SoftPOSIT for Augmented Reality in complex environments: Limitations and challenges

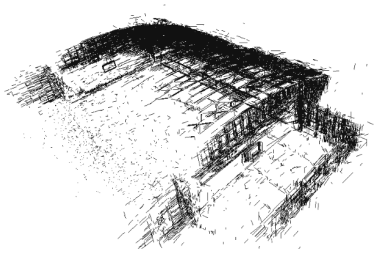
Lewis Baker
Dept. of Computer Science
University of Otago
Dunedin, New Zealand
bakelew@gmail.com

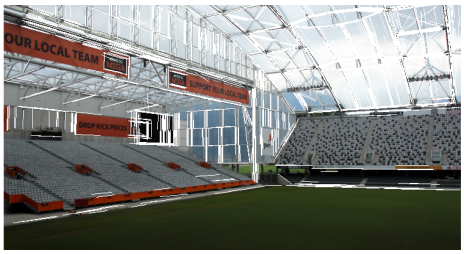
Stefanie Zollmann
Dept. of Computer Science
University of Otago
Dunedin, New Zealand
stefanie.zollmann@otago.ac.nz

Steven Mills
Dept. of Computer Science
University of Otago
Dunedin, New Zealand
steven@cs.otago.ac.nz

Tobias Langlotz
Dept. of Information Science
University of Otago
Dunedin, New Zealand
tobias.langlotz@otago.ac.nz

Image


3D Model



Registration via model-based localization

***Abstract*—Localization in dynamic environments is a challenging problem for Augmented Reality (AR) applications. Model-based localization methods use a prior model and an image to perform localization, and make use of the appearance of the environment. These methods cannot always be relied on, since in dynamic environments, the current appearance may be very different from the model. In this paper, we investigate *SoftPOSIT*, a model-based pose algorithm that does not rely on appearance. We apply line-based *SoftPOSIT* to a range of model types and complexities, and show that while *SoftPOSIT* produces promising results on basic cases, more work is needed in this area to attain useful results in complex real cases and AR applications.**

***Index Terms*—pose, localization, lines, SoftPOSIT**

I. Introduction

Localization in dynamic environments has a wide range of applications to Augmented Reality (AR). A common approach to localization is to use a prior model, and an image to perform the localization. These methods often use the appearance of features to find 2D-to-3D correspondences and compute the camera pose [1], [2]. In some dynamic environments, the appearance is constantly changing, so finding these feature matches is unreliable. An example is a sport stadium, where a prior model may not be able to capture the appearance of the spectators at any given sport event. In this paper, we investigate what can be achieved with localization methods that ignore the appearance of features.

By modelling only the unchanging features in the environment, and detecting those features in 2D, we can eliminate problems caused by appearance-based descriptors. Line features can help achieve this since lines will often correspond with structural components of a given scene, whereas points may correspond with textures. Previous work has demonstrated that line features can be effectively used for localization such as the work by Zhang and Koch [3], but there has been little research in line-based localization with appearance-free lines.

In most cases, appearance-based features are used to help find correspondences between 2D and 3D points or lines. By considering the colour of features, many incorrect correspondences can be easily discarded. Appearance-free methods also exist such as *SoftPOSIT* [4]. *SoftPOSIT* is a point-based algorithm that computes both correspondences and pose simultaneously without using appearance-based descriptors. David *et al.* show that this method can also be applied to lines [5]. *SoftPOSIT* has been tested thoroughly in synthetic test cases, and some real applications using computer-aided design (CAD) models. The results are promising, but it is often not possible to attain CAD models of the environment. Structure from Motion (SfM) has been shown to produce accurate models of large environments easily [6], but *SoftPOSIT* has not been thoroughly tested on these types of models.

There have been a variety of related works in the area of model-based localization, also referred to as image-based localization. Sattler *et al.* show a method for localizing images against a large prior point cloud model [1]. Their initial work used 2D-to-3D feature-to-point correspondences, but the authors later expanded their work to employ an efficient search that leverages both 2D-to-3D and 3D-to-2D correspondences using SIFT descriptors readily available in the model [7]. The work of Li *et al.* is similar in that it combines "forward" and "inverse matching", also making use of SIFT [2]. In our case, we cannot guarantee that the appearance of the environment will be consistent with the model, so this method and other similar approaches cannot be used as they rely on SIFT, or other appearance-based descriptors.

