Towards Metric Monocular SLAM using Spoken Language and Object Detection

Jose Martinez-Carranza[1] Delia Irazú Hernández-Farías[1] L. Oyuki Rojas-Perez[1]
Aldrich A. Cabrera-Ponce[2]
[1] Instituto Nacional de Astrofisica Optica y Electronica (INAOE), Mexico.
[2] Benemerita Universidad Autonoma de Puebla (BUAP), Mexico.
carranza@inaoep

Abstract

We present initial results of a novel approach, merging spoken language and visual object detection to generate depth images in real-time monocular SLAM systems without depth cameras. Our method uses a matrix representation for language and objects, and a partitioning algorithm for association. Whisper processes spoken language, whilst YOLO handles object detection. ORB-SLAM handles camera pose estimation and scene mapping. Users can explore the scene, observe detected objects, and offer depth information via speech. Our system converts descriptions into depth images. Experiments in indoor settings reveal results comparable to RGB-D images, maintaining real-time performance.

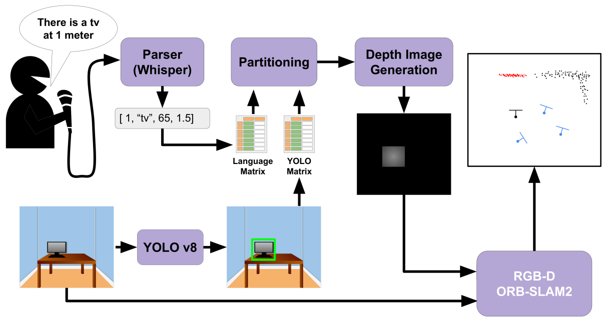


Figure 1. Our approach resolves the association between the objects mentioned in the spoken description and those detected with YOLOv8 , seeking to assign depth values, which is then used to create a depth image for RGB-D ORB-SLAM to build a map and estimate the camera pose in real time.

1. Introduction

Virtual assistants like Siri and Alexa are growing popular for interacting with computational systems. In human-robot interaction, spoken language has been employed for commanding robots to perform tasks [16]. This process entails audio recording and conversion to text, extracting key elements to deduce actions like navigation, manipulation, and carrying for the robot.

Recently, Large Language Models (LLMs) have been employed to create end-to-end models that link language and sensorial data, including visual data obtained from cameras, to robot actions for high-level tasks [6, 9]. However, many of these LLMs require high-end GPUs and large amounts of memory or access to cloud computing over the internet.

This research focuses on depth estimation for monocular SLAM [5], a crucial task in robotics, motivated by the benefits of artificial systems interacting with users. Monocular SLAM allows robots to create maps using environmental visual information while estimating their position, offering advantages like low-cost processing, minimal sensor needs, and energy savings [12]. This is particularly relevant for small robots and wearable smart devices where depth cameras may not be available [14].

In this work, we propose a new approach to generate depth images using spoken descriptions processed online in conjunction with object detection in images captured with a monocular camera, facilitating metric monocular SLAM in real-time, see Figure 1. Therefore, our contributions are:

- A methodology to codify and associate spoken description and visual object detection to assign depth information to detected objects.
- Our initial results show that our approach produces effective depth images, providing sufficient metric information for the SLAM system to build maps and estimate camera poses close to those obtained when using RGB and depth images obtained a depth camera.

2. Related Work

Natural language communication is essential, and incorporating this understanding into systems can improve their performance. Combining vision and language tasks is an emerging research field impacting artificial intelligence systems. Prominent models like ALIGN [10] and CLIP [15] have been developed for tasks like Vision-and-Language Navigation [2, 8], aimed at developing agents that communicate with humans in natural language for 3D environment navigation.

Another less explored task is using natural language for monocular depth estimation, with DepthCLIP [19] being a zero-shot monocular depth estimation method. ObjCAViT [3] leverages auxiliary information from object detection and language models (using YOLOv7 and CLIP).

Despite neural network models' capabilities, limitations like processing static images restrict their real-time application [7]. Our proposal contributes to this area, eliminating keywords or wake words for speech recognition and employing indefinite adjectives for more natural interaction.

