Machine Vision - Deriving AKAZE/SIFT Keypoint Pairings from YOLO Segmentation Pairs to Generate 3D Point Clouds

Amari Urquhart

1 Background

This paper covers a series of techniques I employ to derive multiple 3-D object point clouds automatically from a stereo image scene. This involves analyzing the natural epipolar geometry of the images and leveraging that geometry for sub-image pairings we will find. I utilize YOLO (You Only Look Once) object detection to automatically identify and segment objects of a specific class. From this, we can derive 3-D point clouds from each object that is identified in stereo imagery.

Firstly, a set of stereo image inputs is registered for YOLO. These images will preferably contain multiple objects of the same class, as a single YOLO sweep will segment multiple objects of this class at once and streamline the classification process. For each classification derived from the stereo images, separate sub-images of the identified object are generated using segmentation masks rather than bounding boxes.

Secondly, we manually load each sub-image matching into both SIFT and AKAZE keypoint detectors. SIFT (Scale Invariant-Feature Transform) and AKAZE (Accelerated-KAZE) find a large quantity of keypoints in both sub-images. We then compare the keypoint matches using our selector algorithm, which selects AKAZE or SIFT based on the quantity and quality of descriptor distances. The Lowe ratio test is employed to examine the quality of keypoint matchings between the stereo sub-images. Since the sub-images we use are crops of the original stereo images, the intrinsic parameters will still apply here, though we may need to adjust the epipoles. The segmentation masks allow keypoints to be drawn only on the objects themselves, eliminating the need for standard discernment to rule out outlier keypoints. Utilizing keypoints also allows for easy sub-image pairing as the same distinct pixel sections are often found in both stereo sub-images (Barring significant occlusion or low image quality).

Thirdly, we derive at minimum 8 correspondence points between each sub-image from the keypoints and generate 3-D point clouds based on these correspondences. This is repeated with each sub-image matching.

Deriving classified sub-image pairings allows us to extract relevant keypoints primarily from key objects in the stereo images. It is essentially a 2-step keypoint identification process that can be utilized for other tasks. The first step is to analyze each segmented object descriptor set and find a suitable match from the corresponding sub-image object segmentations (through keypoint matching). The second step analyzes each object matching for keypoint matching to create 3-D point clouds. Using object segmentation masks instead of bounding boxes ensures that keypoints are accurately placed on the objects, automating multi-point cloud generation.

Descriptor matching and object segmentation are the two central tools to automating a traditionally manual process.

2 Methodology

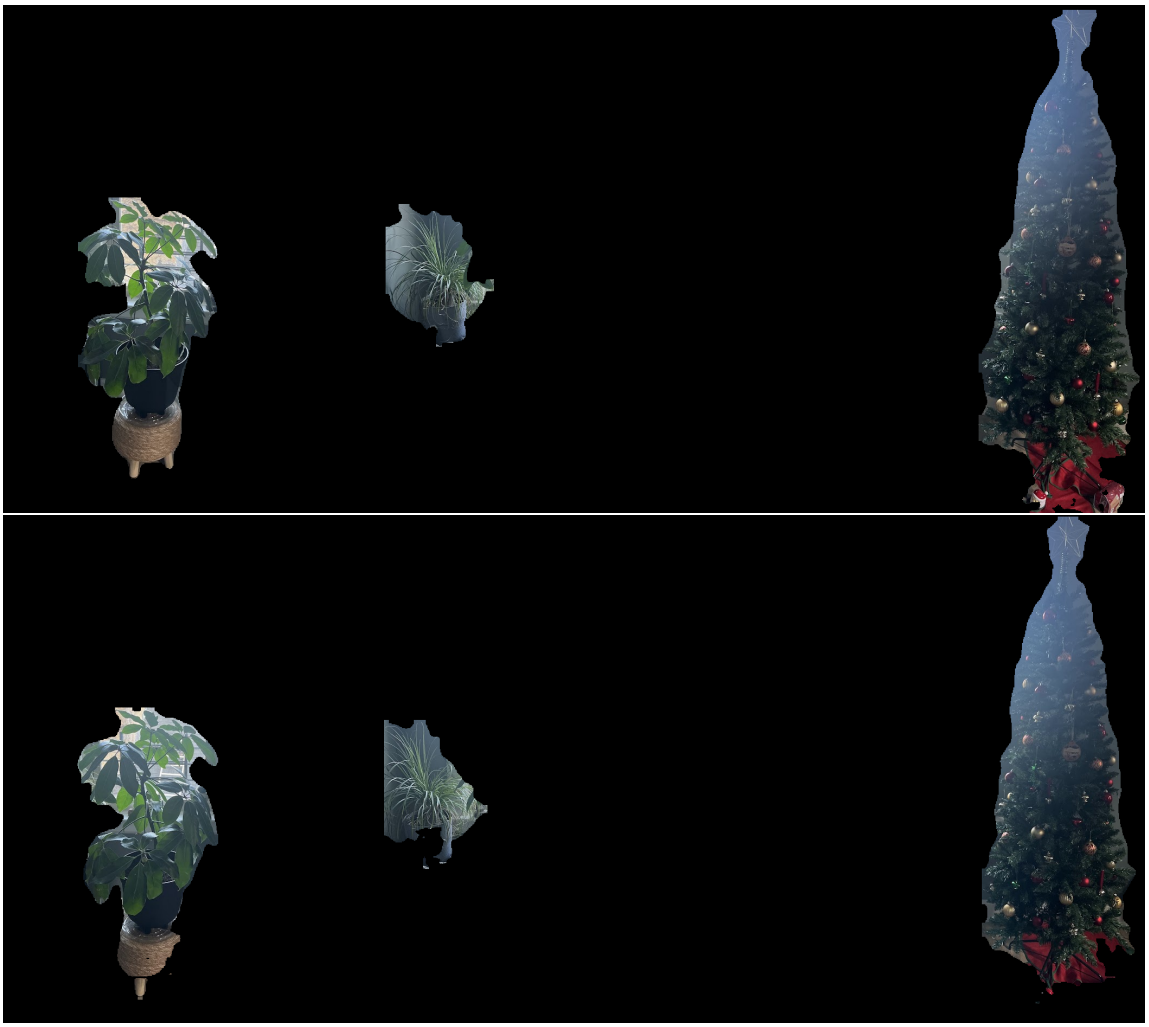
2.1 You Only Look Once (YOLO)

