Localizing Google SketchUp Models in Outdoor 3D Scans

Flavia Grosan, Alexandu Tandrau, and Andreas Nüchter

Jacobs University Bremen gGmbH
School of Engineering and Science, Automation Group
Campus Ring 12, 28759 Bremen, Germany
{f.grosan|a.tandrau|a.nuechter}@jacobs-university.de

Abstract. This work introduces a novel solution for localizing objects based on search strings and freely available Google SketchUp models. To this end we download and preprocess a collection of 3D models to obtain equivalent point clouds. The outdoor scan is segmented into individual objects, which are sequentially matched with the models by a variant of iterative closest points algorithm using seven degrees of freedom and resulting in a highly precise pose estimation of the object. An error function evaluates the similarity level. The approach is verified using various segmented cars and their corresponding 3D models.

1 Introduction

One fundamental topic in mobile robotics is simultaneous mapping and localization (SLAM). In SLAM, the robot builds a map without prior information about its positions. Laser range scanners are helpful in solving SLAM since they provide a series of scans taken from different poses. To obtain a complete, consistent and precise map of the exploration site, scans are often matched with iterative closest points algorithm [9]. This algorithm is commonly used in real time and it iteratively alters the rigid-body transformations needed to minimize the distance between consecutive scans. Finally a GraphSLAM relaxation algorithm is used to refine the map and to yield high precise pose estimates [14].

Once the world is mapped, the next step towards a knowledge based robotic system is obtaining semantic information about the environment and thus enabling the robot to understand and interact efficiently with the exploration site. A semantic map adds information to the obtained scene by identifying static objects or dynamic events. Through localization the robot determines where a particular object is found. Our approach towards semantic mapping takes advantage of the large amount of information publicly available on the Internet. Google Warehouse, a vast collection of user-made 3D models, offers valuable content. For instance, given a precise goal ("There is a BMW Z3 in the environment. Go find it!"), the robot downloads the 3D models available for this object, converts them into 3D cloud points and the localization process is then simplified to common scan matching with an additional attention to scale (Fig. 1).

Fig. 1: Left: 3D model BMW Z3 found in Google SketchUp Warehouse. Right: Actual car in a 3D laser scan.

2   Related Work

3D models are becoming the standard representation in applications ranging from the medical sector to architecture. They are used for visualization and object design because both humans and machines interact with them naturally [2, 11]. Tangelder et al. discuss ways in which models are queried based on text and shape [13]. The similarity between objects is achieved using the nearest neighbor algorithm and improves the scan matching process. The scans are matched using iterative closest points algorithm (ICP). Given an initial relative distance between two scans, ICP matches points and computes an estimate of the transformation between the two states. The process is repeated until convergence. Horn proposed a closed form solution of absolute orientation using unit quaternions, which supports all the basic geometric transformations (translation, rotation and scale) [3].

Nüchter et al. [10] make use of SLAM and ICP to achieve semantic mapping. After determining the coarse scene features (e.g, walls, floors), a trained classifier identifies more delicate objects. Our work improves the semantic map by creating an even finer differentiation between objects in the environment.

Object localization is another application of 3D models. The robot is given a precise target to detect in an unknown environment. This topic is thoroughly studied in the 2D field. Li-Jia Li et al. propose a hierarchical generative model to classify, label and thus recognize objects in a 2D scene using both a visual and a textual model [7]. Similar approaches are implemented during the Semantic Robot Vision Challenge, where Meger et al. designed a robot capable of detecting objects in images collected directly from its camera [8]. In the 3D field, Kestler et al. use a probabilistic representation. The targets are modeled in a knowledge base (descriptive identifiers and attributes) and spatial inference. Feature detection and hierarchical neural net classification are used for object identification, while robot localization is solved through sensor fusion, spatial abstraction and multi-level spatial representations [5]. The drawback of this approach is the need to maintain internal, neural net trained data. Recently, Lai and Fox treat the aforementioned problem and propose the use of Google Warehouse for training classifiers in order to improve and extend object detection [6]. The classifier is based on a distance function for image features, which are obtained via spin images implemented by Johnson and Hebert [4]. The objects in the outdoor

environment are classified in seven categories: cars, people, trees, street signs, fences, buildings and background. The paper does not give a solution for identifying very precise targets, e.g., a BMW Z3. We solve this problem, thus taking an extra step within the field of object recognition and localization.