978-1-7281-0125-5/18/$31.00 ©2018 IEEE

Campbell *et al.* show a method for estimating the globally optimal camera pose and correspondences of a set of 2D points to a 3D point cloud [8] without appearance information. The method uses *SoftPOSIT* as part of its computation to speed up convergence to the global optimum. This method is point-based, and has a large computation time for relatively small point clouds. This makes it unsuitable for large line-based SfM models, and interactive AR applications. However, their work supports the need for a pose computation method that does not require appearance-based correspondences, and works toward a similar goal as this paper, and *SoftPOSIT*.

Work by Reitmayr and Drummond supports the idea of using lines (or edges) for correspondence matching for model-based tracking [9]. In their work, they use a textured 3D model and an appearance-based *edgel* matching method in order to align a projection of their 3D model to their mobile camera feed. Their method requires a pose initialization from a known location, so cannot be directly applied to localization from arbitrary positions. This work shows that edges can be applied reliably to localization and tracking problems, though edge appearance is also an important factor.

There is also the Simultaneous Localization and Mapping (SLAM) approach, which aims to create a map of the environment and localize the user with respect to this map concurrently. *PTAM* by Klein and Murray [10], and *ORB_SLAM2* by Mur-Artal and Tardos [11], are some examples of this approach. These SLAM methods, while showing huge potential for AR, have several limitations which make them unsuitable for stadium localization, and multi-user AR. Due to their use of monocular vision in triangulating 3D map points, they require significant translational motion to produce the parallax required to build accurate maps. It is also challenging to align these SLAM maps to a global co-ordinate system that can be shared among multiple users. In our case, we relieve these problems by assuming that a prior model is already available through SfM methods. In this way, we have accurate 3D data, and a global co-ordinate system that multiple users can share.

As previously introduced, *SoftPOSIT* is a method that can be applied to line-models to compute pose without relying on the appearance of the features. *SoftPOSIT* and some of its variations have been applied in some basic real-life cases such as work by Diaz and Abderrahim who employ *SoftPOSIT* for localization of mechanical parts [12]. Since *SoftPOSIT* can also be applied to line models, and does not use appearance-based line descriptors, we have elected it as our localization method for the experiments in this paper.

SoftPOSIT seems to be the most appropriate method that theoretically accommodates our requirement of appearance-free line models. In this paper, we investigate the effectiveness of *SoftPOSIT* in a range of scene complexities and two model formats. With our initial objective being to localize in a sport stadium, we also demonstrate the limitations of this method when applied to other complex SfM models. As *SoftPOSIT* has not thoroughly been applied to real data, our contribution is an overview of the kind of performance that can be expected from this method under real-life use cases.

II. Method

In this section we outline our method for experimentation and evaluation of *SoftPOSIT*. Since we already know from previous research that this method works in basic cases, we build a dataset of increasing complexity from a small CAD model of a line-based object to a large 3D reconstruction of a sport stadium. The process consists of: ground-truth pose acquisition, model creation, and localization to this model with *SoftPOSIT*. The method is applied to five test cases across two types of models. In the following subsections, we will explain our line-based *SoftPOSIT* implementation, our two model types, our method for acquiring ground-truth pose, our pre-processing steps, and our dataset.

A. Localization with SoftPOSIT

As explained in Section I, we choose *SoftPOSIT* as our candidate for localization since it requires no information of line appearance, and is less computationally complex compared to other approaches. In this section, we explain our implementation of *SoftPOSIT*, and some preprocessing steps that we introduce that can reduce computational complexity.

SoftPOSIT is an algorithm that can be used for model-based localization. It takes as input an initial estimate of the pose, a 3D model (line or point based), and an image of the model [4], [5]. The algorithm has been extensively tested on synthetic examples, and virtual examples [14].

The method works by using the pose estimate to project model lines into the image. It then creates an initial assignment matrix based on the geometric differences between the projected 3D lines, and the observed 2D lines. This step is based on the *SoftAssign* method [15]. This matrix is normalized, and then used by the *POSIT* algorithm to produce an updated pose estimate [16]. Finally, this process is repeated for several iterations until an accurate pose is achieved. Figure 1 shows the first six iterations on our basic Dodecahedron test case.