To perform object detection and segmentation, we utilize the YOLOv9 segmentation model (from ultralytics import YOLO). The model is loaded with the YOLOv9 Segmentation Model.

Object detection and segmentation are then performed. The process involves simply passing the input stereo images to the model, which returns results containing detected objects, their confidence scores, and corresponding masks. Masks are saved as images if the confidence score exceeds a specified threshold. This step ensures that only significant detections are considered.

Once the masks are obtained, they are applied to the original image to highlight the segmented areas. Each mask image matches the dimensions of the original image to ensure alignment. Separate images are generated where only the masked areas are visible. Notably, these separate sub-images are also the same resolution as the original image. In fact, all images we work with will be of the same resolution, intrinsic/extrinsic parameters and epipolar geometry, streamlining the pre-processing step. A lot of BFMatcher functionality requires this exact resolution matching. These images are saved

in a specified directory to be processed individually in the next steps.



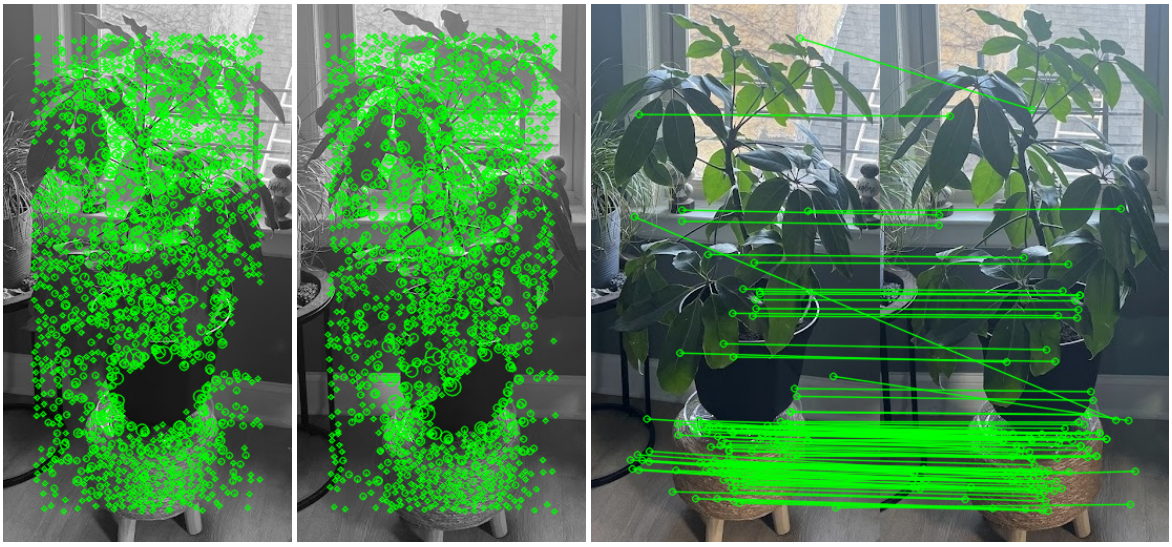
Note that the above images are 3 images separated and blacked out to isolate each object (In these cases, plants). This represents all potted plants found in both stereo images.

If needed, segmentation masks can be saved as separate image files. This involves saving each binary mask as an individual image, which can be useful for further analysis or processing.

2.2 YOLO Pre-processing for SIFT/AKAZE

Originally, I wanted to create sub-images based off bounding boxes of the objects found. This presented multiple issues that are solved instead by object segmentation. Firstly, objects that are distinguished with bounding boxes will still present the region they are contained in for the keypoint detector, allowing erroneous detections. Secondly, a lack of masking does not provide a distinguishing element for keypoint detectors to work. Finally, A lack of cropping around bounding boxes

prevents resolution and sizing issues that may tamper with BFMatcher preprocessing requirements. These issues are illustrated below from when I was originally trying to use YOLOv4.



That being considered, for preprocessing images before applying SIFT and AKAZE, we prefer YOLOv9's image segmentation and masking. These masks help isolate the objects of interest. This is essential for accurate keypoint placement that doesn't mark the background or other impertinent features.

After segmentation, the masks are used to overlay the segmented areas onto black backgrounds while preserving the original image dimensions. Each segmented sub-image is saved separately, ensuring that the original context is maintained. This obviously makes major improvements over the bounding box method.