Hertzberg et al. use CAD models to introduce semantics in maps obtained with SLAM [1]. They created an ontology modeling relationships between objects in an indoor environment scene and use standard ICP to match objects with their corresponding CAD models. The main disadvantage of this approach is the need to store size information in the ontology about the scan components, problem mitigated in this work through our implementation of ICP with scale.

3   Obtaining Point Clouds from 3D Models

Google 3D Warehouse is a collection of user made SketchUp models. Thousands of models are freely available and searchable by strings or tags. A SketchUp model is composed of different entities, i.e., Face, ComponentInstance, or Group. A Face is a triangle defined by three 3D points. ComponentInstances and Groups are entities (with an associated transformation) recursively composed of other entities. Any model is decomposed into a list of faces defined by their respective points. The obtained point clouds are not uniformly sampled, thus requiring an additional sampling procedure. Our sampling procedure adds random points inside each triangular face proportionally to the area of the triangle. We center the point clouds in their respective centroid and bound the coordinates in $[-\alpha, \alpha]$ to avoid initial transformations which hinder the matching algorithm in the following steps. Fig. 2 shows an Audi A4 model before and after the sampling procedure.
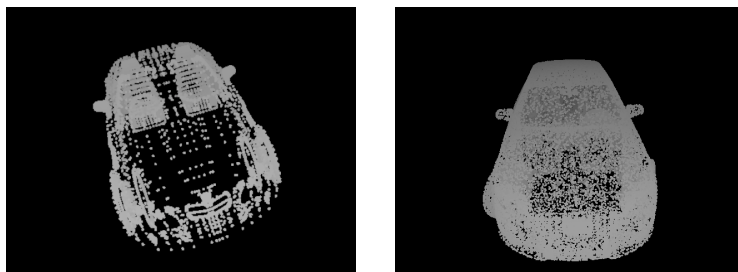

Fig. 2: Audi A4 SketchUp Model before (left) and after (right) sampling.

4   Matching Scanned Objects with Generated 3D Point Clouds

To find the desired object in the outdoor scene, the 3D laser scan is segmented. We exploit the assumptions that all objects are on the ground, i.e., a car al-

ways has the wheels on the ground and so do the downloaded models. Therefore the first step is to remove the ground. For this purpose, we use the approach described by Stiene et al. in [12]. The points are converted into cylindrical coordinates and the nearest neighbor within the vertical sweep plane is identified. The angular gradient defines the nature of the point (ground or not ground). Fig. 3 shows the obtained result. The components of the scan become distinguishable and spatially disconnected.

Fig. 3: Scan with the ground points removed

After the ground has been removed, we automatically select an object with an interactively provided starting guess. First, we find the point p which is closest to the starting guess. Then, a region is grown around p by iteratively adding points p' satisfying the conditions $p_x - \varepsilon \leq p'_x \leq p_x + \varepsilon$, $p_y - \varepsilon \leq p'_y \leq p_y + \varepsilon$, $p_z - \varepsilon \leq p'_z \leq p_z + \varepsilon$. To quickly find the points which match the criteria a k-d tree is used. The ε threshold is dependent on the point density of the scans and the distance from the scan origin to the starting guess. Fig. 4 (left) shows an obtained segmented object.

A complete automatic system repeatedly applies the above scheme using random points as starting guesses. Region growing is applied to automatically segment the environment scan. The procedure is repeated with the remaining points, thus obtaining non-overlapping components. Regions with at least 500 points are considered valid object candidates. They are centered in their respective centroid and scaled in the α-bounding box as in the interactive scheme.

