Exploration Of VSlam

Nicholas Shaheen

The Process Of VSlam

The goal of visual Slam:

The goal of VSlam is to compare multiple frames from a video and highlight key points and features that can be compared to their other frames to detect angular motion and shifts in the alignment of the scene.

The Objective:

I will be following along with the VSLAM code provided as a sample in MATLAB(**found at https://www.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html)** to determine an office. I will then try to replicate this work myself with my home office.

Creating 2 basic frames of the video:

Two images are thrown into a system that creates an algorithm including key points for each. These images are then placed into data files that allow code to create the initial generation of their plotting of key points into the system. The code then uses a triangulate function to determine the key points and create.

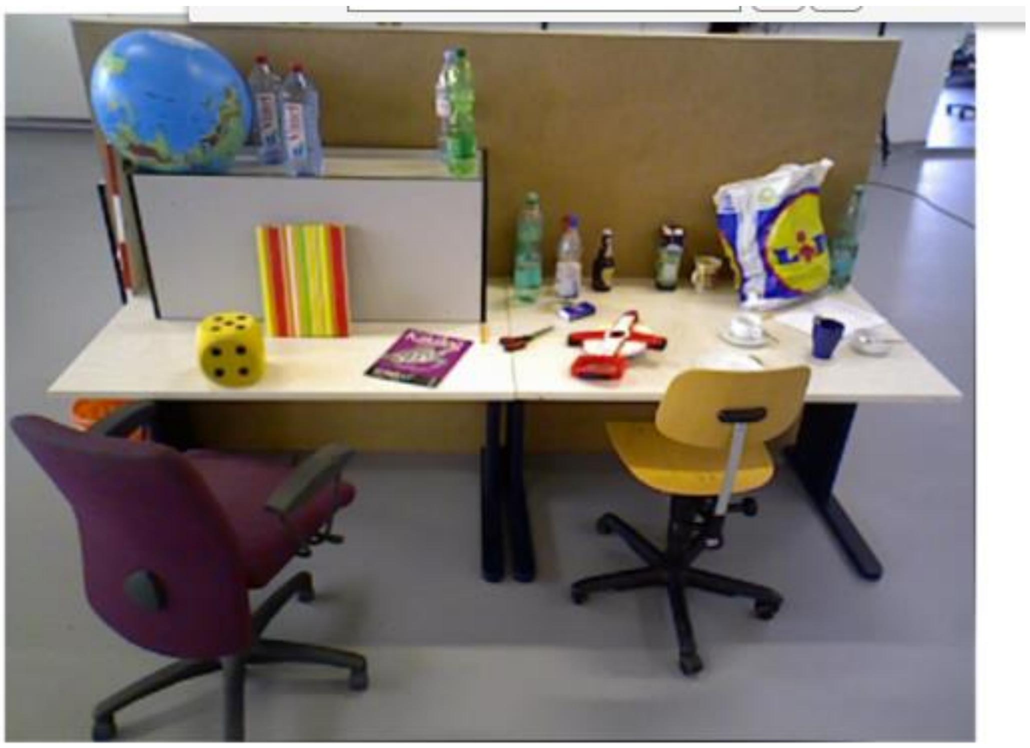


Figure 1 Initialization of photo based on where MATLAB has run the extraction and found the first frame of the system to identify the first key point.

Map Initialization:

The code then uses a triangulate function to determine the key points and create a pixelated map and determine a principal point to extract the key features from the image. A matrix is generated using this information to map the key points based on their pixel coordination. These points are then compared based on the storage to find where the frames are aligned in comparison to each other.

```
Map initialized with frame 1 and frame 28
```