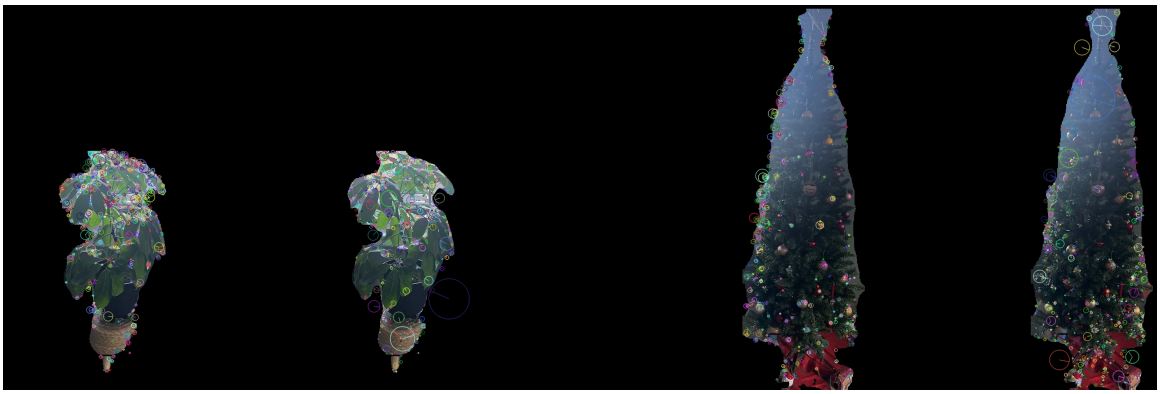
2.3 Scale Invariant-Feature Transform (SIFT) vs. Accelerated-KAZE (AKAZE)

SIFT is particularly popular for automated descriptor locating due to its ability to match keypoints across images under varying scales and rotations. AKAZE keypoint detection, like SIFT, is used for detecting and matching keypoints between stereo images. While both SIFT and AKAZE are designed to handle rotation invariance, AKAZE tends to have a speed advantage due to its use of nonlinear scale spaces.

Both projects share a common goal of extracting relevant keypoints for 3-D reconstruction. In my project, after isolating objects using YOLO, AKAZE keypoints are used to establish correspondences between sub-images, creating detailed 3-D point clouds. Similarly, the OpenGenus SIFT project uses SIFT keypoints to find correspondences that can be used for 3-D modeling and other vision tasks. The main difference is the choice of keypoint detector, with SIFT offering a detail focus, while AKAZE provides a faster alternative.

In my project, we optimize keypoint detection by dynamically choosing between SIFT and AKAZE

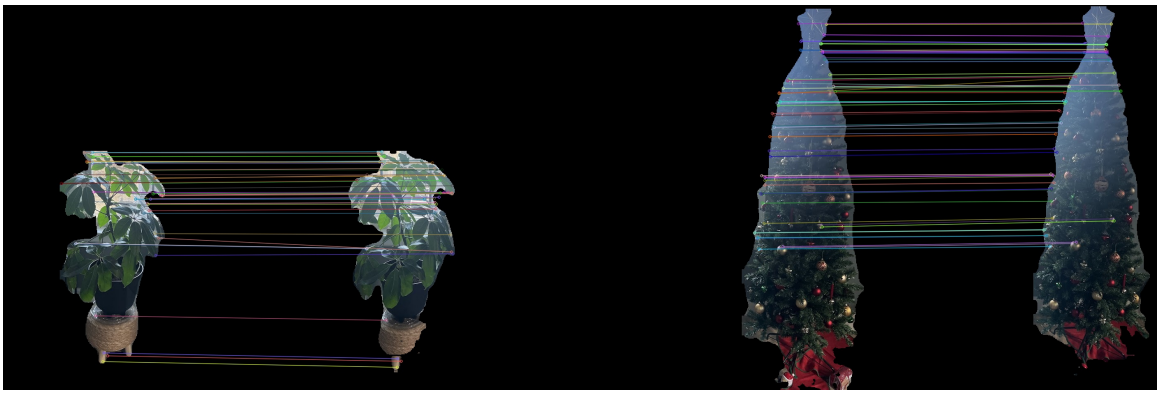
based on an optimizing algorithm. Firstly, both SIFT and AKAZE keypoints and descriptors are detected for a sub-image set. We then perform keypoint matchings on both descriptor sets using BFMatcher. Below is a desriptor detection with AKAZE (left images) and SIFT (right images). These will be matched with their counterpart and a match evaluation will assess both methods.



The quality of the matches is evaluated based on the number of matches and the descriptor distances. Specifically, we count the number of good matches for each method and calculate the average descriptor distance. If AKAZE provides at least 80% of the number of matches that SIFT provides and the average descriptor distance is within 20% of SIFT's average distance, we choose AKAZE for its efficiency. Otherwise, we go with SIFT for its superior accuracy. This decision-making process ensures that we balance speed with high-quality matches. The algorithm also balances this decision against the computational needs of the stereo images we are dealing with. A set that has more matchings, more sub-image pairings, and more pixels will necessitate an efficient method far more than an image with less pixels and less pairings.

So another quick run-down of this section's descriptor algorithm is...