Afterwards, the obtained segmented objects and the Google Warehouse models are matched using a variant of iterative closest points algorithm (SICP). ICP determines the translation and rotation between the two point clouds (object and model), but it does not include the scale factor. Following the derivation for the scale factor by Horn [3], we compute for n matching points of the two point clouds (*model* and *scan*) $\{p_{m,i}\}$ and $\{p_{s,i}\}$ (both centered in their respective
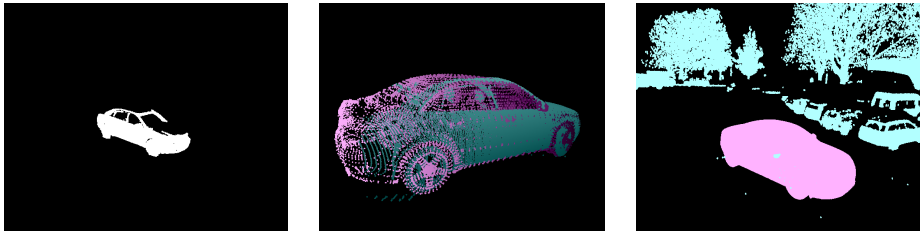
Fig. 4: Left: Segmented Audi A4. Middle: Audi A4 scan and model matched using SICP (magenta model and cyan scan data). Right: Recovered transformation matrix and application to positioning the object in the scan.

centroid):

$$s = \sqrt{\sum_{i=1}^{n} ||p_{m,i}||^2 \Big/ \sum_{i=1}^{n} ||p_{s,i}||^2}.$$

to determine the scale factor. However, it requires symmetric point matching in ICP: points from both left and right are matched with their closest counterpart. Initially, the maximum matching distance between points, Δ, is large to enforce coarse alignment. This scales the model roughly to the same size as the segmented object. We run i iterations with this distance or until the error converges. The matching is further refined by running the same number of iterations with a smaller maximum distance, δ. This step allows fine matchings between points and the objects are consequently perfectly aligned. Fig. 4 (middle) shows a result of the matching between scan and model using SICP. The right subfigure shows the model aligned with its equivalent object in the initial scan.

5 Evaluation Metrics

Running the matching algorithm and applying the recovery transformations results in two sets of 3D points which share the same coordinate system: S – the scan, and M – the model. It is important to note that the scanned set depends on the scanner view point and has occluded parts. We designed an error function which penalizes points in the scan without model correspondence.

Let $c(p) \in M$ be the point in the model which is closest to p by Euclidean distance. Then, the error function is defined as:

$$E = \frac{\sum_{i=1}^{|M|} dist(M_i, c(M_i))}{|M|}$$

A small E denotes a very good match, while larger E values suggest either a poor model or a different object. In the experiments we discovered that the error function also ranks models by similarity to the original scanned object.

6 Experiment and Results

We acquired several scans using a Riegl VZ-400 3D laser scanner in the Jacobs University Bremen parking lot, without focusing on any particular car. Each scan has roughly 4 million 3D points and 5 cars were segmented. For each segmented object we automatically downloaded its corresponding models from Google Warehouse and ran SICP with 4 starting rotations as $(\theta_x, \theta_y, \theta_z)$, where θ_y specifies the rotation around the vertical axis. We only consider the rotation around the y axis and used four initial angles, i.e., $0, \pi/2, \pi$ and $3\pi/2$.
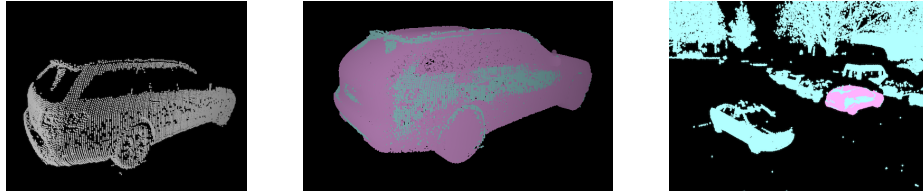
Fig. 5: Segmented Mercedes C350 and its best SICP match.

