EXTRACTING 3D COORDINATES OF OBJECTS IN BUILDING COLLAPSES FROM DRONE IMAGERY

An Undergraduate Research Scholars Thesis

by

KARIM ZAHER

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Robin R. Murphy

May 2022

Major: Computer Engineering – Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Karim Zaher, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Research Faculty Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Extracting 3D Coordinates of Objects in Building Collapses from Drone Imagery

Karim Zaher
Department of Computer Science and Engineering
Texas A&M University

Research Faculty Advisor: Dr. Robin R. Murphy
Department of Computer Science and Engineering
Texas A&M University

When out on a search and rescue mission, it is important to have tools that can easily keep track of the situation that is being handled. Autonomous drones have the ability to quickly collect a batch of images of the scene and its surroundings in order to provide emergency responders with an overview of what they are dealing with. These images are also used to identify hazardous anomalies such as tiny cracks on collapsed buildings. In many cases, however, identifying the exact location of these anomalies may be too difficult, especially when the anomaly is relatively minuscule in size when compared to the structure that it inhabits. The conducted research focuses on developing a system which search and rescue teams may utilize in order to extract the exact coordinates of any point found on an image taken by a drone. In order to do so, a series of images containing the scene of the area of interest is taken from a high altitude. Once that is completed, the images are loaded onto an application called Agisoft Metashape, where the images are combined in order to create a 3-Dimensional model of the location. Finally, the Image Coordinate Point Extraction algorithm, which was created using Metashape's Python API, is run. The algorithm takes in an image as an input, presents it to the

user, and asks the user to click a point on the image to extract its exact coordinate. In the case of this study, the entire process was tested on data from the Surfside Condominium building collapse that occurred in the summer of 2021.

ACKNOWLEDGEMENTS

Contributors

I would like to thank my faculty advisor, Dr. Robin R. Murphy, for her guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my parents and siblings for their patience and love.

The 3D maps analyzed/used for the project were provided by the Florida State University Center for Disaster Risk Policy team.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Undergraduate research was not sponsored nor funded in any way.

1. INTRODUCTION

1.1 Background and Information

In the summer of 2021, emergency responders rushed to Surfside, Florida after news of a building collapse shook the state. The Surfside Condominium collapse was deemed the third worst building collapse in the United States, following the Twin Towers and the Alfred P. Murray Federal Building, yet it seemed as though emergency responders did not have every single tool at their disposal. According to Dr. Robin Murphy, who aided in the search and rescue operations at Surfside, the National Institute of Standards and Technology Construction Safety team wanted a way to get the exact location of a feature simply by clicking on a pixel in images that their drones had captured, however, such an algorithm was not within their possession. Having such a tool would prepare emergency responders for the next deadly building collapse.

This algorithm would work by identifying the region of a 3D digital elevation model that a drone is looking at, then superimposing the image on a digital elevation model to get the exact location of a pixel on the image given a set of parameters from the drone that took the picture to help responders deal with detecting anomalies that may contribute towards a successful mission. In addition, this algorithm would not work solely with images that fit a specific description. And unlike other algorithms, this algorithm would not make rough estimates on the location [1]. Instead, it would give the exact coordinates.

Currently, algorithms assume that the image was taken from a high altitude and that the image is looking at a completely flat surface. This may become an issue when rescuers would like to know the coordinates of an object on the side of a mound of rubble.

By creating this algorithm, assessing a situation would become much simpler and quicker. In the case of the Surfside collapse, images taken by drones could be used to find the exact location of a piece of rubble that may contain a survivor beneath it. Using the coordinates that the algorithm provides, emergency responders could then spend less time figuring out where the piece of rubble is and spend more time on taking action. Meanwhile, in bridge inspections, lasers attached to UAVs, Unmanned Aerial Vehicles, are used to measure the distance between 2 points [2] even though lasers may be unreliable under certain circumstances. Using my algorithm, inspectors could take pictures to get the exact coordinates of each point, then calculate the distance between them.

1.1.1 Drones

A drone is a small, flying vehicle which typically comes with 2 different forms of control. The first type of control is via a remote controller that sends instructions on motion to the drone through radio signals. The second type of control is through automation. With automation, the drone could either be programmed to move to a specified set of locations in its surroundings or on a pre-drawn path.

Nowadays, drones are a key component in rescue missions. Although ground robots and drones are both good alternatives to sending humans to go scout a hazardous or unstable location, drones have additional advantages. Unlike ground robots that have to deal with dynamic obstacles and other unknown factors cluttering the ground, drones have much more freedom in their movement. Their ability to fly allows them to overcome any rugged terrain or the multitude of harsh and treacherous conditions that may be present on the ground. It also allows them to move at a much quicker pace, a factor that heavily weighs in on the outcome of a mission. Additionally, drones may get close to a hazardous or unstable object or location without

risking the chance of making any contact with the location. This prevents human lives from having to be risked to get an assessment of the aforementioned location. Finally, drones are much easier to replace than ground robots. Ground robots are much more expensive to purchase or build, so losing one in a building collapse or a fire is not as ideal as losing a drone.

During the Surfside Condominium collapse, our rescue workers used drones to take images for multiple reasons. First, getting near the building was a safety hazard. Only half of the building had collapsed, and there was fear of the other half coming down eventually. Automated drones were also used to keep a record of the building's condition. At certain times throughout the day and during the span of the entire mission, images of the entire scene and its surroundings were taken using drones. These images were later used to create orthomosaics as well as 3-Dimensional models.