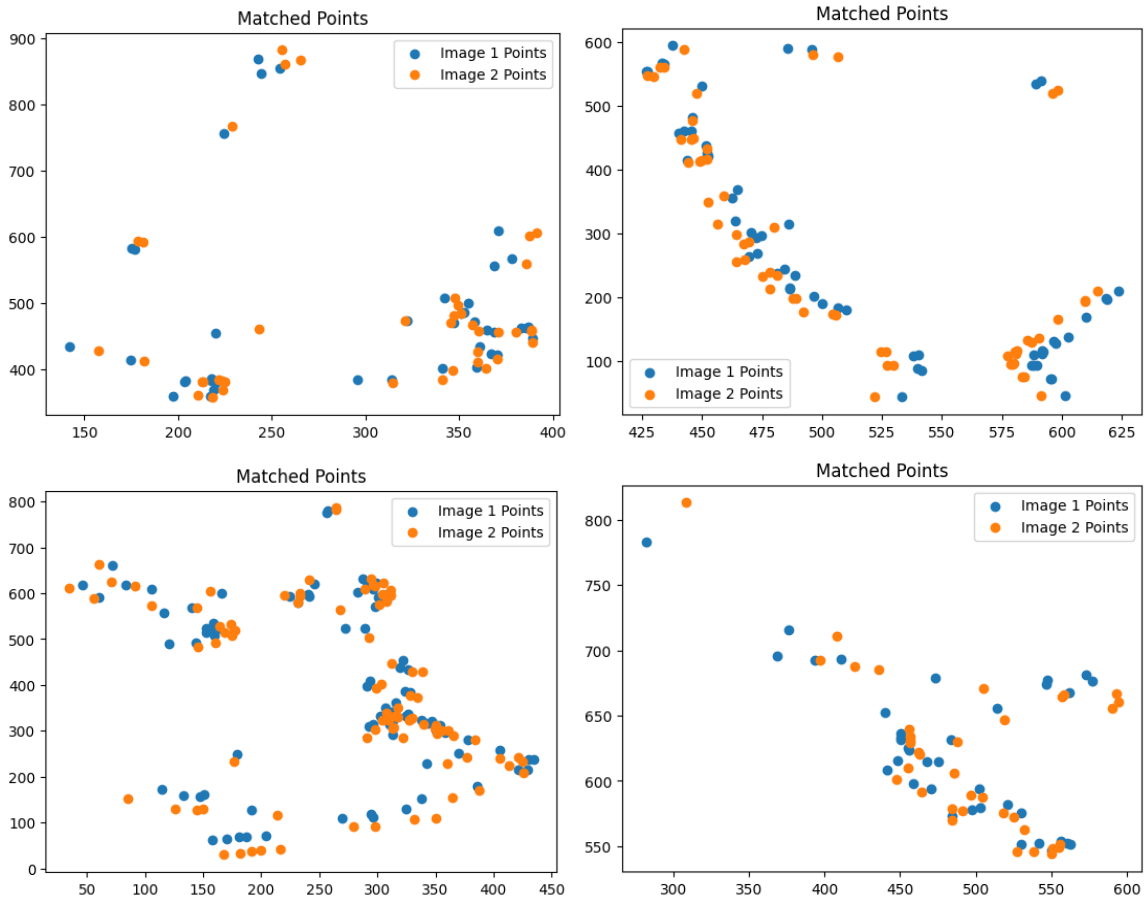
1. Detect keypoints and descriptors using both SIFT and AKAZE.
2. Match keypoints using BFMatcher for both descriptor sets.
3. Evaluate the number of matches and average descriptor distances for both descriptor sets.
4. If AKAZE matches are at least 80% of SIFT matches and the average distance is within 20%, use AKAZE; otherwise, use SIFT. In the below example, AKAZE is selected.



This method has the intent of making the pipeline more adaptive for varying scenarios, as SIFT and AKAZE are known to have different scenarios of expertise. I think this approach beneficially utilizes a decision-making procedure of either SIFT or AKAZE, increasing the pipeline's situation adaptivity.