The first segmented car, a Mercedes C350 (see Fig. 5), contains 8920 points and we downloaded all 89 models available in Google Warehouse. This car presented a large number of very accurate models and the results are observed in tables 1 and 2. In each column, we present the error value on the left, the starting rotation used to obtain this score and the image of the model provided by Google Warehouse. The 89 models downloaded contain not only cars, but also random components as chosen by their creators. Our error function ranks the accurate car models first. As the models resemble less a Mercedes C350, they have a lower rank and the very last are other random objects, which do not resemble a car at all. The Mercedes C350 is a good balance between the quality of the segmented object and reliable collection of 3D models.

Table 1: Mercedes C350 - Best matched models (3)

Error	Y-axis Rotation	Google SketchUp model
49.0	$\pi/2$	

Continued on next page

Table 1 – continued from previous page

Error	Y-axis Rotation	Google SketchUp model
49.0	$\pi/2$	
49.0	$\pi/2$	

Table 2: Mercedes C350 - Models with high error (2)

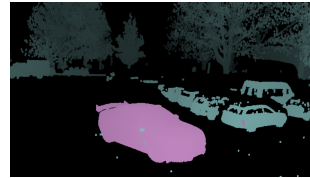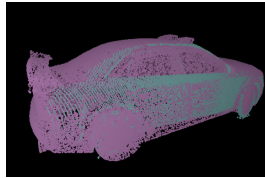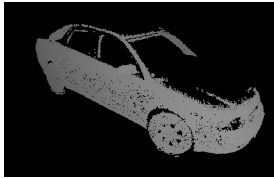| Error	Y-axis Rotation	Google SketchUp model
19865.0	$3\pi/2$	
21221.0	0	

Fig. 6: Segmented Audi A4 and its best SICP match.

We extracted an Audi A4 from the environment scan containing 18801 points and we found 80 models responding to the equivalent search query. Unlike the

Mercedes, the models vary more and have additional decorations. The error function does not take into consideration the model's unmatched points, because in every case the model contains more points than the segmented object. In contrast, the segmented object has points on maximum 3 sides. To solve this problem, a shape descriptor could be used to differentiate between models with extra elements. Being a sports car, Audi A4 is prone to user design experiments, thus making the model collection less reliable in some respects (Fig. 6).
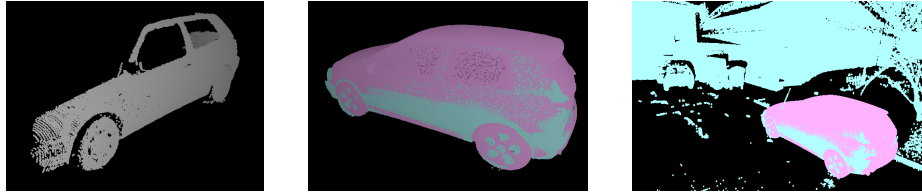

Fig. 7: Segmented Volkswagen Golf and its best SICP match.

The extracted Volkswagen Golf is an unidentified older version of the popular German car and it has 44686 points and 233 downloaded models. The difficulties in matching this model stem out exactly from the fact that it is an old version and the models in Google Warehouse focus mostly on the latest Golf models. However, SICP identified and ranked higher the models resembling an older version as opposed to newer versions of the same car (Fig. 7).
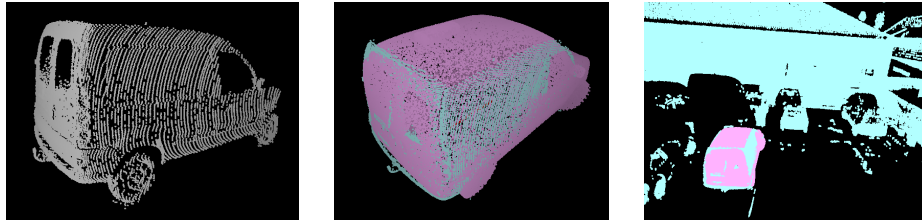

Fig. 8: Segmented Renault Kangoo and its best SICP match.