3. Methodology

In this section, we describe our proposed approach, where the transcription of the spoken description using Whisper and the object detection using YOLOv8 produce matrix representations that are used by the partitioning algorithm to associate depths with the objects visually detected by YOLOv8. After the association, a depth image is created and published for RGB-D ORB-SLAM to use for feature initialisation.

Our system uses Whisper [4], a state-of-the-art Automatic Speech Recognition (ASR) system to perform transcription and speech translation. The resulting text string is then validated to ensure that the user has provided a valid description of the image.

3.1. Syntactic Parser for Array Representation

Our syntactic parser receives as an entry a validated sentence and produces as output a four-tuple (or a set of them) encoding the information regarding the scene description generated by the user. Each four-tuple $(QUANTIFIER, OBJECT, DISTANCE, INDEX)$ represents the description of a given $OBJECT$ in the scene, how many $(QUANTIFIER)$ objects of the same class are being described, which is the estimated $DISTANCE$ provided, and the corresponding $INDEX$ of the $OBJECT$ in the object detection module.

As we allow the user to make brief pauses in between each object description or to use the word "and" to indicate the connection with the following object description, the pre-processing consists into separate the transcription of the input spoken by the user into phrases to be analysed. Afterwards, we split the whole transcription considering the connection word "and", then we have a set of n sentences depending on how many descriptions were identified.

For each sentence S, we verify it contains a description for a valid object recognized by YOLOv8. We used the labels included in coco128 [1], which is composed of 80 different entries like "laptop", "person", "cup", "cell phone", "stop sign", etc. Sentences that do not contain a valid object are discarded. In each sentence describing a valid object, the user can explicitly mention how many objects there are in the scene (for example *"There are **5** OBJECT at ..."*) or not (for example *"There are **some** OBJECT at ... "*), then it is needed to correctly determine the quantity associated to the recognised object. For doing so, we look for terms referring to a particular quantity like "a couple of", "a", "an", and "half" and replace them with the corresponding numerical value: "2", "1", "1", and "0.5", respectively. Other quantifiers like "a lot of", "some", "few", "many", etc., are treated as particular cases that need to be disambiguated by the Partitioning Algorithm (See Section 3.3), and during parsing are identified by a "0". Once we identify how many valid objects are described by the user, it is needed to extract the information related to the distance estimation made by the user. In the final stage of the parsing module, we generate a four-tuple for each sentence in the transcription. As an example, given the transcribed text: "There is one cup at 1.2 meters, a keyboard at 1.1 meters and one laptop at 1 meter and another cup at 1.2 meters", the resulting vector would be: [[0, 'cup', '1.2', 41], ['1', 'keyboard', '1.1', 66], [1, 'laptop', '1', 63], [1, 'cup', '1.2', 41]].

3.2. Object Detection using YOLOv8

For real-time object detection on a frame-to-frame basis, we used the pre-trained neural network You Only Look Once (YOLO) version 8, which we refer to in this work as YOLOv8 [17, 18]. YOLOv8 was trained on the coco128 dataset [11], which contains 80 objects, and provides the bounding box and label of the object detected in the image.

In our approach, YOLOv8 runs for each frame captured by the camera held by the user. The user views the objects detected by YOLOv8 , with their corresponding labels displayed within the bounding boxes. The user can then describe some or all of the objects in the scene, following the format outlined in the previous section. Each time a matrix representation of a spoken description is generated, the image callback in the YOLOv8 node generates the corresponding matrix representation. This matrix contains a row for each bounding box area, and each column represents each object detected by YOLOv8. A cell is non-zero if the object in the column was linked to the corresponding area in the row. For further referral, we will call these matrices the *language matrix* and the *YOLOv8 matrix*

3.3. Partitioning Algorithm

We rely on the user's spoken description to assist in the association process, thereby guiding the visual SLAM process by producing valuable depth images derived from these