Most cameras that take images from drones have the ability to store information regarding the camera while taking the image. The position – longitude, latitude, and altitude – as well as the orientation – yaw, pitch, and roll – of the camera are common values that are stored in the image. This data is generally referred to as EXIF metadata and will be utilized throughout this research.

1.1.2 Orthomosaic Maps and 3D Models

To understand what an orthomosaic map is, one must first be familiar with orthophotos. When images are taken by drones from a high altitude, objects that are further away from the center of the image become distorted in a phenomenon known as radial displacement. Although it is not as noticeable when dealing with smaller objects, radial displacement makes tall objects appear as though their base and their top are not perfectly aligned or connected by a

perpendicular median. Radial displacement in images can be removed through a process called orthorectification, the process used to create orthophotos.

At its core, an orthomosaic map (Figure 1.1) is an image that is made up of multiple, overlapping orthophotos. The image looks as though it has been taken from an infinitely high altitude. In Figure 1.1, there are void-like spaces throughout the image, most noticeably at the very bottom of the image. These void-like spaces stem from a lack of images taken of that area, typically due to the location being very steep – the void contains a very tall skyscraper.

Figure 1.1: Orthomosaic of Surfside Condominium Collapse

3D map models are generated in a similar fashion. Orthophotos are generated from a large dataset of images, then the overlapping points between them are taken and mapped onto a 3D plane by calculating the location of the points based on where they appear in different images. Finally, points are placed accordingly.

1.2 Problem Overview

With the use of drones becoming more commonplace in the field of rescue robotics, so will orthomosaics and 3-Dimensional models. Ultimately, having a toolbox for these models would allow for an easy, efficient, and insightful analysis. At its base, this toolbox should contain tools that are able to:

- Draw a box around the content of an image that is fed to the 3D Map Model: Looking through the box on 3D Map Model from the position of the camera that took the image should look exactly like the image taken.
- Place and report a coordinate point on the 3D Map Model that reflects the selected point of interest on an image.
- Highlight any anomalies detected in the image on the 3D Map Model: Anomalies include things such as piles of rubble on the ground and cracks on buildings.
- Detect objects on the 3D model: Having some way to detect different objects within image and save their coordinates in a categorical database.

This research focuses on solving the first two points from the list above, however, these two points may simplify, or serve as building blocks for the implementation of the other points.

2. METHODS

2.1 Previous Methods

Previous methods to getting the 3D coordinates of an object in an image involved extremely hands-on approaches. For example, if an image taken by an autonomous drone contained a point that was of interest, a physical visit to the exact location in which the image was taken would be necessary. Once there, a search for the point would have to take place. In some locations, however, the scene may chance rapidly enough to where the location may be unrecognizable in comparison to the image. In that case, finding the point of interest may be difficult, time-consuming, or impossible.

If more than one image containing the point of interest is available, a mathematical approach could likely be taken to calculate the exact coordinates of the point. Although this approach may be less time-consuming that physically visiting the location and searching around, there would be no method of analyzing the point and its surrounding beyond the images on hand.

Our approach aims to handle both efficient point-finding while also allowing for a convenient method to explore the scene without having to physically move to that location. Using our method, we would be able to quickly find the coordinate of one or multiple points and would also have an entire 3D model for location-based analysis.

2.2 Extraction via 3-Dimensional Model

2.2.1 Creating the 3D Model

To begin the process of extracting the coordinates of a point on an image, there are tools which must be created to both simplify the job of the rescue team and to serve as a useful aid in any mission or job. First, a 3D model of the location in which the image was taken must be

created. Having a generated 3D model provides bountiful insight into the location in which the image was taken as well as the location's surroundings. For example, the location may be surrounded by a rough terrain that could not be traversed easily without certain components or tools. In addition, having an 3D model would serve as an assurance that the location of the image provided does in fact match with the coordinate that is returned by looking at the coordinate on the model.

To create a 3D model of a certain location, multiple overlapping images of the location of interest are required. One method is to manually take images from a high-altitude location such as a plane or a crane. Taking a large quantity of images with a consistent quality using a manual method, however, may take a long period of time. Taking a single image may require multiple attempts just to get a stable image. Failure to do so would result in a noisy, final output.

There are also other factors that must be taken into consideration when taking this approach. For instance, the amount of time that it takes between one image and the next image cannot be too far apart. In an outdoor environment, sunlight, clouds, and other weather-based modifiers may alter the quality of the image or shift shadows around, and in a dynamic environment, the landscape may change due to an unstable structure finally collapsing on itself and its surroundings. To combat these issues, planning a photoshoot ahead of time and during times of relative stability may be the best approach. In addition, decreasing the amount of time that it takes to complete a photoshoot may also prove to yield better results. One approach to decrease the photoshoot time is by using a drone. Using a drone with an autonomous-driving or path-planning feature and a camera attached to it removes the need to manually adjust the camera.

While working on the routing of the drone's path, it is important to consider what part of the environment is important to include on the 3D model. If there is a particular location that requires the most amount of focus, it may seem ideal to solely take overlapping images of that area of interest. However, in focusing solely on the area of interest, certain aspects of the surrounding area may be missed out. Part of the edge of the area of interest may get cut out of the 3D model. The location's surroundings may also provide some insightful information regarding the terrain that rescuers are dealing with. For example, if the terrain is rugged, rescuers could prepare vehicles that are able to traverse through the terrain. Therefore, the best traversal path for a drone when taking overlapping images is in a snake-like pattern, starting from a point outside of the region of interest. Along the snake-like path, the drone should stop, take an image of the scene below, then continue. The distance between each stop, or waypoint, should be uniform, and each waypoint's image must overlap with the image of the waypoint taken before and the image of the waypoint taken after the current waypoint. This pattern is visualized in Figure 2.1. The amount of overlap will depend on the region of interest. If the region of interest is homogeneous or bland, more overlap may be necessary.