Another class of cars is represented by a segmented Renault Kangoo, containing 13597 points and with 18 models in Google Warehouse. Even though the number of available models is reduced, they have a very good quality and we found a very good match. This suggests that the proposed algorithm is not only fit for small cars or limousines, but also for larger objects such as vans (Fig. 8).

The last segmented vehicle is a Citroen C5 with 10089 points and only 11 available models in Google Warehouse. Unlike the previous car, the quality of these models is very low and all SICP matchings have a high error rate as it can be observed in Fig. 9. This means that for the less popular type of cars Google
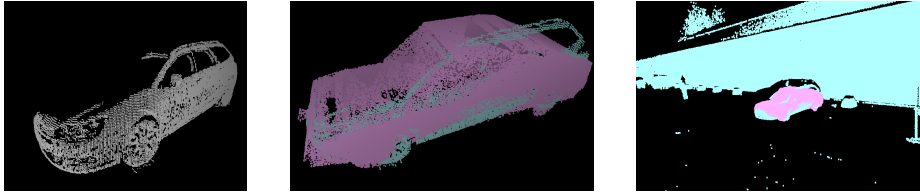
Fig. 9: Segmented Citroen C5 and its best SICP match.

Warehouse is useful in identifying the class of the object, but it is probably not enough in determining a finer classification and finding out the brand of the car.

We considered the segmented Mercedes C350 versus the top 3 models from all the other cars. SICP ranked the models starting with the Mercedes C350 as being the best, followed by two Audis, then, with larger errors, Citroen and lastly Golf and Renault. SICP found the right brand of car among different models and moreover, the next matched models were similar in shape and size with the segmented car, which gives us an insight in similar shapes and objects that share particular properties as observed in table 3.

Table 3: Mercedes C350 vs different car brand models (10)

Error	Y-axis Rotation	Google SketchUp model
49.0	$\pi/2$	
49.0	$\pi/2$	
57.0	$\pi/2$	
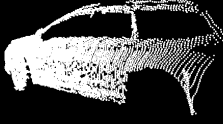
Continued on next page

Table 3 – continued from previous page

Error	Y-axis Rotation	Google SketchUp model
59.0	0	
60.0	0	

We matched an Audi A4 with all objects automatically segmented by using the best matched Audi A4 as the model. By applying the growing regions algorithm we identified 67 candidate objects and the best matches are shown in Table 4.

Table 4: Find Audi A4 in entire scan

Error	Y-axis Rotation	Object
105.0	$\pi/2$	
178.0	$\pi/2$	
400.0	$\pi/2$	

The best matching object is the Audi A4. The next two best-matching objects are also cars, but different brands. The rest of the objects have error values of 2000 to 77000 and are thus very poor matches.

For the purpose of this evaluation, we have used the following constants and thresholds: $\alpha = 500$, $\Delta = 500$, $\delta = 25$, $\varepsilon = 5$. Matching animations

are available at `http://www.youtube.com/watch?v=uVz8yzUn4FA` and `http://www.youtube.com/watch?v=8VRvmgmQV4g`

7   Conclusion and Future Work

In conclusion, the current work presents a feasible approach, proven to work in practice by thorough experiments and results, which solves the problem of identifying a given object in an outdoor environment. Semantic mapping is thus improved and the method is applicable for a better understanding of the scene. The work raises a myriad of new questions and future work sheds some light on short term and long term improvements for SICP.

The error function presented does not take in account extra points which are present on the model but not on the scan. A further difficulty in the attempt to improve the error function is that the glass areas of cars are not captured by a laser scanner. In future work, we plan to project all points to a plane and compare the resulting 2-dimensional convex hull shapes by shared surface.