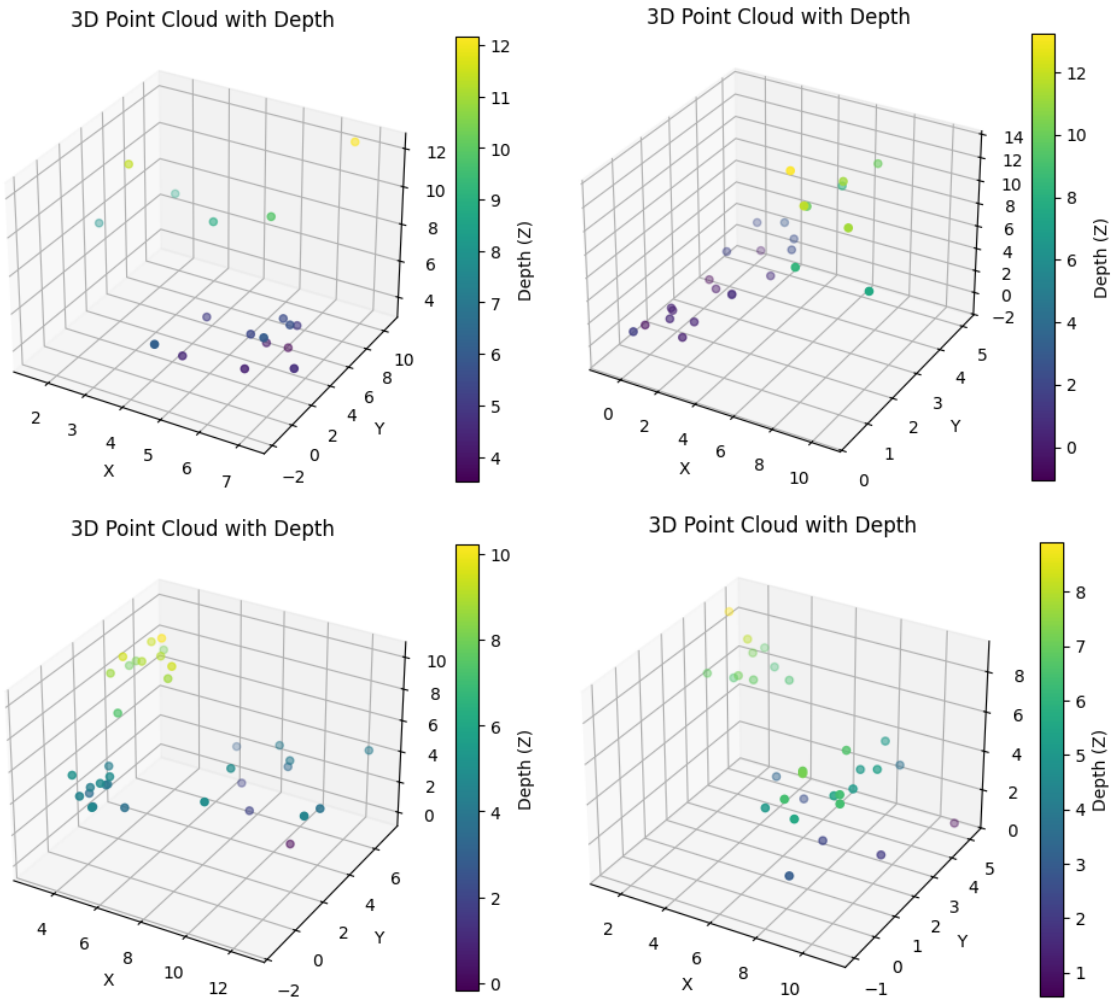
2.4 3D Point Cloud Generation

After applying SIFT and AKAZE for keypoint detection and matching, the next step involves computing the 3D point clouds. The corresponding keypoints between the stereo images are used to establish the epipolar geometry, ensuring that the keypoint matches are geometrically consistent between the two views. To refine the accuracy of the point cloud, only those matches with more than 8 valid correspondences are considered. This threshold helps to eliminate outliers and improve the precision of the final 3D reconstruction. Some examples of 2D plots of the image coordinates in the stereo images are shown below.



Once the correspondence matches are established, the triangulation method is applied to compute the 3D coordinates of each matched point. This step generates a 3D point cloud that represents the

spatial distribution of the objects detected in the stereo imagery. The resulting point cloud is then saved for further processing or analysis. Some examples of the computed 3D clouds are shown below.



I place less emphasis on this step as it's assumed that if we can present a good enough set of n > 8 correspondence points, we have overcome the central task of this pipeline. Ultimately the general goal of this pipeline is to automate the more tedious aspects of point cloud generation and be able to apply it to a scene of objects of a given class and find measurable success. This won't be better than manual matching, but is certainly faster if it's of desirable quality.

3 Related Works

3.1 Kunjeti, C., & Dande, S. (2024). Stereo Camera 3D Map Generation

Chandravaran Kunjeti and Saikumar Dande's project employs a series of techniques to derive 3-D maps from stereo imagery, which closely align with the methods used in my project. Their approach involves capturing stereo images and processing them to generate disparity maps, which are then converted to depth maps. This foundational step is crucial for obtaining accurate depth information, similar to the initial steps in my project where stereo images are used to derive 3-D point clouds.

In their project, Kunjeti and Dande utilize stereo image rectification to align the images properly, ensuring precise disparity calculations. This technique mirrors my use of image rectification to maintain consistent intrinsic parameters across the stereo pairs. Additionally, they employ keypoint detection and matching, although their project focuses on the broader depth mapping, whereas my project integrates YOLO (You Only Look Once) object detection to isolate and process specific objects within the stereo imagery. This allows for the semi-automatic generation of multiple 3-D objects by identifying and separating objects of a specific class, followed by generating 3-D point clouds for each identified object.

Furthermore, both projects leverage keypoint detectors to enhance the accuracy of 3-D reconstruction. In my project, AKAZE and SIFT keypoints are akin to Kunjeti and Dande's use of disparity maps to understand the 3-D structure of the scene. Both projects achieve a 3-D reconstruction, with my project adding an additional layer of object-specific detail through the use of advanced object detection techniques. Both mine and their projects demonstrate the possibility of stereo imaging for various applications, including topography simulation and object detection.

3.2 Tusen-ai, $LiDAR_RCNN$: LiDAR Region-based Convolutional Neural Network

This repository focuses on applying LiDAR (Light Detection and Ranging) technology. LiDAR-RCNN employs a Region Convolutional Neural Network (RCNN) to process LiDAR data, providing high-resolution 3-D environmental information. This method would be used for autonomous driving, robotics, and other applications requiring precise spatial awareness.

LiDAR measures distances to objects using laser pulses, offering high accuracy and dense point cloud data, less dependent on lighting conditions and object textures, making it suitable for dynamic and complex environments. While this differentiates heavily from my pipeline, they can both be leveraged for similar use cases.

My approach with stereo images and AKAZE and SIFT keypoint detection is more accessible and obviously cost-effective, using standard iPhone cameras rather than expensive LiDAR equipment. However, of course, keypoint detection will never capture the same detail as LiDAR.

LiDARRCNN's object detection with stereo imaging techniques could massively enhance the accuracy of 3-D maps. AKAZE and SIFT keypoint detection are useful for deriving 3-D structures from image pairs, but combining it with LiDAR data could eliminate the decision process in place of a more robust sensor.

3.3 Li, D., Xu, Q., Yu, W., & Wang, B. (2019). SRP-AKAZE, Sparse Representation-based AKAZE: Augmented AKAZE for Feature Detection and Matching. IET Computer Vision

Dan Li, Qiannan Xu, Wennian Yu, and Bing Wang present SRP-AKAZE (Sparse Representation-based AKAZE), an augmented approach to AKAZE that combines sparse representation techniques with the AKAZE descriptor. The SRP-AKAZE method enhances the traditional AKAZE with sparse coding to enable the detection and matching of features with higher efficiency in complex environments. The sparse representation aspect helps improve feature matching in scenarios where conventional descriptors like AKAZE may struggle, such as with noisy or low-texture images.

However, SRP-AKAZE does not provide a direct comparative method for optimizing descriptor matching, such as comparing the performance of multiple descriptors or algorithms to find the best fit for a specific task. SRP-AKAZE combines them for improved feature detection and matching in a broader range of scenarios. If I had more time, I would love to implement this and perhaps compare it to SIFT keypoint detection. This would however add additional computation intensity to the project that slightly operated against the original intention I had for this pipeline (That being automation and speed).