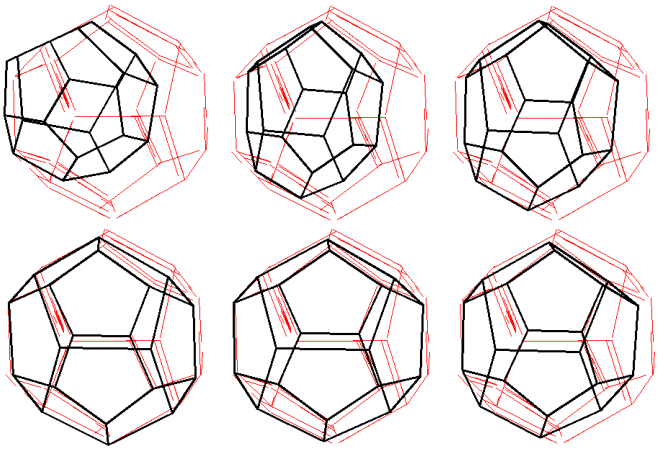


Fig. 1. Six iterations of *SoftPOSIT* running on our basic Dodecahedron case (clockwise from top-left). The thin red lines represent the 2D lines in the photograph detected by LSD [13]. The black lines represent the CAD model rendered with the current pose. Initially the pose estimate is far from correct, but iteratively gets closer until the black and red lines align.

SoftPOSIT has several parameters that we fix empirically in our experiments. The parameter α acts as a threshold for the assignment matrix. When 3D lines are projected using the pose estimate, each 3D line has its *difference* to each 2D line computed, and differences above this threshold will be assigned a correspondence that tends to zero. The difference is computed as a weighted combination of the angle between the lines, and the distance between their endpoints. We found that an α value of 0.5 produced good results.

The second parameter β controls the rate of convergence, and is increased at each iteration of the algorithm. Small values for β will search a larger space of possible poses, but can create larger execution times. As suggested in the original work, large β values are better suited for cases where we know that the pose estimate is close to the actual pose [5]. We have this parameter set to 20. We observed that values lower than 20 caused the first few iterations of *SoftPOSIT* to consistently diverge from the correct pose on our Stadium test case.

B. Model types

There are two different types of 3D models that we consider. The first, which we refer to as *CAD models*, are created manually in a 3D modelling software, where all visible lines on the object are represented in the 3D model. The physical objects that we targeted for these cases have simple geometry and lack complex texture which make them suitable for manual modelling. Our CAD models were measured and modelled to scale using *Blender* modelling software [17].

The second model type is a line-based *SfM model*. For these models, we use point-based *VisualSFM* system by Wu [18], [19], and feed its output to *Line3D++* by Hofer *et al.* to create the final line model [20]. In this paper, we assume that creating these models in advance is a required step for localization in a large space such as a stadium. This is a fair assumption, as these types of software are straightforward to use, and can produce high quality models with little effort. In our stadium example, we captured 604 photographs from various locations on one stand, and fed them through the SfM pipeline above to create the final result.

C. Ground-truth pose

In our SfM test cases, we simply take the output generated by *VisualSFM* and consider these as our ground-truth poses. The images we use for testing the localization method are a random selection of 20 SfM input images.

For the CAD models, we placed a chessboard marker of known dimensions at a measured distance from the target object. Using standard techniques and known intrinsic camera parameters, we detect the chessboard corners in the image, and use this to compute the pose [21]. We then offset the origin of our 3D model by its known translation from the chessboard origin. Finally, the chessboard is masked out of the input when passed to the localizer to prevent the chessboard lines from interfering with the localization result.

While both of these methods can be expected to produce a small amount of error, our main interest is in the rate of localization, and speed rather than accuracy. In our experiments, we compute a reprojection error by projecting the model lines with both the ground-truth pose, and the estimated pose, and comparing the corresponding projections. We then introduce an error threshold which will encompass the small errors in our measurements, resulting in a metric for the rate of success.

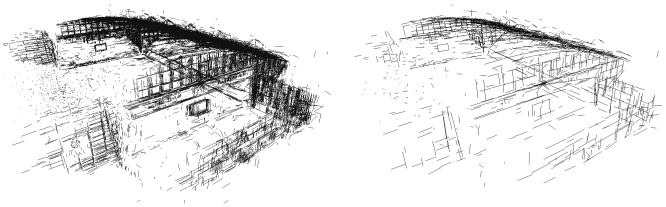


Fig. 2. Example output from k-means clustering of the 3D line model. The left image shows the original output from *Line3D++* which contains over 22,000 lines. The right image shows the output of performing k-means with k set to approximately 2,200 (10%).