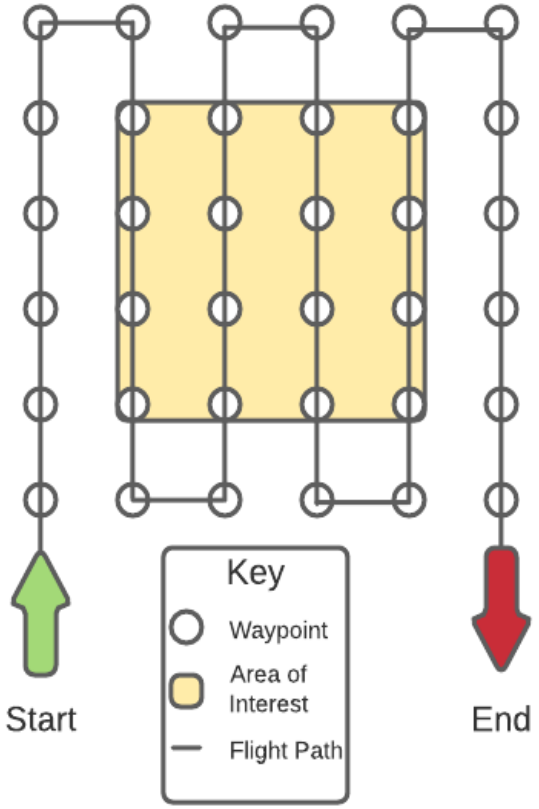

Figure 2.1: Flight pattern of drone during image-taking process

2.2.2 Image Pinpointed Coordinate Extraction Algorithm

Once the Digital Elevation Model is created, the process of coordinate extraction may begin. The Coordinate Extraction Algorithm was written using Agisoft Metashape's Python API. This means that Agisoft Metashape, the software that is used to create the 3D model will also be used to run the Image Pinpointed Coordinate Extraction Algorithm. Specifically, Agisoft Metashape's Professional Edition must be used. Purchasing the software is not necessary, however, since the free demo version of Agisoft Metashape's Professional Edition allows Python scripts to run without any restrictions.

There are 4 core steps in this algorithm:

1. Take input image and point(s) of interest.
2. Insert input image into the 3D model as a Camera object.
3. Perform camera alignment for the 3D model to integrate input image into model.
4. Use aligned input camera to unproject point.

The algorithm begins by prompting the user to select an input image that they would like to analyze. Although the algorithm does not check for the type of the input image, the image must be a raw image captured by a drone. This is a key component since Agisoft Metashape depends on the data contained within the raw image. Raw images taken by drones typically contain some very useful data that is embedded within the image file called EXIF metadata. As previously mentioned, EXIF metadata contains useful information regarding the image as well as the camera that was used to take the image. The information within the EXIF metadata will vary from camera to camera, however, most cameras attached to drones will embed the latitude, longitude, and altitude of the camera when the image is taken as well as the image's pixel width and height. These attributes will be used in the Image Pinpointed Coordinate Extraction Algorithm.

After selecting the input image, a pop-up window containing the image appears (Figure 2.2). From this window, users can click on any location to indicate that they would like to obtain the coordinate of the point.