Despite the advantages of SRP-AKAZE, I chose to implement the differentiated AKAZE and SIFT for my project. This was more so a creative decision to automate and open the floodgates for AI-based decision making within the pipeline. Additionally, the dual approach of AKAZE and SIFT allows me to optimize each method independently, providing flexibility to adjust the keypoint matching process for the specific needs of my project without the added complexity of understanding and implementing sparse representation techniques.

4 Results

The success rate of 3D point cloud generation using the YOLO-SIFT-AKAZE method was evaluated across several categories of imagery differentiations, with results presented as the number of successfully generated 3D point clouds out of the total number of objects expected in each category.

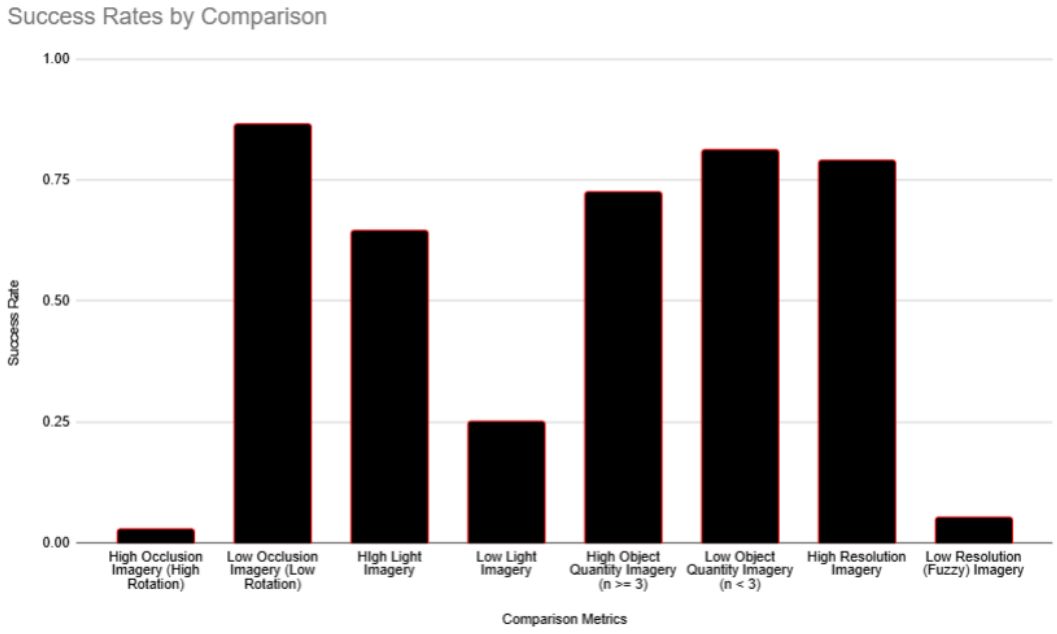
The total number of objects in this case is the number of potted/non-potted plants that were present in all of the stereo images in the data input to the algorithm. The point cloud generation was considered a success for an object if it was detected and matched in both stereo images, had at **least** 8 correspondence points generated between the sub-images that were not clearly wrong (match is not accurate, matches something other than the object, etc.), and it was successfully run through a 3D point cloud generator. The quality of the 3D point cloud is not assesssed here as noise is expected using the correspondence point method, and we just want to be able to verify this method did **something**.
It's important to note that we are essentially comparing the automation pipeline to manual generation. Given the nature of manual 3D point clouds, we are essentially comparing against a 100% success rate. The goal is not to do better than manual generation, but to get as close as possible!

The comparison categories were high occlusion against low occlusion, bright lighting against low lighting (often images with a lot of natural daylight, vs images captured at night), images with high potted plant quantities (typically lower resolution objects as a result) against low potted plant

quantities, and higher resolution against lower resolution. I obtained these results:

Column 1	# 3D point clouds generated	# 3D point clouds expected	Success Rate
High Occlusion Imagery (High Rotation)	1	37	0.02702702703
Low Occlusion Imagery (Low Rotation)	32	37	0.8648648649
HIgh Light Imagery	31	48	0.6458333333
Low Light Imagery	12	48	0.25
High Object Quantity Imagery (n >= 3)	37	51	0.7254901961
Low Object Quantity Imagery (n < 3)	13	16	0.8125
High Resolution Imagery	15	19	0.7894736842
Low Resolution (Fuzzy) Imagery	1	19	0.05263157895

- **High Occlusion Imagery (High Rotation):** 1/37 (2.7%)
- **Low Occlusion Imagery (Low Rotation):** 32/37 (86.5%)
- **High Light Imagery:** 31/48 (64.6%)
- **Low Light Imagery:** 12/48 (25%)
- **High Object Quantity Imagery (n $\geq$ 3):** 37/51 (72.5%)
- **Low Object Quantity Imagery (n ¡ 3):** 13/16 (81.25%)
- **High Resolution Imagery:** 15/19 (78.9%)
- **Low Resolution (Fuzzy) Imagery:** 1/19 (5.3%)



4.1 High vs. Low Occlusion Imagery

The difference between high and low occlusion imagery is stark. High occlusion (and high rotation) imagery yielded only 1 successful 3D point cloud out of 37 objects (2.7%), while low occlusion (low rotation) imagery yielded 32 out of 37 successful 3D reconstructions (86.5%). The sharp contrast is likely due to the challenge that occlusion poses for both keypoint detection (SIFT and AKAZE) and object detection (YOLO). When objects are occluded or rotated significantly, keypoints are either obscured or misaligned, leading to fewer reliable correspondences between stereo images. Additionally, YOLO's object detection performance is negatively impacted by occlusions, as the model may struggle to identify and classify objects that are partially hidden. The low occlusion images, on the other hand, allow for clearer keypoint matching and better object detection, leading to higher success rates.