D. Pre-processing and reduction of complexity

We found that the execution time of *SoftPOSIT* was high when run with large SfM models. Our initial stadium model produced by *Line3D++* contained over 22,000 3D lines. An example photograph at original resolution contained over 9,000 lines. Execution time at this level of complexity is ungainly and not useful or practical, so we considered some approaches to reduce the complexity for our experiments.

We reduced the number of 2D lines by scaling our input images down to quarter-size. This scaling was applied to all five of the test cases in our dataset. For the SfM models, the number of 3D lines was further reduced by clustering similar 3D lines in the model using k-means [22]. The result of our clustering (with small k for illustration purposes) is shown in Figure 2. For each SfM model, we set k to be equal to half the number of 3D lines in the original model, which is notably denser than the example on the right in Figure 2.

Computing k-means clustering involves a metric for determining the distance between the line segment observation and the cluster centres. In our implementation, the distance was taken as the 6D distance between each line described as the union vector of its endpoints. Two 3D lines can be identical but have different ordering of their endpoints when they are described as a 6D vector. For comparing two line segments L and L', we use

$$\begin{aligned} D_1 &= ||L_{p1} - L'_{p1}||^2 + ||L_{p2} - L'_{p2}||^2 \\ D_2 &= ||L_{p1} - L'_{p2}||^2 + ||L_{p2} - L'_{p1}||^2 \end{aligned} \tag{1}$$

where L_{pn} is the n^{th} endpoint of L, and take the minimum of the distances D_1 and D_2 to determine which of the line pair's endpoints are corresponding, and re-order the endpoints in the vector if necessary.

Since we are assuming that we have an initial estimate of our pose (as this is a requirement for *SoftPOSIT*), we employ one other basic line filtering mechanism. We assume that the pose estimate is reasonably good. Before *SoftPOSIT* is run, we remove all lines from the 3D model that are not visible given

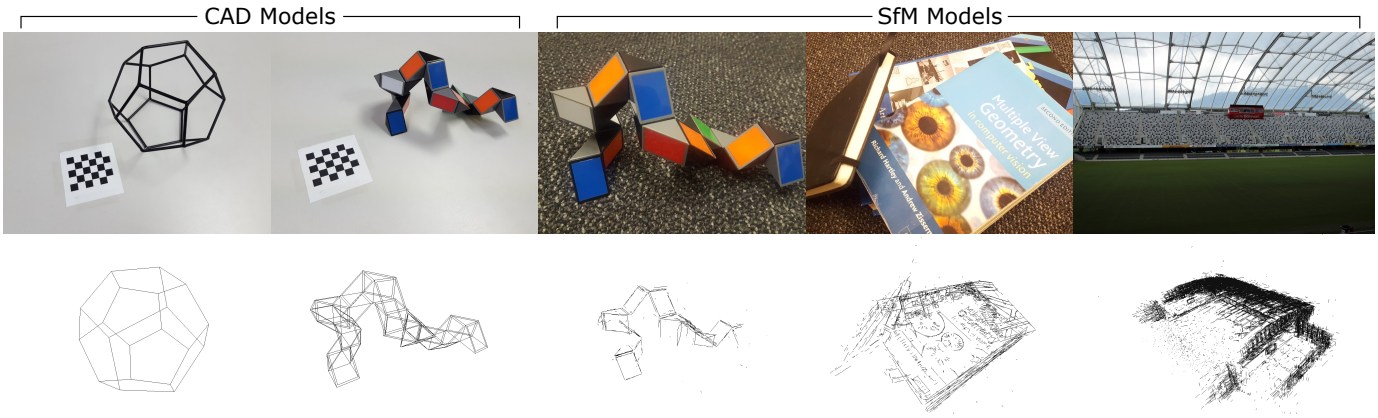


Fig. 3. The top row shows a sample image from each test case, the bottom row shows a render of the 3D model we used for that test case. The CAD models were created manually, and contain all of the visible lines in the object. The SfM models are generated by a combination of *VisualSFM* and *Line3D++*, and are generally sparser and noisier than the CAD models.

the initial pose estimate. This will remove a large portion of lines which are in the model but not the image. Some observed model lines will also be removed as the estimate is not always perfect, but the idea is that *SoftPOSIT* will be robust to some occlusion as was shown by David *et al.* [4].