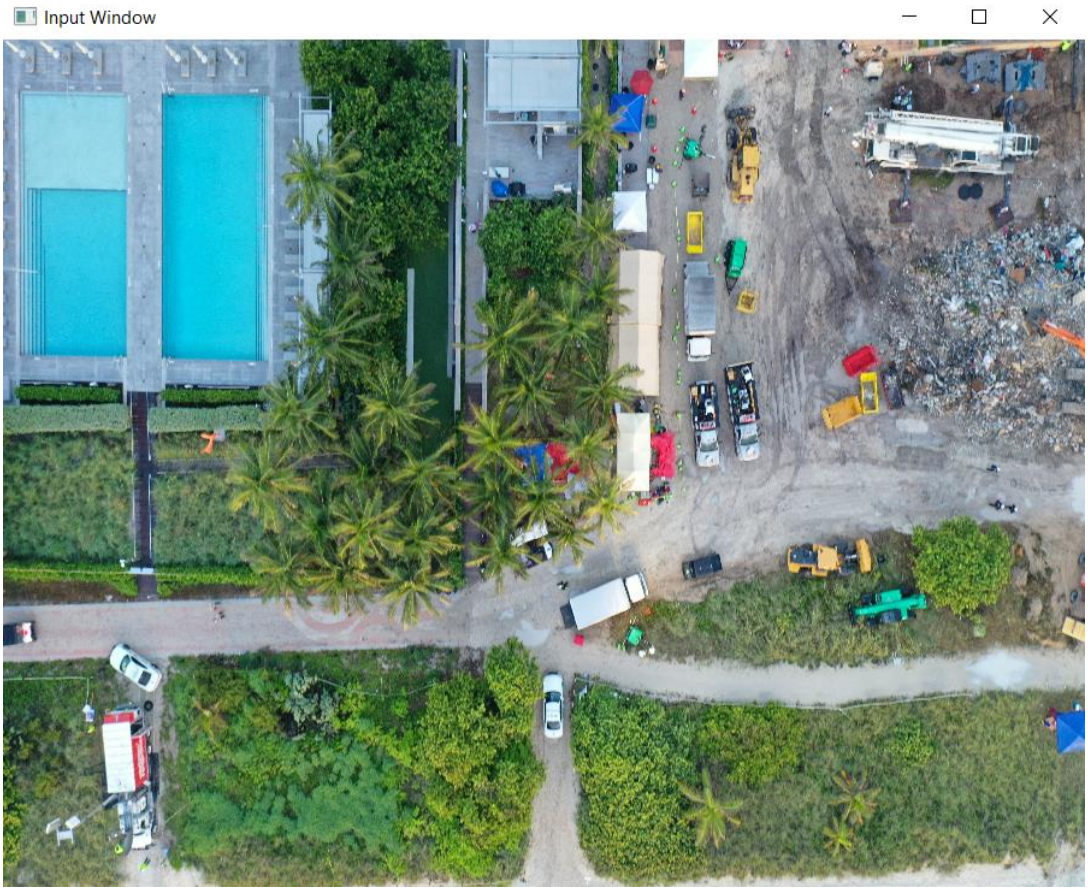

Figure 2.2: Input image window

Upon clicking the point, the algorithm stores the location of the pixel that was clicked. In the case of OpenCV, the Python library used to handle this step, the origin (0,0) of the image is located at the top-left. In addition, a circle appears over the clicked point to indicate that the algorithm has registered the interaction. Users may also click on multiple points of interest. In Figure 2.3, three points of interests were selected – one in the top-left (711,884) represented by the red dot, one in the top-center (3167,729) represented by the green dot, and one in the bottom-right (5411, 2980) represented by the blue dot.

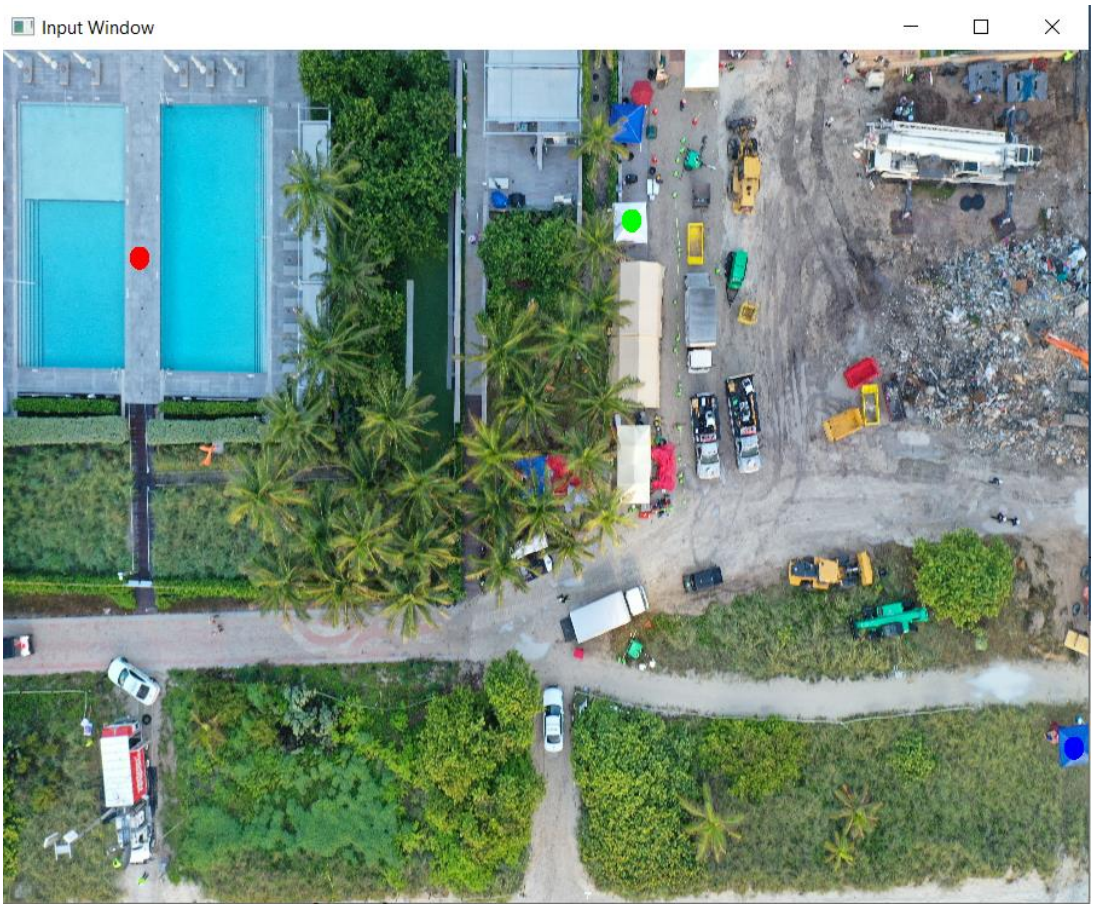

Figure 2.3: Input image window with selected points

After collecting the points of interest, the algorithm now has everything it needs to run. It begins by taking the input image and feeding it to Agisoft Metashape. In doing so, Metashape converts the image into a Camera object and stores it. Once the Camera object has been added to the 3D model, the algorithm runs a camera alignment algorithm that takes the camera and aligns it with the rest of the model. By extracting the image's EXIF metadata to obtain the drone camera's position and orientation and comparing it to the cameras located nearby in the model, the camera becomes integrated into the model which provides the camera's location accuracy relative to the rest of the model. This becomes important when utilizing Agisoft Metashape's API, which is necessary to avoid reinventing the wheel in the next step of the algorithm. Without aligning the camera, the accuracy of the API suffers significantly, as touched upon in the Results

