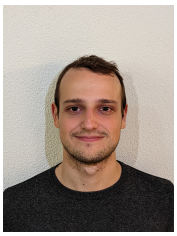
Floor Plan Generation from Tracked Physical Objects

Mixed Reality Midterm Report
Supervised by: Dr. Anton Savov, Wenqian Yang
November 14, 2022

GROUP MEMBERS

Thomas Poyet 

Hugo Queinnec 

Lucas Teissier 

Gonca Yilmaz 

I. CURRENT STATUS

The responsible group member for each task is specified below between curly brackets: {Name}.

A. Furniture Tracking

During the first phase, we experimented with libraries such as Azure Object Anchors {Thomas} and Vuforia {Thomas, Lucas}, that can track 3D objects.

Azure Object Anchors was not able to detect our 3D printed objects which is certainly due to their size. The recommended object size is between 1m and 10m, however, our objects measure only around 10cm. Hence, we focused our work on Vuforia.

First, we tried a 2D image target approach {Lucas}, which turned out to work quite well under specific conditions. For this approach, we need furniture that is flat enough to stick 2D printed images (like a geometrically textured blanket) that allow Vuforia to detect key points to track the object. However, there are some limitations, described in section II. Still, once this tracking worked, we were able to augment the sliders and links on top of the furniture so that they stick to them.

We also tried to use Model Targets by Vuforia, which allows 3D objects tracking {Thomas}. There are two ways to use it. The first one requires aligning the 3D object with an outline displayed on the screen. However, this first step is too constraining and not very user-friendly. Moreover, the tracking was easily lost when moving the objects or the camera around. The second way, the one we chose to explore more, is called Advanced Model Target. It is based on a deep learning approach and requires a training step, done on Vuforia servers. With this method, users do not have to align the object on a target, and tracking is more stable. In this case, the user scenario will be to have users seated, such that the 3D objects are seen from their side.

We also looked at some research papers on this subject (cf. section IV) {Thomas}. The BOP challenge [2] was a precious source of information. The winner of the challenge was CosyPose [3]. However, using this model would require fine-tuning it on a custom dataset, which can be generated with a Blender tool developed by BOP: BlenderProc4BOP. Then we would need to find a way to run this model on Unity, Barracuda seems to be one. It is difficult to predict the inference time on the Hololens, as speed measurements are done on computer hardware. For all these difficulties and unknowns, we chose to work with Vuforia.

B. User Interface for Inputs

User input needs to be collected to obtain the size of the room and the connections between the furniture which represents the links between rooms. During floor generation, those rooms will have a passage connecting them.

First, we designed an intuitive user interface for the inputs we have to collect {Hugo}. After creating a sketch of the user interface, we presented it to the advisors to get some feedback. Once we agreed on the user interface, we implemented it in Unity using C# {Hugo, Gonca}. Our first implementation works without furniture tracking. We show the user interface on virtual furniture (that can be virtually moved). The user can use the slider to choose the

room size, and can drag a virtual sphere from one furniture to another to create a link. Links can be removed by repeating this action.

Then, a graph containing all the information (from user inputs and object positions/rotations) is generated in real time {Hugo}. This graph could be passed to the API as JSON for the floor plan generation.

II. Identified Challenges

- For 2D Image tracking, we need to fulfill some conditions to make it work well. First, it works because we assume to look the furniture from above with the Hololens, so that the 3D objects appear as 2D images. Then, we must have well-detailed textures and a good lighting so that Vuforia detects the key points. Vuforia provides some practices to help get a good texture, such as enhancing the contrast and using well-distributed features.
- For Model Target, our objects are challenging to detect and track as they are small, texture-less, color-less and symmetrical. That is why, we plan to tune their design.
- Physical objects will have to be placed in a virtual boundary showing the perimeter of the apartment to be taken into account. It will be necessary to recognize if an object is actually inside this boundary or not.

III. Next Steps

A. Object Tracking

1) Image-Based Tracking {Lucas}:

- To get better 2D images for all the furniture, following the Vuforia documentation: Image target Best Practices
- To continue to integrate the UI interface with tracking.

2) 3D-Based Tracking {Thomas}:

- To improve the tracking by tuning the 3D models of the objects according to Vuforia documentation: CAD Model Best Practices
- To run the tracking using a Hololens and not a webcam.
- To integrate the UI interface with tracking.

B. User Interface

Now that the basic user interface works and can generate a representative graph, the goal is to integrate it with the floor plan generation API {Hugo, Gonca}. We should be able to quickly send the JSON graph to receive the final visualization of the room with the 3D mesh representing the walls.

