UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

AUTOMATIC ANNOTATION OF 3D POINT CLOUDS FROM 2D IMAGES

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Pablo Hernández Lázaro

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ENGINEERING

Advisor: Javier Ruiz Hidalgo

Barcelona, October 2018

Abstract

Generation and processing of 3D models for real life objects is playing a major role for today’s computer vision applications. From autonomous cars to surveillance drones, passing through AR; the ability to understand and model what’s around us is being key to success.

There has also been a tremendous progress in the field of machine learning on increasing the accuracy for 2D image object recognition.

In this context, the project aims to expand and further investigate the capabilities of 3D models, and be able to extrapolate previously generated label information from 2D sources to the 3D space without human interaction.

Resumen

La generación y procesado de modelos 3D para objetos de la vida real está jugando un importante papel en la actuales aplicaciones de visión por computación. Desde coches autónomos hasta drones de reconocimiento, pasando por Realidad Aumentada; la habilidad para entender y modelar lo que nos rodea está siendo la clave para el éxito.

También ha habido un enorme progreso en el campo del machine learning alcanzando cotas cada vez más precisas en referencia al reconocimiento de objetos en imágenes 2D.

En este contexto, este proyecto quiere expandir e investigar las posibilidades de los modelos 3D, y ser capaz de extrapolar etiquetas previamente generadas desde fuentes 2D al espacio tridimensional sin la interacción humana.

Resum

La generació i processament de models tridimensionals de la vida real està tenint un molt important en l'àmbit de les aplicacions de visió per computació actuals. Des de cotxes autònoms fins a drons de reconeixement, passant per la realitat augmentada; l'habilitat per a entendre I modelar el que ens envolta està essent la clau per a l' èxit.

També hi ha hagut un important progrés en el camp de machine learning, arribant a altes cotes de precisió en el reconeixement d'objectes en imatges 2D.

És en aquest context que aquest projecte vol engrandir I investigar les possibilitats dels models 3D, I ser capaç d'extrapolar etiquetes prèviament generades des de fonts 2D cap a l'espai tridimensional sense la interacció humana.

Acknowledgements

I would like to begin by thanking the thesis’ supervisor Javier Ruiz Hidalgo, for his very helpful guidance through the whole duration of the project.

I am also thankful for the access granted to the Fuji dataset, acquired by Grup de Recerca en Agròtica i Agricultura de Precisió, Departament d'enginyeria agroforestal, Universitat de Lleida whithin the AgVANCE (AGL2013-48297-C2-2-R) project with the support of the Ministry of Economy and Competitiveness.

Revision history and approval record

Revision	Date	Purpose
0	21/09/2018	Document creation
1	01/10/2018	Document revision
2	08/10/2018	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Pablo Hernández Lázaro	herlazp@gmail.com
Javier Ruiz Hidalgo	j.ruiz@upc.edu

Written by:		Reviewed and approved by:	
Date	08/10/2018	Date	01/10/2018
Name	Pablo Hernández Lázaro	Name	Javier Ruiz Hidalgo
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

Although the digitalization of 3D real life objects is becoming more popular with new civil and military applications, the complexity of data acquisition and need of power consumption to process the resulting data constitutes a major withdraw for the development of more community-oriented solutions.

The object of the research is to provide a labelled 3D model from a 2D previously labelled dataset. Such solution could provide a response to 3D modelling without the need of complex systems for input data, nor the computing needs that come with processing 3D data, but taking advantage of using the 2D source and its additional data.