Our current work focuses on outdoor scans in urban environments and does not cover any experiments with indoor data. One very important assumption for our rotation model is that the objects are on the ground, thus favoring rotations on the y-axis. While the major indoor objects (chairs, tables etc.) are similar in nature to the outdoor objects, refining to smaller, harder segmentable objects with unknown rotations (objects on a shelf) is an open problem. Solving this issue is important in the context of completely identifying real-world objects with Google SketchUp models.

The candidate models are currently found based on the search results of Google Warehouse database. However, it is desired to identify a scanned object with a SketchUp model without a search string. For this purpose, further research work is needed in order to devise an accurate classifier. We currently have no methods to reject objects which share no similarities to the real-world scanned object without running the SICP algorithm.

The presented method is extendable to a full-scene understanding algorithm – both labeling and completion of occluded areas. The SICP algorithm is run on each real-world object of a segmented scan thus finding the best matching SketchUp model and its orientation. The real-world objects are then replaced with the SketchUp point cloud thus resulting both a higher-resolution scan and semantic labeling. Further work will be conducted to allow SICP to adjust the guess based on future scans (backward corrections).

References

1. Sven Albrecht, Thomas Wiemann, , Martin Guenther, and Joachim Hertzberg. Generating 3D Semantic Environment Models using Ontologies and CAD Model Matching. In *Proceedings of ICRA 2011 workshop: Semantic Perception, Mapping and Exploration, SPME '11*, Shanghai, China, May 2011.

2. Michael Beetz, Moritz Tenorth, Alexander Perzylo, and Reinhard Lafrenz. A Knowledge Representation Language for RoboEarth. In *Proceedings of the IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, Workshops and Tutorials, IROS 2010*, Taipei, Taiwan, October 2010.

3. Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629, April 1987.

4. Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433, May 1999.

5. Hans A. Kestler, Stefan Sablatnog, Steffen Simon, Stefan Enderle, Axel Baune, Gerhard K. Kraetzschmar, Friedhelm Schwenker, and Gunther Palm. Concurrent object identification and localization for a mobile robot. *International Journal of Intelligent Systems*, 14, March 2000.

6. Kevin Lai and Dieter Fox. Object recognition in 3d point clouds using web data and domain adaptation. *The International Journal of Robotics Research (IJRR '10)*, May 2010.

7. Li-Jia Li, R. Socher, and Li Fei-Fei. Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, Miami, FL, USA, June 2009.

8. D. Meger, P.-E. Forssen, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J.J. Little, and D.G. Lowe. Curious George: An Attentive Semantic Robot. *Robotics and Autonomous System Journal*, June 2008.

9. Andreas Nuechter, Jan Elseberg, Peter Schneider, and Dietrich Paulus. Study of parameterizations of the rigid body transformations of the scan registration problem. *Journal Computer Vision and Image Understanding (CVIU), Elsevier Science*, 114(8):963–980, 2010.

10. Andreas Nuechter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56:915 – 926, August 2008.

11. F. Remondino, S. El-Hakim, S. Girardi, A. Rizzi, S. Benedetti, and L. Gonzo. 3D Virtual Reconstruction and Visualization of Complex Architectures - The 3D-ARCH Project. In *Proceedings of International Society for Photogrammetry and Remote Sensing (ISPRS '09)*, Trento, Italy, February 2009.

12. Stefan Stiene, Kai Lingemann, Andreas Nuchter, and Joachim Hertzberg. Contour-Based Object Detection in Range Images. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT '06*, Chapel Hill, NC, June 2006.

13. Johan W.H. Tangelder and Remco C. Veltkamp. A Survey of Content Based 3D Shape Retrieval Methods. In *Proceedings of International Conference of Shape Modeling and Applications (SMI '04)*, Genova, Italy, June 2004.

14. S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.