Algorithm 1 partitionGen

```
1:  procedure PARTITIONGEN(dV, hdV, aV, haV, P, n, k, s, level)
2:      if $k = 0$ then
3:          if $s = n + 1$ then
4:              $score \leftarrow 0$
5:              $\mathbf{pdV} \leftarrow \text{zeros}(n)$
6:              for $i \leftarrow 0$ to $|\mathbf{P}| - 1$ do
7:                  $\mathbf{saV} \leftarrow \mathbf{aV}[\mathbf{P}[i] : \mathbf{P}[i+1]]$
8:                  $\mathbf{shaV} \leftarrow \mathbf{haV}[\mathbf{P}[i] : \mathbf{P}[i+1]]$
9:                  $\mathbf{pV} \leftarrow []$
10:                 for $si \leftarrow 0$ to $|\mathbf{saV}| - 1$ do
11:                     for $ni \leftarrow 0$ to $|\mathbf{shaV}[si]| - 1$ do
12:                         $\mathbf{pV}.\text{append}(\mathbf{saV}[si])$
13:                 $mv \leftarrow \text{mean}(\mathbf{pV})$
14:                 $accS \leftarrow 0$
15:                 for $ni \leftarrow 0$ to $|\mathbf{saV}| - 1$ do
16:                     $accS \leftarrow accS + (mv - \mathbf{saV}[ni])^2$
17:                 $score \leftarrow score + accS$
18:                 $\mathbf{pdV}[\mathbf{P}[i] : \mathbf{P}[i+1]] \leftarrow \mathbf{dV}[i]$
19:             if $score <$ bestScore then
20:                 bestScore $\leftarrow score$
21:                 bestDV $\leftarrow$ pdV
22:     else
23:         for $i \leftarrow s$ to $n - k + 2$ do
24:             $\mathbf{P}.\text{append}(i)$
25:             if $\mathbf{hdV}[level] \neq -1000$ then
26:                 $nobjs \leftarrow \text{sum}(\mathbf{haV}[s-1 : i])$
27:                 if $\mathbf{hdV}[level] \neq nobjs$ then
28:                     $\mathbf{P}.\text{pop}()$
29:                     continue
30:             partitionGen($\mathbf{dV}, \mathbf{hdV}, \mathbf{aV}, \mathbf{haV}, \mathbf{P}, \mathbf{n}, \mathbf{k} - 1, \mathbf{i} + 1, \mathbf{level} + 1$)
31:             $\mathbf{P}.\text{pop}()$
```

associations. Therefore, if the scene contains multiple objects of the same category, the user can describe only the depths of the objects whose bounding areas adhere to the assumption that they are inversely proportional to the depth. Thus, under these assumptions, we propose a partitioning algorithm to solve the association between the matrix representations.

To accomplish this, the rows of both matrices are sorted in ascending order for the language matrix and in descending order for the YOLOv8 matrix. Note that both matrices have the same number of columns but may vary in the number of rows, although it is expected that the language matrix contains less or equal rows than the YOLOv8 matrix. For each object in the $i-th$ column of both matrices, a vector $\mathbf{hdV}$ will store non-zero values of the $i-th$ column in the *language matrix*, and $\mathbf{dV}$ will contain their corresponding depths. Similarly, a vector $\mathbf{haV}$ will store non-zero values of the i-th column in the *YOLOv8 matrix* and $\mathbf{aV}$ their corresponding areas. Let the size of these vectors be $k = |\mathbf{hdV}|$ and $n = |\mathbf{haV}|$. To find an association, the vector $\mathbf{haV}$ has to be split into k partitions. For the first partition, let j be a vector index from $[1, \ldots, n-k]$, thus creating two partitions. The second partition can be split again, choosing from $[j, \ldots, n-k-1]$, and so on until the $\mathbf{haV}$ is split into k partitions, see Figure 2.

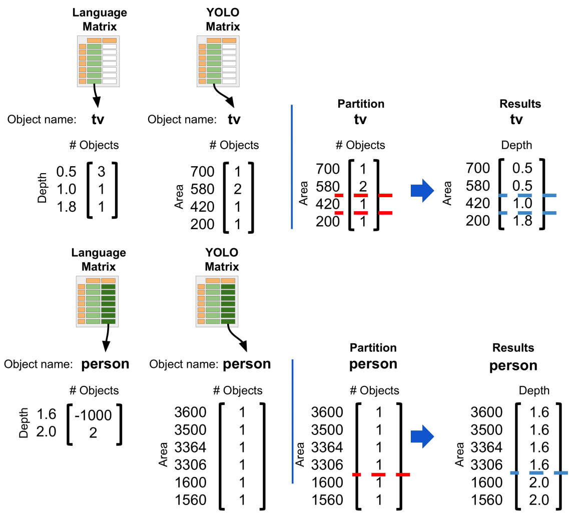


Figure 2. Our partitioning algorithm matches object depths in language and YOLOv8 matrix representations (see Algorithm 1). It assigns depths to areas, considering indefinite adjectives as -1000, as seen in the 'Book' example.