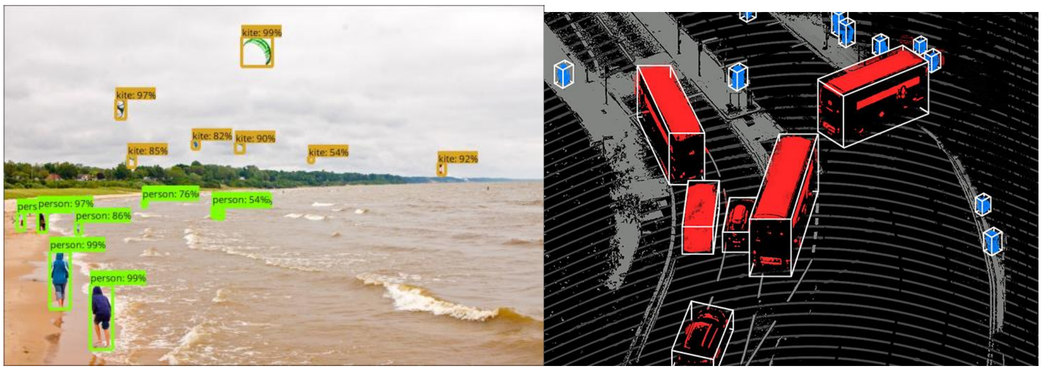


Figure 1. *2D annotation [22] (left) vs 3D annotation [23] (right)*

This project aims to complement the 3D object acquisition and modelling with more advanced technologies of the 2D image world. Moreover, machine learning is having a major impact in the object recognition field, being able to achieve impressing accuracy and efficiency results. These improvements have been mainly been done for 2D imaging and video, and although the same models could be modified for 3d objects input data, this work follows the path of taking advantage of the new and promising advances to enrich the 3D object reconstruction world.

1.1. Statement of purpose

The main goals of the project are:

- Reconstruct a 3D model from the 2D dataset. Ensure a good reconstruction pipeline for the input data.
- Transform and attach the 2D label information to the 3D space. Implement a solution integrated in the reconstruction process.

1.2. Requirements and specifications

The project aims to generate a tool to automatically annotate 3D point clouds from labelled 2D images. In order for this to be fullfiled, we establish some requirements and specifications.

The system must be able to process multiple input data, we expect to have hundreds of inputs, scalability is a must.

The output of the system must be in a known format so that it can be easily made use of for external 3D tools.

Resolution of the object must be enough to identify labelled objects in the structure.

Some error is expected when labelling the 3D object since the same point viewed from different cameras can be attached to different labels.

1.3. Project Background

This project is proposed by the supervisor, Prof. J. Ruiz of the Image Processing Group, and is placed in a context of a collaboration with Universitat de Lleida (UdL). This work is neither the start or end of it, but plays a role in the possibility to go forward with the major project.

Because of this situation, we are provided from data of previous stages, 2D images with manual annotation and 3D Point Clouds. We also are meant to produce results that fall in line and have sense inside the scope of the collaboration, meaning that next steps should be able to handle the new generated data as well.

1.4. Work plan with tasks and a Gantt diagram

The work plan is structured in 6 blocks or work packages, each with a list of related tasks assigned.

Work Package 1. Research

WP1 Tasks:

- 1.1. Reconstruction algorithms research
- 1.2. Reconstruction software research

Work Package 2. Local environment configuration

WP2 Tasks:

- 2.1. Configure colmap build and dependencies
- 2.2. Configure IDE with colmap
- 2.3. Extending colmap configuration

Work Package 3. 3D Reconstruction

WP3 Tasks:

- 3.1. Process Input images
- 3.2. SIFM implementation
- 3.3. Bundle Adjustment implementation and tuning
- 3.4. Dense reconstruction implementation and tuning

Work Package 4. Annotation

WP4 Tasks:

- 3.1. Process Input labels.
- 3.2. Attach annotations to point cloud reconstruction.