E. Dataset acquisition

Our objective is to explore *SoftPOSIT*'s ability to function under test cases of varying difficulties and model types. We constructed a dataset of five test cases, where each test case contains 20 sample photographs. The subject of each test case is an object or environment of which we have a prior model. Figure 3 demonstrates a sample image from each test case, and the corresponding 3D line model that is tested against.

The dataset consists of CAD models of two subjects: a wireframe dodecahedron, and a snake puzzle. It also contains SfM models of three subjects: the same snake puzzle, a stack of books, and a sport stadium. The test cases cover some of the key properties that can cause problems for a model-based localizer, such as occlusion, noise, distractor lines, and imperfections in the model.

III. Experiments and Results

In all tests, we set an upper limit of thirty seconds on the execution time unless stated otherwise. We deem any execution time over thirty seconds to be well and truly over the threshold of what is useful for an AR application, even allowing for careful optimization. In these cases, the tests are terminated early, and the reprojection error is taken based on the last computed pose.

To determine the accuracy of the localization, we project the visible model lines into the image using both the ground-truth pose, and the final pose output by the localizer. We then compare these projections, and compute the reprojection error as the average of the distances between each line endpoint. We present the final error as a percentage of the image width, as well as pixels at 1920×1080 resolution

We are also interested in the rate of successful localization. To determine the success rate, we set a threshold on the reprojection error, and compute the percentage of test cases that produce a pose that falls below this threshold. In the following experiments, we set the error threshold to 2% of the image width. This is a rather accommodating threshold, corresponding to over 30 pixels at 1920×1080 resolution. This value was chosen as it is large enough to encompass any error contained in our model and pose data.

Our first experiment looks at the effectiveness of our pre-processing methods, the second and third experiments show the overall results of *SoftPOSIT* on our dataset under both basic, and typical use cases respectively.

A. SfM model preprocessing

In Section II-D we introduced the idea of preprocessing the SfM models in order to reduce the complexity of localization with *SoftPOSIT*. In this experiment, we show that the execution time of *SoftPOSIT* can be reduced using a preprocessing method that aims to reduce the number of 3D lines. In the Stadium case, we have a large SfM model that details nearly all lines in the stadium. However, in most cases, a lot of these 3D lines do not lie in the spectator's field of view.

In this experiment, we tested 2 variations of k-means clustering of the 3D lines, and a visibility filter based on the initial pose estimate. The first variant of k-means filters the 3D model and outputs the means (or cluster centres) as the final model (we refer to this as KM-M in Figure 4). The second variant of k-means outputs the longest input line assigned to each cluster as the final model (referred to as KM-L). The visibility filter is applied to each individual test case, rather than the entire model. It essentially discards any lines that are not visible, given the initial pose estimate (referred to as VF).

We apply this experiment to two of our SfM datasets, but not the CAD datasets, as the purpose of the filtering is to reduce spurious lines and noise which are not present in the CAD models. We leave out the Books SfM case, as we show in the following subsections that this case is infeasible. We apply a random pose error to the ground-truth pose of each sample in order to simulate real-world inaccuracies in the pose

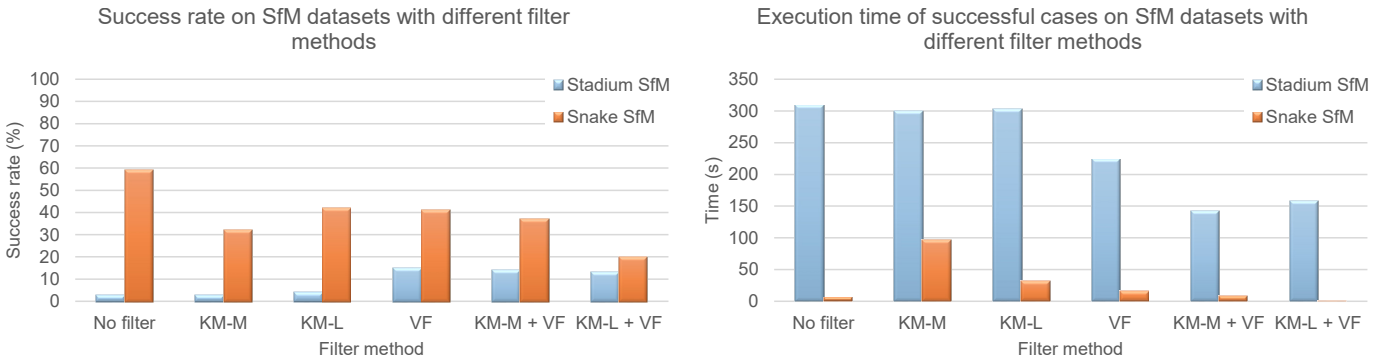


Fig. 4. Comparison of success rate (left) and execution time (right) of different line clustering methods on two SfM models in the dataset. The filter methods are as follows. KM-M: k-means outputting the means of each cluster. KM-L: k-means outputting the longest line in each cluster. VF: visibility filtering.

estimate. Figure 4 shows the success rate, and execution time of each clustering method in the two selected datasets.