section. Once camera alignment is completed, the algorithm runs the code displayed in Figure 2.4 to find the point of interest as well as the boundaries that contain the image. The unproject function handles the unprojection calculation.

```
def process_camera(chunk, camera, imgPoint):
    sensor = camera.sensor
    corners = list()
    markedPoint = []

    for (x, y) in [[0, 0], [sensor.width, 0], [sensor.width, sensor.height], [0, sensor.height]]:
        ray_origin = camera.unproject(Metashape.Vector([x, y, 0]))
        ray_target = camera.unproject(Metashape.Vector([x, y, 1]))

        corners.append(surface.pickPoint(ray_origin, ray_target))

        if not corners[-1]:
            corners[-1] = chunk.point_cloud.pickPoint(ray_origin, ray_target)

       corners[-1] = chunk.crs.project(T.mulp(corners[-1]))

    for index, val in enumerate(imgPoint):
        point_origin = camera.unproject(Metashape.Vector([val[0], val[1], 0]))
        point_target = camera.unproject(Metashape.Vector([val[0], val[1], 1]))
        point_location = surface.pickPoint(point_origin, point_target)

        point = chunk.addMarker(point_location, True)
        point.label = camera.label + str(index)

    if not all(corners):
        print("Skipping camera " + camera.label)
        return

    if len(corners) == 4:
        shape = chunk.shapes.addShape()
        shape.label = camera.label
        shape.attributes["Photo"] = camera.label
        shape.type = Metashape.Shape.Type.Polygon
        shape.group = footprints
        shape.vertices = corners
        shape.has_z = True
```

Figure 2.4: A Python implementation of the Agisoft Metashape boundary and point drawing

2.2.3 Unprojection

In mathematics, many topics are built upon a foundation of the concept of projection. In simple terms, projection is the ability to project a vector, a multidimensional point that contains both a magnitude and a direction, onto a lower dimension. A common analogy is to think of projection as taking a 3-Dimensional object and shining a light directly above the object. The shadow that this object casts onto the ground is a 2-dimensional projection of the 3-Dimensional object. Unprojection is the concept of taking a lower dimensional object and projecting it onto a higher dimensional plane. Using the previously mentioned analogy as an example, unprojection occurs when the shadow is taken and moved in the direction that it first came from until it hits the object that the projection came from. In doing so, the higher dimensional object is remapped. Likewise, assuming that a Camera object (see Figure 2.5) on Metashape contains the image it represents on its surface, we can select a point on the Camera's "surface", then move in the direction the camera points to until we hit a solid object. In this case, we will hit a part of the 3D model. Figure 2.6 provides a visual to understand the relationship between projecting and unprojecting.

Figure 2.5: A series of Camera objects appear as blue rectangles in Metashape

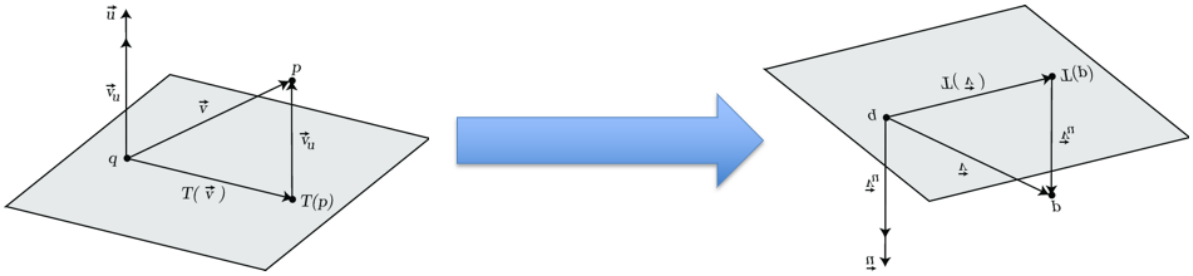

Figure 2.6: Unprojecting from a 2D to a 3D plane

3. RESULTS

3.1 Types of Images Used in Testing

During this research project, three types of images were used for data testing and analysis: nadir images, oblique images, and straight images.

Nadir images are images taken by a drone at a top-down view. These images are captured by flying a drone to a high altitude and pointing the camera straight down. This means that the pitch and roll of the camera are essentially zero and the yaw of the camera is around 270°.

Oblique images are images taken by a drone at an angle. In these images, the roll is set to zero, however, the pitch is set to a slightly larger angle than zero – around 25° – and the yaw can vary.

Finally, straight images are images taken by a drone at a completely flat angle. This means that the camera is parallel to the ground and perpendicular to the scenery or object that it is trying to capture.

3.2 Initial Results of Algorithm

Figure 3.1 shows the input that was used to test a nadir image. In Figure 3.2, we can see the generated bounding box 2D and point of interest 2D0 of Figure 3.1. It is important to note that he names of the point and the bounding box are derived from the image's file name – "2D.jpg", in this case – and hold no significant importance. In addition, adding an additional point of interest would create a new point named 2D1, and so on. Based on the output, it is evident that there is some accuracy problem present.