One thing remaining to be implemented is the house boundary to be displayed to the user once they start the application {Hugo, Gonca}. The user should be able to rotate and transform the displayed boundary before they start placing the furniture.

If the object tracking works reliably well in real-time, we may adapt the user interface so that the user can drag-and-drop the actual object instead of the virtual sphere to create the links {Hugo, Gonca}. This could be challenging due to occlusions and lag in the computations. As final step, we will integrate the user interface with other components {All}.

C. Visualization of 3D Mesh

The floor plan generated by the server needs to be visualized to the user as the final outcome. The 3D mesh of the floor plan should be overlaid on the furniture based on the given transformation and rotation. This part can be done independently by taking a floor plan mesh. We will implement this component in Unity with C# {Gonca}. Once this component is working and the API to generate the floor plan is available, we will integrate this component with the rest of the system {All}.

References

[1] Yannick Bukschat and Marcus Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach. *arXiv preprint arXiv:2011.04307*, 2020.

[2] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. Bop challenge 2020 on 6d object localization. In *European Conference on Computer Vision*, pages 577–594. Springer, 2020.

[3] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[4] Van Nguyen Nguyen, Yinlin Hu, Yang Xiao, Mathieu Salzmann, and Vincent Lepetit. Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6771–6780, 2022.

[5] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 292–301, 2018.

IV. Annex

A. 6-dof Estimation Review

Our project requires detecting and tracking objects, which means estimating :

- The position of the object
- The orientation of the object
- The type of object (bed vs. sofa...)

Indeed, this information is necessary to be able to display the virtual user interface over the objects.

This type of task is generally called 6 degrees of freedom estimation, in the literature. To have an idea of the state of the art in the field, one can have a look at the BOP challenge [2] and also at this web page PaperWithCode which references articles with code on this subject.

1) Already Available Dataset: To compare the different methods, one should compare their test score. However, different datasets are available to train and evaluate a model. Here, we compare the characteristics of the objects found in each dataset.

	Symmetrical	Texture Less	Color Less	Same Size	Occlusion
LM	No	Yes	No	No	Low
LM-O	No	Yes	No	No	High
T-Less	Yes	Yes	Yes	Yes	Low

In our case, all objects are 3D printed without texture, with the same white color, have the same size, and are symmetrical. Therefore, T-Less is more appropriate. However, since strong occlusions will occur when users move the shape, attention should also be paid to the LM-O score.

2) Synthetic Dataset: Regarding the dataset, we also have to answer the question of creating our own dataset.

Indeed, all the methods presented below are deep learning based. Therefore, almost all of them are only able to estimate the 6dof of objects they saw during training. Therefore, in order to apply these methods to our own objects, we must first fine-tune the model on a dataset consisting of these objects.

Creating a dataset composed of real images will be very tedious because each image must be associated with the bounding box of the object in it. Therefore, it will be more realistic to use a synthetic dataset. Instead of real images, it would be synthetic scenes rendered by computer, which would make the bounding box immediately available.

Fortunately, tools can be found to generate such synthetic datasets:

- BlenderProc4BOP, used in the BOP challenge, creates photorealistic images. This leads to a better test score as photorealistic images look more like real ones than OpenGL-rendered images.
- In the CosyPos GitHub [3], there is a tool to create synthetic data, but generated images are not photorealistic.

3) BOP results: Among the 26 models evaluated in the BOP challenge [2], the one that performed best on the T-Less dataset is CosyPose-1view [3]. The inference time is about 0.5 s, which makes the model almost suitable for real-time.

4) Paper with Code results: The real-time requirement of our application motivates the search for papers presenting real-time methods, which is not really the case with CosyPose [3].

- EfficientPose [1]: inference time takes 0.04s
- Real-Time Seamless Single Shot 6D Object Pose Prediction [5]: inference time takes 0.02s

However, for both cases, the inference time is calculated on hardware that is not comparable to the Hololens. The inference time could only be slower on the device. Moreover, both models have not been evaluated on the T-less dataset, making it more difficult to know how well they will work on our objects.

Finally, some methods bypass the issue of retraining/fine-tuning on our own dataset. For example, Templates for 3D Object Pose Estimation Revisited [4], once trained, is able to generalize to unseen objects whose 3D files are known.

5) Conclusion: The two most promising models to explore are CosyPose [3] and Templates for 3D Object Pose Estimation Revisited [4], both implementations can be found already trained on the T-Less dataset on their GitHub. CosyPose [3] would require to be fine-tuned on a custom dataset, which can be generated with the Blender tool: BlenderProc4BOP. Then we would need to find a way to run this model on Unity, Barracuda seems to be one. However, it is difficult to predict the inference time on the Hololens, as speed measurements are done on computer hardware.