Work Package 5. Experimental

WP5 Tasks:

- 5.1. Validate system output

Work Package 6. Writing

WP6 Tasks:

- 6.1a. Project Proposal Draft
- 6.1b. Project Proposal Final Document
- 6.2a. Project Critical Review Draft
- 6.2b. Project Critical Review Final Document
- 6.3a. Final Report Draft
- 6.3b. Final Report Final Document

1.4.1. Gantt diagram

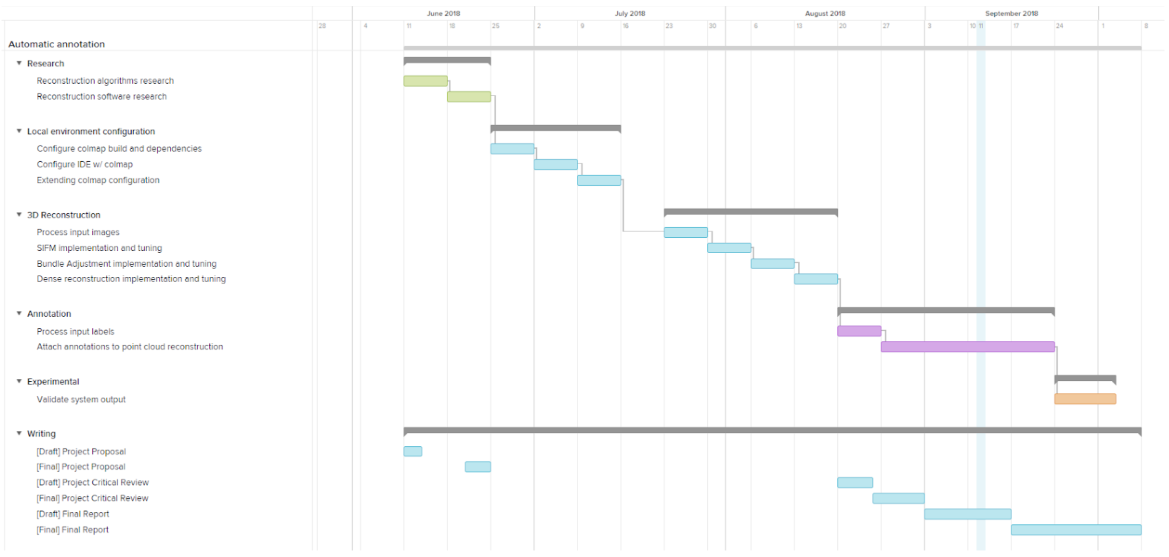


Figure 2. *Gantt diagram of the thesis*

1.5. Incidences

Incidences causing delays in the project have resulted in having to readjust the objectives of the project.

Local environment development was chosen in the initial stages for convenience. Issues configuring builds, dependencies and different version compatibility have been common and have taken longer than expected, causing a bottleneck and delaying the project.

Chosing to integrate our solution in an already operating framework has translated to a good amount of time understanding it and identifying the correct steps to make for building an operating solution.

2. State of the art of the technology used or applied in this thesis:

3D reconstruction from 2D images has been an active area of research for past years. An analysis of the state of the art will be done from two perspectives, those being: 1. Fundamentals and 2.Software solutions.

Since annotation tools are not a common characteristic in 3D reconstruction solutions, an analysis of its situation will also be done in its own section.

The current state of both is analysed in separate ways.

2.1. Fundamentals

2.1.1. 3D Reconstruction from 2D

A 3D object can be represented as a set of points in a 3D space, or a Point Cloud. In order to understand how to get the positions of the 3D points from multiple 2D images, some related fundamentals are exposed.

2.1.1.1. Structure from Motion

Structure from Motion works on the basis that a 3D object is reconstructed by multiple overlapping 2D images, taken by motion of the camera resulting in different perspectives of the same object. Other approaches to the reconstruction issue may require original coordinate information of the camera position or even the 3D position of the objects in the image, Structure from Motion (SfM) solves this information requirement by automatic feature detection and matching between multiple input images. This makes the reconstruction live in a relative coordinate-system instead of an absolute one, which should be taken into account if scale and orientation are required.