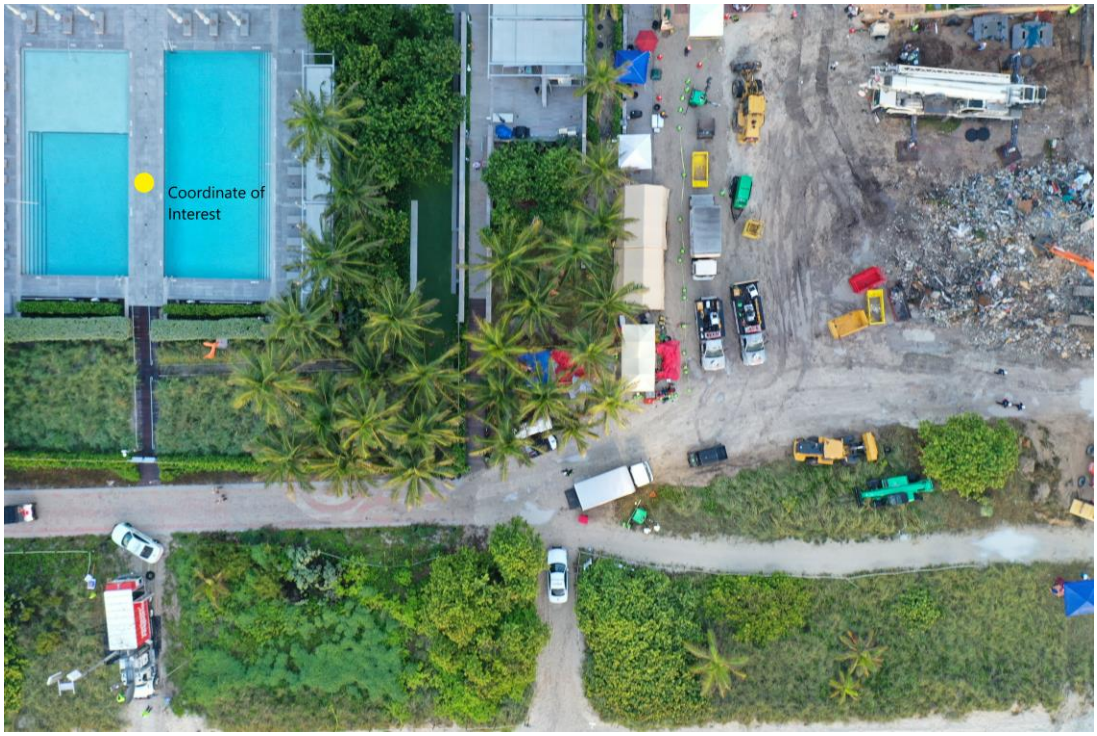

Figure 3.1: Nadir input image and point of interest

Figure 3.2: Output of nadir input

Figure 3.3 shows the input that was used to test an oblique image. In Figure 3.4, we can see the generated bounding box and point of interest of Figure 3.3. Although both the bounding box and the point both seem extremely far off, their percent error is relatively similar to the nadir images.

Figure 3.3: Oblique input image and point of interest

Figure 3.4: Output of oblique input

During the Surfside Condominium building collapse, no straight images were taken of the scene. Therefore, a new map was generated on a different site, and straight images were taken of the scene, as seen in Figure 3.5. Although the predicted output RubbleTest0 and the actual value yellowX appear as far apart as the nadir and oblique images generated and actual points, the output of the straight image is extremely close. The image in Figure 3.6 was taken from a much closer distance than the other two output images due to a bug that caused the 3D model to completely disappear when zoomed out past a certain distance threshold. In addition, the drawn boundary in Figure 3.6 may appear awkward at the top-right corner, however, unprojecting the top-right corner of the image in Figure 3.5 onto the 3D model accurately pushes the boundary location to a relatively far location.

Figure 3.5: Straight image input and point of interest

Figure 3.6: Output of straight input

3.3 Error Handling

While analyzing the entire process of the 3D extraction algorithm in order to determine why the algorithm was not more accurate, I noticed that the alignment phase of the algorithm was not completely accurate in aligning the camera objects of interest. I concluded that my recreation of Agisoft Metashape's alignment process through their Python API was not functioning correctly and that I needed to take a different approach. Rather than aligning the cameras through Agisoft Metashape's Python API, I decided to align the camera through Metashape's GUI, or Graphical User Interface. To do so, the image, or images, of interest must first be added into the 3D model as demonstrated in Figure 3.7. Once the photos are added, camera alignment can begin.