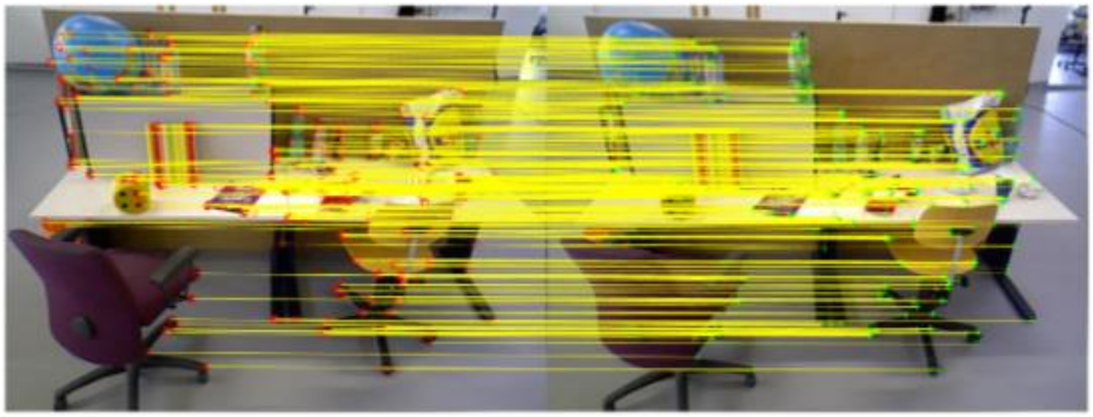
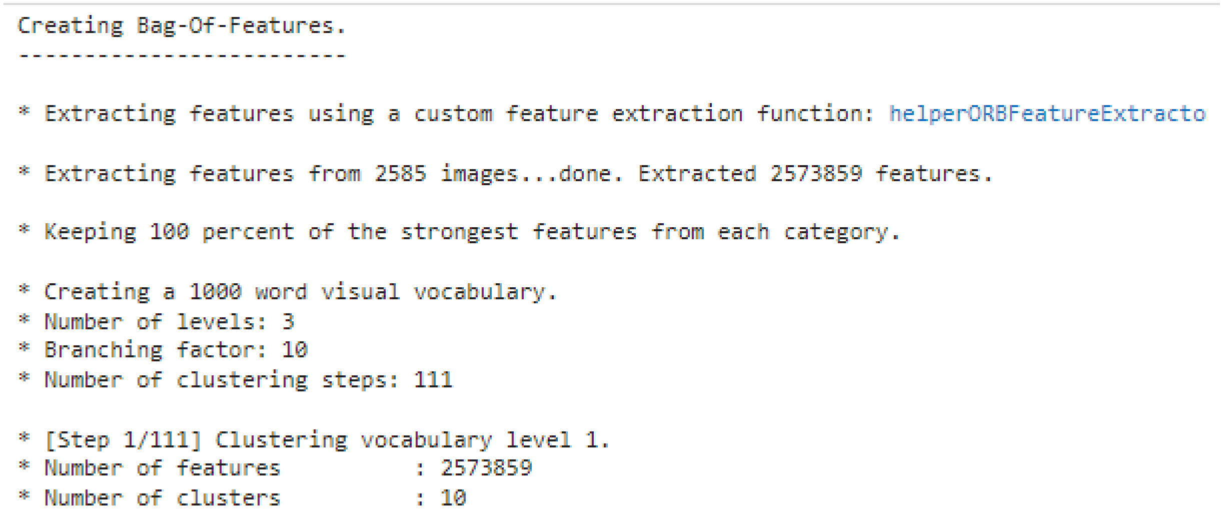


Figure 2 Comparison of frames 1 and 28 of the office sample provided by MATLAB's sample code of VSLAM

Storage and Place Recognition:

The dataset of these comparisons is then contained in the system where it can then be reconstructed to find three-dimensional locations. This step expands the matrix to the third dimension. This then sets points to determine where the key points are in the third dimension. These points are plotted to create an image where each new location in the third-dimension changes. It creates clusters of pixels and stores them to project onto the system.



```
Creating Bag-Of-Features.
-------------------------

* Extracting features using a custom feature extraction function: helperORBFeatureExtracto

* Extracting features from 2585 images...done. Extracted 2573859 features.

* Keeping 100 percent of the strongest features from each category.

* Creating a 1000 word visual vocabulary.
* Number of levels: 3
* Branching factor: 10
* Number of clustering steps: 111

* [Step 1/111] Clustering vocabulary level 1.
* Number of features          : 2573859
* Number of clusters          : 10
```