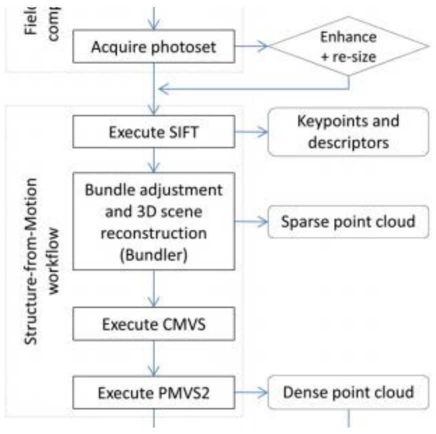


Figure 3. *Structure from Motion workflow.* [18]

Feature detection and matching is a key process that is achieved using the Scale Invariant Feature Transform (SIFT). By analyzing variance in the image when different scale, rotation and illumination are applied, keypoints are set. The number of keypoints in an image is related to its texture and resolution.

Keypoints in at least 3 images are matched using RANSAC[27] or approximate nearest neighbour in order to be used for the point cloud reconstruction.

The quality of the reconstruction is directly related to the number of keypoints matched between the input images, so as usual, more input images might result in more keypoints featured along multiple pictures thus producing a better reconstruction.

Reconstruction from the keypoints results in a low-density point cloud, so a dense reconstruction block is added using CMVS or PMVS algorithms.

Scale Invariant Feature Transform (SIFT)

The SIFT[1] is used for feature/keypoints extraction and matching. It is able to transform image data into scale-invariant coordinates relative to local features. This makes it a very robust procedure since the keypoints seem to have low variation when exposed to image transformations and distortions. The fact that a large number of features can be extracted without big processing costs makes it the go-to method for this specific area.

The selection of the keypoints candidates goes through different stages at which robustness against scalability, rotations, low contrast among other factors is tested. The result of the process is a set of robust keypoints represented as keypoint descriptors, computed from the gradient and orientation for each image sample.

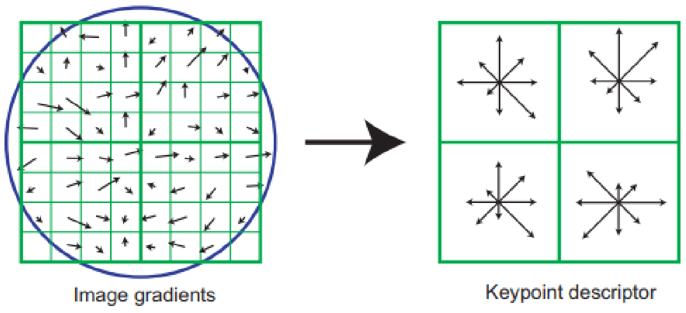


Figure 4. *Keypoint descriptor [1]*

Projections

To be able to translate 2d coordinates to the 3d world and vice versa we need some structures and methodology [20,21]

Image coordinates. (x,y) pixel coordinates.

Camera coordinates. (x,y,z) The camera coordinates are shaped by the intrinsic parameters of the camera (focal length, principal point or other distortion parameters)

World coordinates. (u,v,w) The reconstructed 3D object coordinates

In order to understand how the 3D coordinates from a 2D image are related, the fundamentals of the projection process is analysed.

From World to Image coordinates (forward projection). To go from world to image coordinates, first going from world to camera coordinates is needed. Although camera and world share the same space, they can have some distortion in the shape of translation and rotation. To parametrize this transformation, the following structure is used:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

***Figure 5** . Camera coordinates are achieved by computing the rotation r and the translate c to the world coordinates[21].*

Now, to go from the camera coordinates to the 2D space, we take advantage of the perspective projection. The perspective projection matrix introduces the camera intrinsic parameters to get the 2D coordinates of the image.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f/s_x & 0 & o_x & 0 \\ 0 & f/s_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

***Figure 6.** Perspective projection matrix[21] .Focal lentgth f, offset o and effective scales s.*

Coordinates are now achieved by solving the matrix equation, where x = x'/z' and y= y'/z'.

2.2. Software solutions

State of the art solutions to the exposed subjects is analysed in this section.

2.2.1. 3D reconstruction from 2D

The analysis aims to expose possible software solutions for 3D reconstruction from 2D images in the context of our work's development. The project does not focus on the reconstruction itself but on the automatic annotation of the reconstruction, that's why we need a solution for this block without compromising previous or next steps.