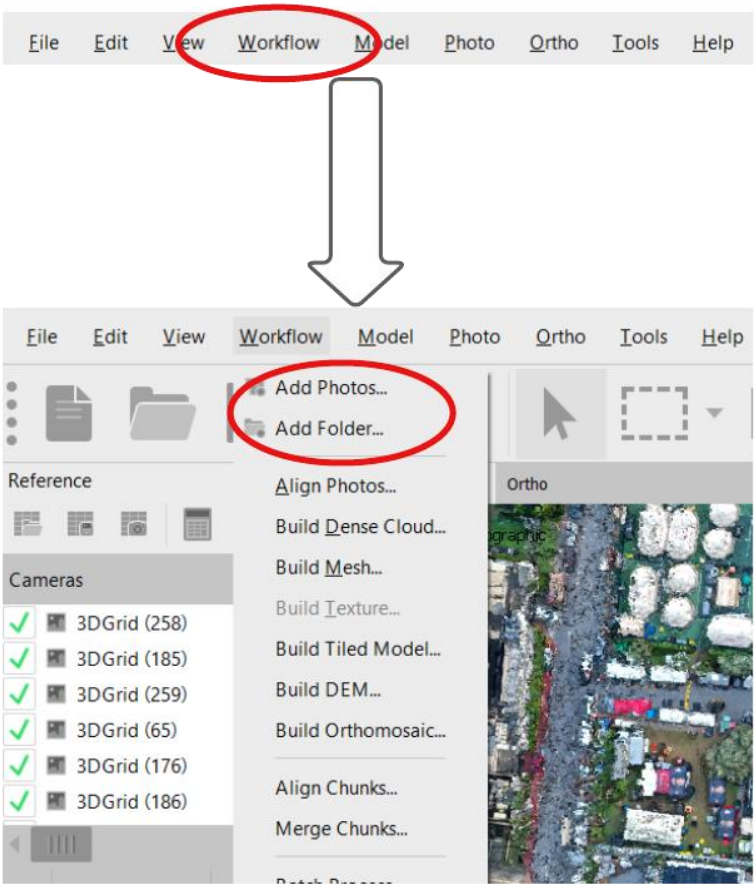

Figure 3.7: Adding images to 3D model

Figure 3.8 shows the process of running camera alignment through Metashape's GUI. Once alignment has been completed, running the unproject portion of the algorithm yields a much lower error result as demonstrated in Figure 3.9 and Figure 3.10.