Figure 4 shows the results of this first experiment. All test samples were capped at an execution time of 5 minutes instead of 30 seconds to increase the rate of success. Even this large cap is reached, which accounts for the low success rate of the Stadium SfM cases with KM-M, and KM-L filtering. The VF filtering method seemed to have the most visible effect on execution time, bringing the Stadium SfM test samples below the 5 minute execution cap, and increasing the success rate.

The Snake SfM results suggest that if execution time is not a priority, the best results will be obtained with no filtering applied. But in practice, it is not feasible to run for extended periods of time. So applying a filtering method such as VF or a combination of VF and KM-M, can reduce the execution time. More thorough testing of the effect of these filtering methods is left for future work. For the remainder of the experiments, we set the execution cap to 30 second, and use KM-M + VF since it resulted in the best execution times with similar success rate compared to VF for the Stadium SfM test case.

B. Basic use-case

Table I shows the results of our experiments in an ideal use case. In this experiment, we fix *SoftPOSIT*'s parameters, and provided the ground-truth pose as the initial pose estimate. This was done as a base case, to determine how effectively it can localize under the unlikely but valid circumstance that the pose estimate is highly accurate.

TABLE I
SOFTPOSIT BASIC USE-CASE

Test case	Dodec.	Snake_1	Snake_2	Books	Stadium
Model type	CAD	CAD	SfM	SfM	SfM
Success (%)	90	95	40	0	20
Err. Succ. (%)	0.888	0.980	0.865	N/A	0.408
Err. Succ. (px)	17.0	18.8	16.6	N/A	7.8
Time (ms)	50	1585	691	2465	18956

The results show that even in the basic case, localization using SfM models is difficult. The CAD test cases worked relatively well, with an average success rate of over 90% for both. But all of the three SfM cases had low success rates, and the Books case failed to produce an accurate localization result in all of its samples.

The Books case, while not completely planar, contains a large proportion of lines on the surface of the top-most book. Scenes consisting of co-planar lines are known to be a failure case of the *POSIT* algorithm, so this result is not surprising. A variation of *POSIT* (pose from known correspondences) for co-planar scenes is presented by Oberkampf *et al.* [23], but there has been no work in applying this to *SoftPOSIT* for points or lines when the correspondences are unknown.

An interesting result is the Snake test, for which we had both CAD and SfM models. Our SfM pipeline was unable to capture all the lines that were present in the object, so the Snake SfM model was sparse compared to its CAD counterpart. In the cases where the SfM Snake succeeded, it did produce a slightly lower reprojection error than the CAD model. This could be to do with imperfections in the physical joints of the Snake puzzle, which caused it to misalign slightly with the "perfect" CAD model. Since the SfM model was produced using actual observations of the object's edges, the lines that are present have the potential to be highly accurate. Performing a t-test on both complete Snake datasets, we found a p-value of 0.0211, which tells us the Snake SfM model resulted in a small but significant improvement of reprojection error in successful cases compared to the CAD model, at the cost of a much lower success rate.

It is also notable that while the Stadium SfM case had a low success rate, it also produced a relatively small reprojection error in success cases. While the success rate leaves much to be desired, it shows promise in its application to AR with an average reprojection error of 0.408% image width (7.8 pixels at 1920×1080 resolution).

C. Typical use-case

In this experiment, we look at the performance of *SoftPOSIT* under a more typical use case, where the input pose estimate is erroneous. In an AR application, such an estimate could be obtained from mobile device sensors.

In this experiment, we do not use mobile sensors to acquire a pose estimate. This is because our SfM models are scaled

arbitrarily, and none of our models are aligned to GPS coordinates. Instead, we artificially add error by adding random adjustments to the ground-truth camera pose. Since *SoftPOSIT* localizes by minimizing the 2D error of projected model lines, we pick a pose adjustment that results in a random reprojection error in a predetermined range. In this way, the 2D error added is consistent regardless of the scale of the model.