Because of the number of solutions and the different needs each can cover, we specify a set of requirements:

1. **Multiple image input.** To match our dataset characteristics, we shouldn't have a problem with scalability of input images.
2. **Structure from Motion support.** State of the art technology to reconstruct a 3D object from 2D images.
    a. Feature matching (SIFT)
    b. Bundle adjustment
3. **Dense reconstruction support (CMVS/PMVS).** Needed to get higher density rates for the Point Cloud.
4. **Active community.** Since integration with the library might be not easy, an active community being able to support and find bugs is key.
5. **Open Source licensing.** For the solution to have a clear future, an open source license makes it more attractive.

VisualSFM[10]. Developed by Changchang Wu， it is shaped as a GUI integrating SfM solutions and dense reconstruction support. Proven good results and easy to use. Hard to customize the output, so a custom system integration might not be able to succeed. Although it looks great for getting a reconstructed output, the inability of adding extra features fitting our purpose makes it hard to choose.

Bundler[11]. Developed by the Computer Science department of the University of Cornell, this library provides SfM support from multiple images. Although dense reconstruction is not directly supported, the same project incorporates a tool to shape the data so that it can be used with external tools for this purpose. GNU licensed with a community at github of 9 contributors and 111 commits. Last commit dating from October '17.

Photoscan[12]. Developed by AgiSoft, it is a closed software solution for reconstruction of 3D objects and spaces. Although it features ease of use and good accurate reconstructions, it is a licensed product that does not provide a way to customize the output.

Colmap[13]. Colmap is a library written by Johannes L. Schönberger, with funding from ETH Zurich and UNC Chapel Hill under BSD licensing (similar to MIT licensing). Among its functionalities are a SfM pipeline in combination with dense reconstruction capabilities. Open sourced, the source code is available at github so there is the possibility to customize the output. A github community of 22 contributors, +1200 commits with a high frequency.

MicMac[14]. Software developed at the French National Geographic Institute and French national school for geographic sciences, opensource. Provides integral solution for 3D reconstructions. Since it is opensource, access to the source code is granted. Github community of 26 contributors and ~8000 commits with a high frequency.

Among the listed solutions, Colmap fits our requirements the most. Not only does it have a good reconstruction pipeline, it also offers easy access to its different stages so that custom solutions can take advantage of the sharing of information. A key point for the chosen library has also been the great documentation Colmap provides compared to other solutions.

2.2.2. Labelling

There exist a number of tools that allow us to manually annotate Point Clouds. In our case, we will not manually annotate the PC but get it automatically from the 2D image labels. We need to create our own structure in order to achieve this.

2.2.2.1. 3D Annotation tools

Because of the complexity of 3D structures, there exist more limited solutions to the issue, none of them are capable of automatically annotate the 3D point clouds from 2D info, but there exist some solutions to manually annotate the object (not our case).

The idea is basically the same, although there are different approaches to the same issue. One of those is to basically attach the label to each point individually, resulting in more accuracy but more resource consuming and also not being able to filter noise or deviation. The usual approach is to group most of the points into a cuboid (same with squares at 2D) and attach the label to the whole area of the object.

Some 3D editors have built-in tools to annotate the object but also some external ones as Rviz Annotation Tool [26], more oriented to acquired point cloud data from lidar or similar, are found.

3. Methodology / project development:

This unit describes the process followed to transfer 2D label information from multiple images into a 3D point cloud reconstructed from these sources.

The solution is built as an extension of the colmap project[13]. Building on top of an already mature and active project, gives us a great opportunity to benefit from an already optimized basis and give value to the open community projects. Although there is a huge positive side to this approach, integrating a solution into a greater one requires to fully understand not only how to connect those but to understand how the system works as a whole unit.

This section will be divided in three parts, the first one to show the datasets used, the second one to explain the colmap system pipeline and the last one to expose the process followed to integrate our subsystem to the general one.