Given a set of k partitions $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k]$, we can sum up the elements of $\mathbf{haV}$ for a given partition. Let the function $sumObj(\mathbf{p}_i)$ be the function that sums such elements for a partition $\mathbf{p}_i$:

$$sumObj(\mathbf{p}_i) = \sum_{j=1}^{|\mathbf{p}_i|} \mathbf{haV}[\mathbf{p}_i[j]] \tag{1}$$

Then, for $i = 1, \ldots, k$, $\mathbf{hdV}[i]$ must be equal to $sumObj(\mathbf{p}_i)$; this means that the number of objects must be the same, otherwise, the partition is invalid, except when $\mathbf{hdV}[i]$ contains a -1000 value; if that is the case then the comparison is overruled and the next partition is calculated. For a valid partition, and assuming that the area is inversely proportional to the depth, we assume that objects with similar depths must have similar areas. Therefore, the difference between each area in the partition and the mean area of the partition must be minimized. The score is defined as follows:

$$score(\mathbf{P}) = \sum_{i=1}^{k} \sum_{j=1}^{|\mathbf{p}_i|} (\overline{a} - \mathbf{aV}[\mathbf{p}_i[j]])^2 \tag{2}$$

where $\overline{a}$ is the mean area of the partition $\mathbf{p}_i$.

This partitioning method was implemented as a recursive function that expands a tree of possible partitions, see Algorithm 1. Note that pruning is possible if the number of objects of partition i does not coincide with the number of objects in $\mathbf{hdV}[\mathbf{i}]$; hence, the next partition will not be attempted. For each valid partition, a vector *pdV* will store the depths associated to each area according to the partition $\mathbf{P}$. The best partition will be the one with the minimum score.

4. Experimental Framework

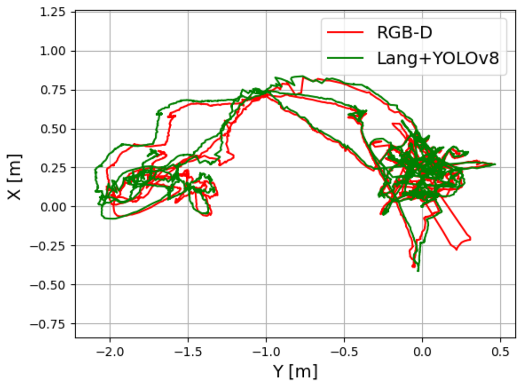
Our experimental framework utilised an Alienware R5 laptop, featuring a Core i7 processor, 32GB RAM, and NVIDIA GTX 1070 graphics card. We used the ASUS PrimeSense Xtion Pro depth sensor, providing RGB and depth images at 640×480 resolution and 30 Hz. The system operated on Ubuntu 20.04.5 LTS with ROS Noetic, running modules such as the spoken speech transcriber, YOLO, partitioning algorithm, and monocular SLAM system. We implemented both Whisper and YOLO in ROS using Python and PyTorch 1.9.0, configuring the visual SLAM system to request 2000 features per frame.

We used two scenarios to evaluate the effect of the quality of depth provided in the spoken descriptions. The first scenario, referred to as *office*, involved measuring objects with a metric tape, following a perpendicular line from the object to the camera plane. In the second scenario, the user walked in a non-straight corridor reaching out an office and then walking backwards to the origin; this scenario is referred as *corridor*. In this scenario, the depth information, provided by the RGB-D camera, of objects detected with YOLOv8 was visible to the user. The depth value corresponds to the pixel value of the centroid of the bounding box, used as an approximation only.