Figure 3 Cluster Save data based on VSLAM readings

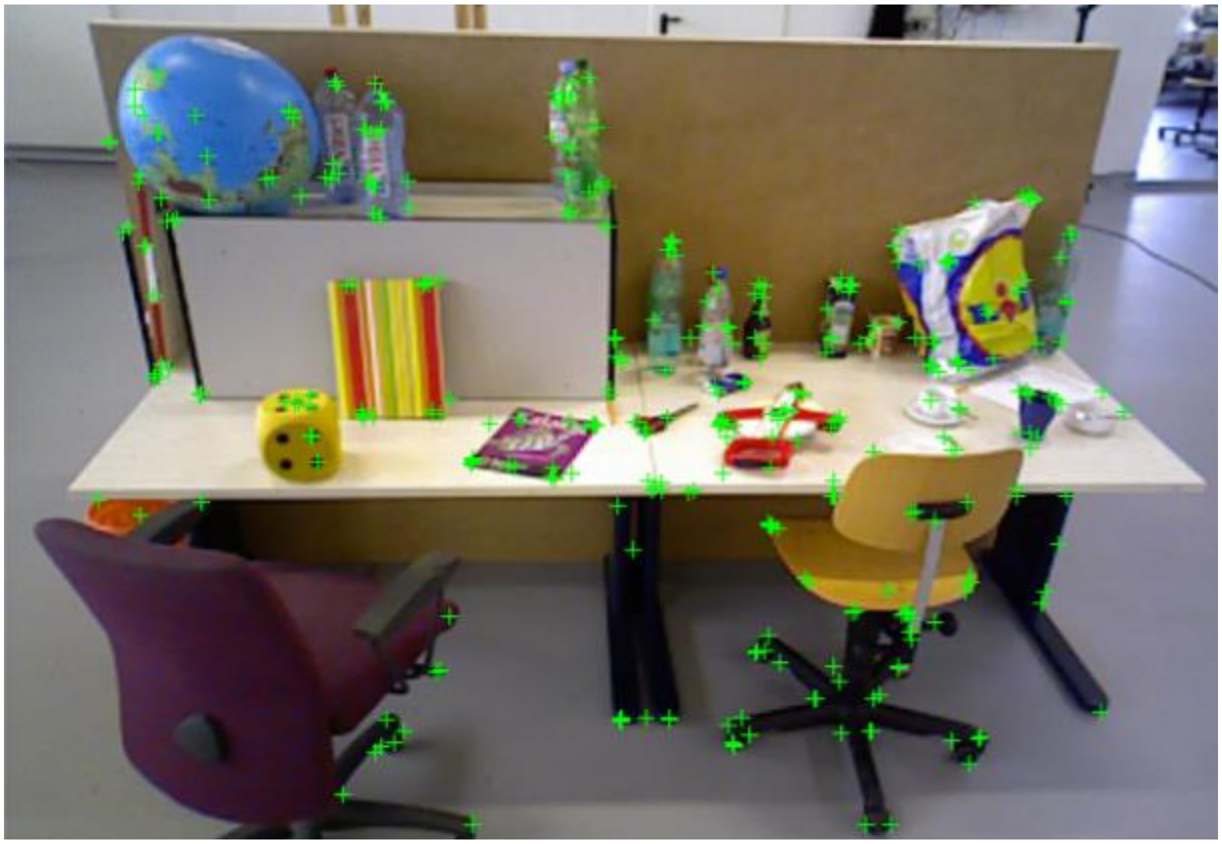


Figure 4 Cluster dataset and key points in 3D

Tracking loop:

The ORB features are then matched with the last key frame for each new frame to create a 3-D and 2-D correspondence to the previous frame. This is repeated for each frame until a key frame is hit. When a key frame is hit it is then identified and compared to a previous key frame. This loop reiterates for each key frame in the system. Typically, 20 frames apart from each other. The system is reiterated and creates these new key frames to set a new comparison value and determine the new and previous locations to this key point.

Comparison with a Ground Truth map:

The Slam system is then set to close its loops by finalizing the frames and their location in relation to each other and determine the ground truth map. This map can then be used as a comparison of estimation of the VSLAM algorithms in comparison to the actual truth of the system. This creates a final VSLAM map with a coordinate frame showing the reference and the other parts of the VSLAM model that was generated through algorithms. The finalized system then takes the estimated vs actual trajectory and looks for an optimized trajectory to satisfy its tuning parameters which can be adjusted. Unfortunately this part would not work due to a improper library implementation so I was not able to replicate this portion after hours of trial and error.

Replication Issues and Errors:

When trying to replicate this on a video of a room, the main issue I faced was trying to perform the conversion of the file. I was unable to convert the file properly into the format that would be allowed to interact with the system. The file would be saved as a .tgz and would not process into the folders as it needed to. Once the file attempted to replicate it, the folder path and the directive would not be followed properly by the system. This made it so that it was completely impossible to replicate this system with a video of my own. Unfortunately, I have not been able to complete this task after countless hours of looking into this problem. Some solutions that I have tried are changing the format, the folder path, and the system's links in general. This has all proved to be useless in the efforts to make this system work properly though. I believe the main issue lies in the file type not being proper and I have done research and tried many different iterations, but I have not been able to generate the proper file type.

Code Trial

```
baseDownloadURL =
"https://s181.convertio.me/p/_HX5Pm5P_Yp9mavZHoqvwA/7c91f585e9758b62bd9c71c98b472
63c/img_3745.tgz";
dataFolder      = fullfile(tempdir, 'house_rgbd_database', filesep);
options         = weboptions(Timeout=Inf);
tgzFileName     = [dataFolder, 'fr3_house.tgz'];
folderExists    = exist(dataFolder, "dir");




if ~folderExists
    mkdir(dataFolder);
    disp('Downloading fr3_house.tgz. This download can take a few minutes.')
    websave(tgzFileName, baseDownloadURL, options);


    disp('Extracting fr3_house.tgz ...')
    untar(tgzFileName, dataFolder);
end
imageFolder   = [dataFolder,'house_rgbd_database\rgbd_dataset_house/rgb/house'];
imds          = imageDatastore(imageFolder);
%Unfortunately the code stops working at this step and the replication
%becomes impossible with this image path as the file will not be placed
%properly
```