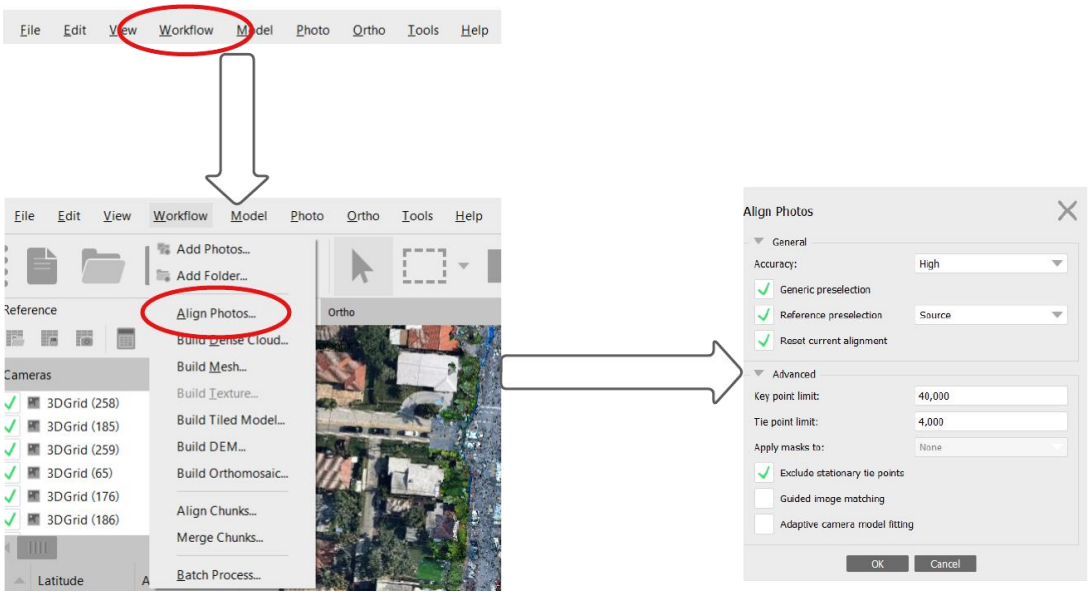

Figure 3.8: Running camera alignment through Agisoft Metashape GUI

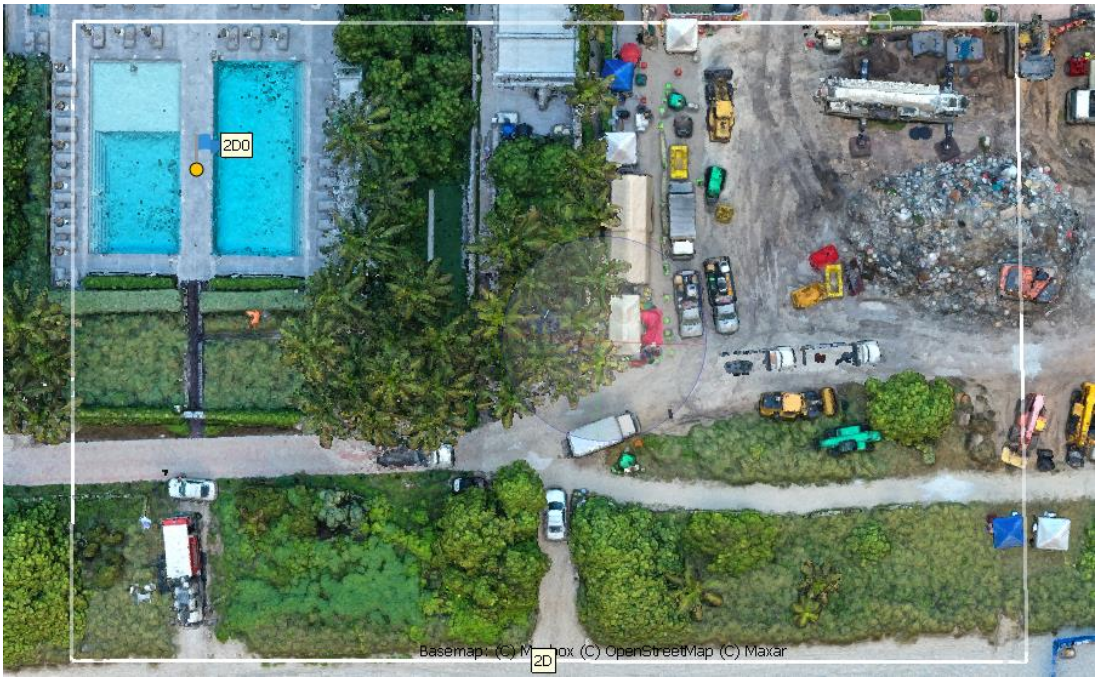

Figure 3.9: Output of Figure 3.1 after GUI alignment

Figure 3.10: Output of Figure 3.4 after GUI alignment

3.4 Results and Comparisons

Table 3.1 shows the actual coordinates of the point of interest, the output coordinate before the alignment issue was caught, and the output coordinate after the alignment issue was fixed. Their respective percent errors are also present. The latitude and longitude errors were calculated by taking the values after the second decimal places, coordinate values that are much more sensitive to slight movements. Then, the difference between the modified true and outputted values is taken, and then divided by the modified true value. In the case of percent error of the latitude before alignment fix in Table 3.1, for example, the percent error between the true value 2228 and the outputted value 2241 was calculated. Altitude percent error was calculated in a normal fashion. Comparing the percent error before and after alignment fix, there is a clear and significant improvement in the accuracy of the algorithm.

Table 3.1: Statistics of nadir image

	Latitude	Longitude	Altitude
Actual Coordinate	25.872228	-80.120331	3.9624
Output Coordinate before Alignment Fix	25.872241	-80.120416	3.998
Percent Error before Alignment Fix	0.5835%	25.6798%	0.8984%
Output Coordinate after Alignment Fix	25.872217	-80.120341	3.998
Percent Error after Alignment Fix	0.4937%	3.0211%	0.8984%

Table 3.2 shows the actual coordinates of the point of interest, the output coordinate before the alignment issue was caught, and the output coordinate after the alignment issue was fixed. Their respective percent errors are also present. The percent error was calculated in the exact manner as in Table 3.1.

Table 3.2: Statistics of oblique image

	Latitude	Longitude	Altitude
Actual Coordinate	25.87309	80.121460	2.4384
Output Coordinate before Alignment Fix	25.872811	80.121229	2.6249
Percent Error before Alignment Fix	9.0291%	15.8219%	7.6485%
Output Coordinate after Alignment Fix	25.87308	-80.121486	2.525849
Percent Error after Alignment Fix	0.3236%	1.7808%	3.5863%

Table 3.3 shows the actual coordinates of the point of interest, the output coordinate before the alignment issue was caught, and the percent error. The percent error was calculated in the exact manner as in Table 3.1. Unlike the nadir and oblique images, the straight image was unaffected by the alignment issue, however, when attempting to align the straight image to the 3D model using the GUI, Metashape would repeatedly return an error that prevented the image from aligning.

Table 3.3: Statistics of straight image

	Latitude	Longitude	Altitude
Actual Coordinate	25.872228	-80.120331	23.7744
Output Coordinate	25.872217	-80.120341	24.1642
Percent Error	0.4937%	3.0211%	1.6396%

4. CONCLUSION

4.1 Implementation and Testing of Algorithm

The goal of this research project was to develop an algorithm that can extract the 3D coordinates of an object using images taken by drones. This was successfully implemented through the utilization of 3D models which can be generated through a large quantity of drone imagery taken of the scene. Once a model is generated, an image of interest as well as a point of interest are selected, and the image of interest is added into the model as a Camera object. From there, camera alignment is performed and an unprojection is performed on the point of interest and the four corners of the image using the 2D pixel coordinates that represent these values. The unprojection finds the 3D coordinate of interest once a solid point on the 3D model is reached.

Tests were performed on three types of drone images: nadir images, oblique images, and straight images. In all three cases, the percent error never exceeded 5%.

4.2 Limitations

Although there was success in creating an algorithm that did exactly what we wanted it to do, it was not done in the most optimized manner. For example, when I initially got the idea of attempting to align the Camera object that was generated by the image of interest, I was hoping that I could simply align that single Camera object with a couple of its neighbors. Doing so would likely have decreased the wait time of the alignment process. In the case of the Surfside Condominium building collapse data, camera alignment took over ten minutes for all 379 images to get realigned as opposed to the smaller subset of images that actually have an effect on the image of interest. Simply focusing on the small subset of images would have likely drastically decreased the wait time to a little over a minute or so.

4.3 Next Steps

In Section 1.2, four main tools necessary to improve 3D model utilization were listed. Of those four main tools, the first two were successfully implemented through this research project. The next step would be to implement the remaining two tools:

- Highlight any anomalies detected in the image on the 3D Map Model: Anomalies include things such as piles of rubble on the ground and cracks on buildings.
- Detect objects on the 3D model: Having some way to detect different objects within image and save their coordinates in a categorical database.

These tools would build on top of this research, using unprojection to mark the location of the points of interest. Upon completion of these two goals, 3D models would likely begin to play a much larger role in situations where an analysis of a location may be necessary, especially in building collapses.

REFERENCES

[1] H. Peel, S. Luo, A. G. Cohn, and R. Fuentes, “Localisation of a mobile robot for bridge bearing inspection,” vol. 94, pp. 244–256, 2018, doi: 10.1016/j.autcon.2018.07.003.

[2] X. Peng, X. Zhong, C. Zhao, A. Chen, and T. Zhang, “A UAV-based machine vision method for bridge crack recognition and width quantification through hybrid feature learning,” vol. 299, p. 123896, 2021, doi: 10.1016/j.conbuildmat.2021.123896.