We compared the results of our approach with RGB-D SLAM. Figure 3 displays top-down views of the camera trajectories obtained with RGB-D SLAM against our method. Table 1 summarises the performance of our approach compared to RGB-D ORB-SLAM using RGB-D data. The mean error shows that our method is comparable to when using RGB-D images. Note that the trajectory length was calculated based on the 3D trajectory obtained using RGB-D images with RGB-D ORB-SLAM ; the percentage error was 0.3% and 0.9%. This indicates that the depth information provided by the user was sufficient to generate a high-quality map, which was very close to the map generated using the Xtion depth camera, RGB-D ORB-SLAM operated at a frequency of $30Hz$.

5. Conclusions

We presented initial results of an approach using spoken language and object detection from YOLO to produce depth images for real-time monocular SLAM systems. Our method allows users to assist in mapping where metric is unavailable by providing simple depth descriptions. We proposed a matrix representation and partitioning algorithm for associating spoken descriptions with detected objects, while maintaining camera pose estimation and mapping at 30 Hz. This could contribute to low-budget robotic systems like miniature drones [13] where RGB-D or stereo cameras aren't available. Future work includes enhancing the association algorithm and exploring more visual and language-based cues.



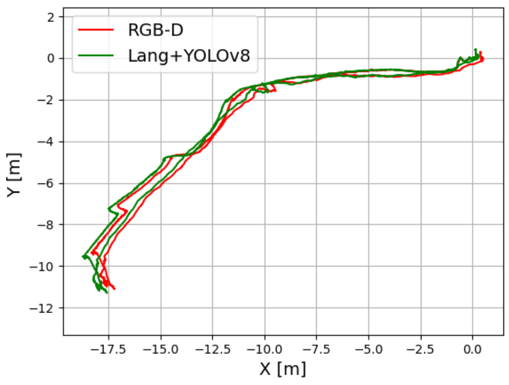


Figure 3. Top views of the map and camera trajectory performed in two video sequences: First row, the office sequence. Second row, the corridor sequences.

Table 1. Trajectory errors are compared w.r.t. to the trajectory obtained with RGB-D ORB-SLAM using RGB-D images from the Xtion camera.

Sequence	**# Images**	**Trajectory**		
		Length [m]	**Mean Error [m]**	**Error %**
Office	12856	55.03	0.139	0.252
Corridor	8296	89.72	0.734	0.853

References

[1] 2086341682@qq.com. coco128 dataset. https://universe.roboflow.com/2086341682-qq-com/coco128-xilzt, dec 2021. visited on 2023-03-15. 2

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CoRR*, abs/1711.07280, 2017. 2

[3] Dylan Auty and Krystian Mikolajczyk. ObjCAViT: Improving Monocular Depth Estimation Using Natural Language Models And Image-Object Cross-Attention, 2022. 2

[4] William Chan, Daniel Park, Chris Lee, Yu Zhang, Quoc Le, and Mohammad Norouzi. Speechstew: Simply mix all available speech recognition data to train one large neural network. *arXiv preprint arXiv:2104.02133*, 2021. 2

[5] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 1

[6] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model. In *arXiv preprint arXiv:2303.03378*, 2023. 1

[7] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration. *ArXiv*, abs/2203.10421, 2022. 2

[8] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, Dublin, Ireland, May 2022. Association for Computational Linguistics. 2

[9] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual Language Maps for Robot Navigation. *arXiv preprint arXiv:2210.05714*, 2022. 1

[10] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *CoRR*, abs/2102.05918, 2021. 2

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in COntext. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 2

[12] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2016. 1

[13] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and Guido CHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019. 4

[14] Sebeom Park, Shokhrukh Bokijonov, and Yosoon Choi. Review of microsoft hololens applications over the past five years. *Applied sciences*, 11(16):7259, 2021. 1

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 2

[16] Caleb Rascon and Ivan Meza. Localization of sound sources in robotics: A review. *Robotics and Autonomous Systems*, 96:184–210, 2017. 1

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[18] Ultralytics. YOLOv8 object detection. https://ultralytics.com/yolov8, 2021. Accessed: March 14, 2023. 2

[19] Renrui Zhang, Ziyao Zeng, Ziyu Guo, and Yafeng Li. Can Language Understand Depth? In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 6868–6874, New York, NY, USA, 2022. Association for Computing Machinery. 2