traversing through the free doorway.

from the global plan. The LIME explanation explains how obstacles and/or parts of obstacles contribute to the deviation. From the explanation images 1c, 1g, and 1k we have:

- C1: "The right (green) wall segment increases deviation, while the left (red) wall segment decreases it, squeezing it to the doorway."
- C2: "The (green) obstacle increases the deviation because if it were not there, the robot would follow the global plan. If the wall (red) were not there, the local path planner could create the connected local plan and deviate from the global plan."
- C3: "Both obstacles increase the deviation, but the round one does so more significantly. If it were not there, the robot would follow the global plan. If the rectangular obstacle were not there, the robot would still deviate, but less."

3.3. Quantitative Results

We analyze *explanation runtime.* LIME's runtime is generally high and increases linearly in the number of perturbations as shown in Fig. 2a, where the runtimes of the most important parts of the LIME are plotted. *Planner total time* takes the biggest part of the total explanation runtime and includes the preparation of input data for the planner (TEB), the planner calculation time of all the paths for each perturbation, and the collection of the planner's outputs. The *planner calculation time* takes the biggest part of the *planner total time.* Both runtimes increase relative to the increase in the number of perturbations. As segmentation only needs to be done once for each explanation, its runtime is unaffected by the number of perturbations.



(a) Runtime (b) Visualization

Figure 2: a) Total runtimes of different parts of the explanation method. b) Visualization of the visual explanations in Rviz in real-time as generated by the GAN: TIAGo is adapting its initial trajectory, deviating from the global plan, which would lead to the collision with the wall.

LIME has clear limitations in that this method alone cannot be used for real-time explanations. Fast-changing and socially complex environments like streets or places with people might require real-time explanations. Even when using a small number of perturbations (which affects the explanation quality), not every TEB call (every 200ms) can be explained in real time.

4. Experiment II: LIME Explanations with GAN

The first experiment showed that LIME can be used to generate meaningful visual explanations but that the generation procedure is too slow for online usage. To approach explanation generation in real time, we utilize GAN as an explanation method. Our main idea is to use LIME only offline to generate a dataset of pairs of local costmaps and respective explanations. With this dataset, we train GAN for image-to-image translation. This way, explanation generation becomes independent of the number of samples.

We have trained an image-to-image GAN for 200 epochs with 240 training image pairs, 60 validation image pairs, and 60 test image pairs. The image-pairs dataset was generated using LIME with the configuration as outlined in the description of our first experiment; see Section 3.1. Of the important GAN settings, *resnet_9blocks* is used as Generator architecture. Other settings are kept as in the *pix2pix* standard implementation. The trained GAN model generates an explanation image by taking a local costmap with a plotted robot's position and local and global plans as input. In the following, we refer to the trained GAN model simply as GAN.

4.1. Results

We assess the quality of the visual explanations generated by the GAN by human visual examination. This is a recommended practice [13, 14]. Figures 1d, 1h, and 1l show GAN explanations for the use-cases C1, C2, and C3, respectively. One can see distortions in the GAN explanations, which, however, do not do any harm to the conveyed meaning. In C1, one can see that the colors of colored segments are not as sharp as in LIME, but the explanation does not suffer qualitatively. GAN's explanation for C2 has similar properties, with even somewhat different coloring of less important segments on the right wall. Still, this does not hamper the explanation very much, as the main contributors are still clearly distinguished. In the GAN explanation for C3, the green color is somewhat duller and blurred compared to LIME, but the contributions of the segments are still visible. We report a mean *GAN calculation runtime* of 0.25 seconds and a mean *GAN model loading runtime* of 0.36 seconds. Hence, once the GAN model is loaded, it can output four explanations per second.

4.2. Demonstrator: Visual Explanations with RViz

We demonstrate how GAN explanations can be visualized in Rviz in real-time in Fig. 2b. The GAN output is published as *PointCloud2* and overlayed over the map view in RViz as a local explanation layer. The GAN model is loaded once at the beginning of navigation and called periodically with every new local plan produced by TEB, allowing for the local explanation layer refresh frequency of 4Hz. This tool thus enables humans to observe which parts of the environment the robot considers important for its navigational decisions. We envision the tool to be used for inspection and debugging, teaching path planning, and demonstrating the robot's internal reasoning processes to interested laymen.

4.3. Discussion

GAN achieves huge runtime savings compared to LIME and approaches the upper real-time performance limit of 200 ms. Most importantly, explanations generated by GAN do not depend on any image segmentation preprocessing, and the performance-hungry process of replanning the local path for every input image perturbation is no longer needed. This translates to the possibility of achieving explanations in near real-time even when many obstacles are considered potential explanations. This allows for explanation generation in highly dynamic environments.

One drawback of the GAN model is some distortions in the visual explanation. However, these are not too harmful as they are local and do not significantly affect the color and shade of color. A limitation of our work is that we have not systematically analyzed how well the GAN explanations generalize to very complex environments. The GAN explanation procedure does not make assumptions about the robot platform and its kino-dynamic constraints. It also does not assume a specific underlying local planner. It generates the explanation only based on an image containing the local obstacles along with the local plan and the global plan. Thus, it may turn out that the GAN has to be retrained for every robotic platform. Another limitation is that our explanation approach relies on the underlying local planner to be deterministic. This is necessary because the procedure must be certain that a variation in the local path is due to the obstacles in the surroundings rather than random fluctuations. In the future, we will also investigate how non-deterministic path planners could be explained.

References

[1] M. Lomas, R. Chevalier, E. V. Cross, R. C. Garrett, J. Hoare, M. Kopack, Explaining robot actions, in: Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, 2012, pp. 187–188.

[2] M. E. Kaminski, The right to explanation, explained, Berkeley Tech. LJ 34 (2019) 189.

[3] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing 10 (2017) 10–5555.

[4] S. Tolmeijer, A. Weiss, M. Hanheide, F. Lindner, T. Powers, C. Dixon, M. Tielman, Taxonomy of trust-relevant failures and mitigation strategies, in: Proceedings of HRI 2020, 2020.

[5] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[6] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence (2019) 1–38.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in neural information processing systems 27 (2014).

[8] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784 (2014).

[9] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[10] R. L. Guimarães, A. S. de Oliveira, J. A. Fabro, T. Becker, V. A. Brenner, Ros navigation: Concepts and tutorial, Robot Operating System (ROS) The Complete Reference (Volume 1) (2016) 121–160.

[11] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, T. Bertram, Trajectory modification considering dynamic constraints of autonomous robots, in: ROBOTIK 2012; 7th German Conference on Robotics, VDE, 2012, pp. 1–6.

[12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, Slic superpixels compared to state-of-the-art superpixel methods, IEEE transactions on pattern analysis and machine intelligence 34 (2012) 2274–2282.

[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, Advances in neural information processing systems 29 (2016).

[14] A. Borji, Pros and cons of gan evaluation measures, Computer Vision and Image Understanding 179 (2019) 41–65.