The stereo images below are a good example of a set that generated 0 point clouds, despite having a fairly high resolution and minimal objects. This is likely because of the low light mode being used, removing a lot of detail from the image while maximizing color information. This is also likely because of the very high occlusion from a high rotation. One of the plants is almost completely obscured, while others take on completely new forms through the large rotation. The result was a partially failed YOLOv9 examination.

```
0: 480x640 3 potted plants, 2 vases, 1346.3ms
Speed: 16.5ms preprocess, 1346.3ms inference, 45.0ms postprocess per image at shape (1, 3, 480, 640)
<class 'torch.Tensor'>
torch.Size([5, 480, 640])

0: 480x640 1 chair, 2 potted plants, 1213.6ms
Speed: 11.0ms preprocess, 1213.6ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)
```

4.2 High vs. Low Light Imagery

The success rate for high light imagery was significantly better (31/48, 64.6%) compared to low light imagery (12/48, 25%). This difference can be attributed to the challenges low light presents for both keypoint detection and object detection. Low light conditions often result in blurry images and diminished contrast, making it difficult for both SIFT and AKAZE to identify distinct keypoints. Furthermore, YOLO's performance can degrade under poor lighting, as the model relies on clear visual cues to detect objects. In contrast, high light conditions enhance the visibility of objects, allowing for more reliable detection and matching.

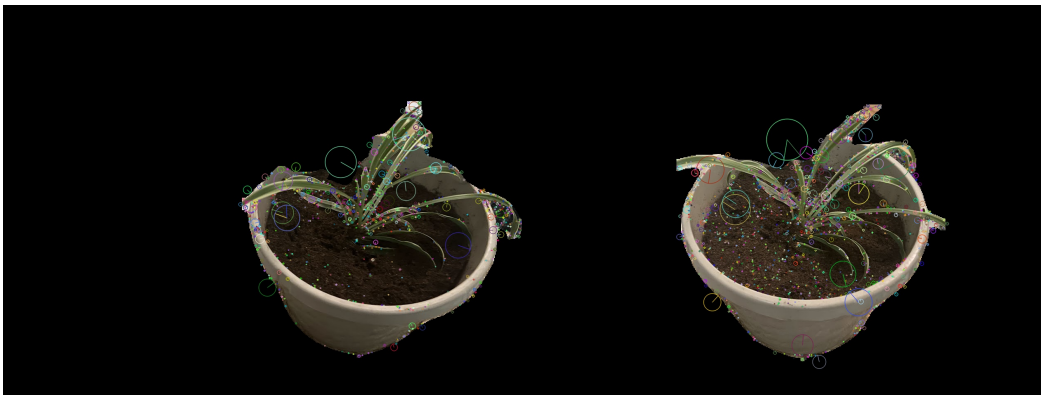
4.3 High vs. Low Object Quantity Imagery

The success rate for high object quantity imagery (37/51, 72.5%) was higher than that for low object quantity imagery (13/16, 81.25%), though the difference is not as pronounced. In scenarios with high object quantities, there is greater potential for occlusion and overlapping objects, which can complicate both the keypoint matching process and YOLO's object detection. However, the higher number of objects in high quantity imagery provides more opportunities for successful reconstructions. For low object quantity images, the relatively sparse number of objects (and possibly fewer occlusions) makes it easier to generate 3D point clouds, but the reduced total number of objects means fewer opportunities to measure success across the dataset.

4.4 High vs. Low Resolution Imagery

The difference between high resolution (15/19, 78.9%) and low resolution fuzzy imagery (1/19, 5.3%) highlights the significant role that image quality plays in the success of 3D point cloud generation. High resolution images provide clearer details, which is crucial for both keypoint detection (SIFT and AKAZE) and object detection (YOLO). In contrast, low resolution images tend to be blurry or have insufficient detail, making it difficult to detect keypoints and perform accurate keypoint matching. The significantly low success rate for low resolution images is likely due to the failure of both keypoint detectors and YOLO to effectively identify and match features.

The below stereo image set is an example of this that passed the YOLOv9 examination and was entered into the SIFT/AKAZE pipeline. However, it failed to generate enough good keypoints due to the image being fuzzy and overly adjusted. The lighting differences are also significant in both, resulting in vastly different descriptor generations. Multiple infractions naturally heightened the failure rate as you can see below.



4.5 Observations

Testing under various conditions and paramters really highlighted conditional success in my pipeline, it is clear that the YOLO-SIFT-AKAZE method works best under conditions where:

- The images have low occlusion and minimal rotation/translation, allowing for good keypoint detections and accurate object segmentation.
- The lighting conditions are favorable, providing high contrast and visibility for keypoint detection and object classification. This one can be less problematic assuming there is enough light for the camera used to capture detail. If the camera is reverting to night mode, that is usually a bad sign.
- The number of objects is moderate, ensuring that occlusion is minimized, and resolution is maximized.
- The image resolution is high, perhaps the most critical element for the accurate key-image/keypoint findings.

The method tends to struggle in scenarios with high occlusion, poor lighting, and low-resolution images. In photos I took that particularly infringed on these categories, the success rate was usually 0. Naturally, in these cases, manual detection or image augmentation may be a necessary alternative. High object quantity scenes, though challenging due to potential occlusion and clutter, can still yield decent results if keypoint/object matching are able to find concise distinctions in both images. Overall, the success of the YOLO-SIFT-AKAZE method is strongly influenced by image quality and scene characteristics, with optimal performance achieved in scenarios with clear, well-lit, high-resolution images with low occlusion and moderate object quantities. However, I would say we succeeded to a measurable extent here!