TABLE II
SOFTPOSIT TYPICAL USE-CASE

Test case	Dodec.	Snake_1	Snake_2	Books	Stadium
Model type	CAD	CAD	SfM	SfM	SfM
Success (%)	94	81	36	0	12
Err. Succ. (%)	0.926	1.049	0.744	N/A	0.496
Err. Succ. (px)	17.8	20.1	14.3	N/A	9.5
Time (ms)	80	1646	680	2556	18693

The results in Table II show the success rate, error, and execution time of each of our models when the ground-truth pose is modified randomly. Each test case is run 5 times, and the averages are presented in the table.

Compared to the basic use-case, the success rates are mostly lower as expected. The exception here is the dodecahedron model. This case was simple, was generally insensitive to inaccuracies in the pose estimate, and often succeeded in cases where the estimate was drastically wrong. The higher percentage here could be attributed to the larger sample size (5 runs per image) due to the randomness in the pose estimate.

IV. CONCLUSION AND FUTURE WORK

In this paper we outlined the problem of model-based localization, specifically in cases where our image may not be similar in appearance to our prior 3D model. We implement a line-variant of *SoftPOSIT*, and apply it to some challenging real-life examples using different model types.

Our results showed that *SoftPOSIT* can be effective in some real-life cases, particularly when the model is simple. In these cases, it can achieve a good rate of localization with short execution times. We also showed that with some SfM models, good results can be achieved but not as reliably, and that more work is needed to achieve good results in complex scenes.

In the future, we would like to experiment with other common types of 3D models, such as triangulated meshes. Meshes are not immediately suited to this type of localization, as they often contain many edges that do not correspond to visible geometry or texture. We also aim to experiment with alternative methods, or formulations of *SoftPOSIT* that are robust in localizing complex SfM models.

ACKNOWLEDGMENT

We thank Forsyth Barr Stadium, and Dunedin Venues Management Ltd. for providing stadium access to capture our dataset. This project is supported by an MBIE Endeavour Smart Ideas grant.

REFERENCES

[1] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2D-to-3D matching," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, nov 2011, pp. 667–674.
[2] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide Pose Estimation Using 3D Point Clouds." Springer, Cham, 2012, pp. 15–29.
[3] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, oct 2013.
[4] P. David, D. Dementhon, R. Duraiswami, and H. Samet, "Softposit: Simultaneous pose and correspondence determination," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 259–284, 2004.
[5] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, "Simultaneous pose and correspondence determination using line features," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. II–II.
[6] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 72–79.
[7] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1744–1756, sep 2017.
[8] D. Campbell, L. Petersson, L. Kneip, and H. Li, "Globally-Optimal Inlier Set Maximisation for Simultaneous Camera Pose and Feature Correspondence," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, sep 2017, pp. 1–10.
[9] G. Reitmayr and T. W. Drummond, "Going out: Robust model-based tracking for outdoor augmented reality," *Proceedings - ISMAR 2006: Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality*, no. May, pp. 109–118, 2007.
[10] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2007.
[11] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, oct 2017.
[12] J. Diaz and M. Abderrahim, "Modified SoftPOSIT algorithm for 3D visual tracking," *2007 IEEE International Symposium on Intelligent Signal Processing*, pp. 1–6, 2007.
[13] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.
[14] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, "Evaluation of the softposit model-to-image registration algorithm," *Evaluation*, 2002.
[15] A. Rangarajan, H. C. I, and F. L. Bookstein, "The Softassign Procrustes Matching Algorithm," *Lecture Notes in Computer Science*, vol. 1230, pp. 29–42, 1997.
[16] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *International Journal of Computer Vision*, vol. 15, no. 1-2, pp. 123–141, jun 1995.
[17] Blender Foundation, "Blender." [Online]. Available: https://www.blender.org/
[18] C. Wu, "Towards linear-time incremental structure from motion," in *3D Vision-3DV 2013, 2013 International Conference on*. IEEE, 2013, pp. 127–134.
[19] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3057–3064.
[20] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3D scene abstraction using line segments," *Computer Vision and Image Understanding*, vol. 157, pp. 167–178, apr 2017.
[21] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.
[22] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
[23] D. Oberkampf, D. F. DeMenthon, and L. S. Davis, "Iterative pose estimation using coplanar feature points," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 495–511, 1996.