5 Analysis & Pipeline Assessment

The method of generating 3D point clouds from stereo imagery using SIFT and AKAZE for keypoint detection and matching, combined with YOLO for object detection, provides a flexible and semi-automated approach to 3D reconstruction. This method presents several advantages and limitations, which will be compared to other techniques like LIDAR, GLiDR, and horizontal block matching.

- **YOLO Confidence Thresholding**: The use of a confidence threshold in YOLO plays a crucial role in the number of detected objects. A higher threshold can reduce bad captures, but will also increase the likelihood of misses, especially in complex or occluded stereo images. This trade-off is particularly noticeable when comparing YOLO to methods like LIDAR, which do not require confidence thresholds and can generate more consistent point cloud data in environments with complex textures or clutter. LIDAR's consistency in challenging scenes will obviously be preferable in sub-image identification, and noise from YOLO detection might be more significant than in the original stereo images. Of course, the goal here is automation, not extreme detail.

- **SIFT and AKAZE in Stereo Images**: Both SIFT and AKAZE have shown to be effective. SIFT's general ability to work with low-texture stereo images gave it an occasional advantage over AKAZE. However, with the thresholds applied, I still managed to get AKAZE to be the preferred method more often than SIFT. This would probably argue that the difference between SIFT and AKAZE exists but isn't large when it comes to good keypoint detection. This may also be reflective of the data we are using. Testing involving other object classes would be able to better answer this question.

- **Homography & Cloud Computation**: The accuracy of cloud computation is heavily dependent on the quality of the keypoint matches. Performance is directly tied to the number of reliable keypoint matches, and if fewer than four good matches are found, the homography fails. This is a limitation compared to LIDAR, which directly measures depth and does not depend on keypoint matching for generating 3D models. However, using high-quality stereo images with clear, distinct features can partially resolve this issue and allow for accurate alignment.

- **Object Detection with YOLO**: YOLO is a highly efficient object detection model that classifies and locates multiple objects in a single pass. This efficiency is particularly beneficial when dealing with large scenes containing many objects, as it significantly reduces processing time compared to manual labeling or other object detection methods. However, YOLO's performance massively reduced in crowded scenes with overlapping objects or environments where objects lack distinguishing texture. Variation in quality also came when I experimented with other object classes. In such cases, LIDAR might offer a more reliable alternative for object detection and 3D reconstruction, as it is less sensitive to texture and occlusion and directly measures depth.

- **Automation**: The combination of YOLO with SIFT and AKAZE for keypoint detection enables an automated process for quick object reconstruction from stereo imagery. While SIFT and AKAZE provide valuable keypoint matching, the quality of stereo images remains critical for accurate results. In situations where image quality is suboptimal or significant distortions are present, more advanced techniques can help foster accurate 3D models.

The combination of SIFT, AKAZE, and YOLO **can** offer a potentially efficient 3D point cloud generation, particularly in environments with clear texture and minimal occlusion. This method is well-suited for applications that require rapid, semi-automated object detection and 3D reconstruction from high-quality stereo images. However, in environments with challenging visual conditions—such as low-texture scenes, occlusions, or indistinct objects—this approach may require further refinement or supplementation with other techniques like LIDAR or more advanced keypoint detection methods. Perhaps integrating other methods, such as SRP-AKAZE, LiDAR enhancement, or other novel approaches into the pipeline can significantly enhance the overall performance and reliability of 3D reconstruction.

6 Conclusion

The pipeline developed for 3D point cloud generation using YOLOv9 for object detection, and AKAZE/SIFT for keypoint matching offers a solid jumping point for an automated solution to creating 3D models from scene objects in stereo images. The testing demonstrated a **highly** variant success rate depending on multiple observed conditions.

The success rates are generally higher when the images exhibit low occlusion, good lighting, and high object resolution. The lower success rates are then obviously resulting from challenging conditions such as high occlusion, low resolution, or poor lighting. This highlights the conditional quality of this pipeline to generate reliable 3D point clouds. Perhaps then, you might not be able to retroactively apply this to any and all stereo-images you have. But that does not mean the model completely failed! Under optimal circumstances, this pipeline works extremely well.

Also of note, this method did not struggle as much with tricky objects either, as YOLOv9 segmentation was able to effectively distinguish plants that were highly fern-like or textured.

This pipeline shows great potential in scenarios with clear, high-quality stereo imagery. Further augmentations could be easily integrated and bolster the adaptivity of this algorithm. Other methods to make this more efficient may also be considered. Currently, this program is able to process a single stereo image set in 1-10 seconds, often based off the many factors we discussed above (resolution, object count, etc.). Some ways that we can build off this include:

- **Hybrid Approaches with LiDAR**: The direction I originally wanted to take this involved integrating LiDAR technology with the stereo vision-based method. As seen in the $LiDAR_RCNN$ method before, LiDAR offers higher accuracy due to its ability to directly measure depth with laser technology. Perhaps LiDAR could also be used to train the correspondence point cloud generator!
- **Enhanced Keypoint Detection with SRP-AKAZE**: Another possible improvement is the incorporation of SRP-AKAZE (Augmented AKAZE) for more reliable keypoint matching. While AKAZE and SIFT work well in many cases, SRP-AKAZE could potentially provide better performance by combining the strengths of both methods.
- **Refining Matching Techniques**: Exploring matching techniques beyond BFMatcher such as machine learning-based matching could massively improve correspondence accuracy.
- **Real Time Processing**: Machine learning models can also make the pipeline more adaptable to real time.

In conclusion, the YOLO-SIFT/AKAZE pipeline offers a solid foundation for 3D point cloud generation with plenty of potential for growth. Incorporating complementary technologies such as LiDAR, SRP-AKAZE would enhance the performance and adaptability of the pipeline, making it more applicable to a wider range of real-world scenarios.