3.1. Datasets

Datasets play a major role in the development of the project. A very large dataset can delay substantially the project in the initial stages, as major changes occur very frequently. Using a small dataset can lead to issues not coming to the surface. A compromise is made by using a medium-sized for the initial stages and then using a bigger one.

Labels will be presented as bounding boxes. Labels can also be seen as masks instead, but the first is chosen due to lower processing needs.

3.1.1. Fuji

This dataset is provided by UdL in the context of the collaboration project with UPC. It shows images of medium sized objects (apple trees) taken from a near distance (1-2m). Images are taken sequentially, following a horizontal line in the ground, and have good overlap, which is good for triangulation.

Size of the images is 5184x3456 px and are divided in two subfolders ("Cara Este" and "Cara Oeste"), one for each side of the tree line. Each group has around 700 to 800 images and a size of around 8 GB each.

Figure 7. *Cara Oeste (left) and Cara Este (right) image folder.*

3.1.2. Truck

This dataset is provided by “Tanks and temples”[4] and contains 251 images from a medium-large object, a truck, taken from 360º. Overlap is not as good, to test if the reconstruction is as required, ground truth is provided to compare with our output result

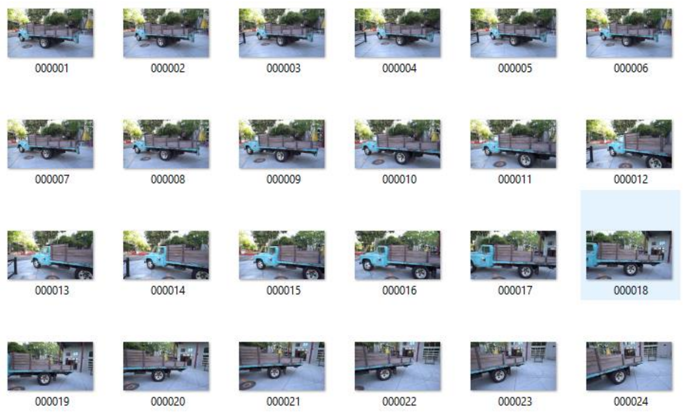


Figure 8. *Truck dataset from Tanks and temples[4]*

Initial to medium stages of the projects will be done using this dataset for timing purposes. Once the pipeline is tested successfully with this one, the Fuji dataset will be used.

3.2. Colmap Pipeline

Colmap provides a pipeline that functions as follows:

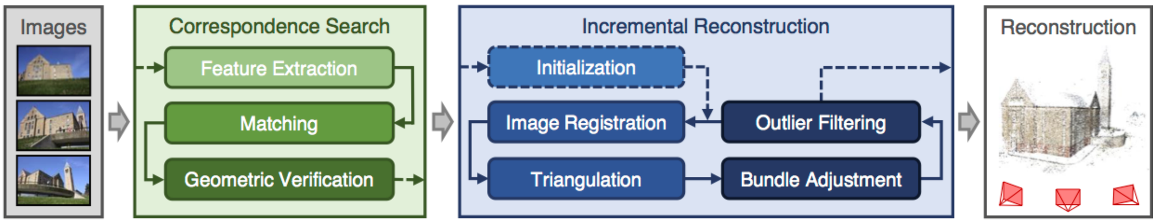


Figure 9. *Incremental Structure from Motion pipeline.*[25]

The two main blocks group the related processes:

3.2.1. Correspondence Search.

This block groups the processes that take the dataset images as an input and generate the necessary information to begin the reconstruction.

Feature Extraction. More relevant pixels of the image are saved as features. The extraction relies on the Scale-Invariant Feature Transform (SIFT) [1] and its derivatives to identify those pixels with more information.

Feature Matching[2]. For each feature in I_x , the most similar feature in I_y is found. An overlap between both images can be computed using the correspondence of the features, if it exists.

Colmap offers different options of matching, depending on the dataset characteristics.

- Exhaustive feature matching. Best reconstruction results, it compares image by image. Problematic with datasets of several hundreds of images.
- Sequential feature matching. As its name indicates, it takes advantage of images taken as a sequence. Relies on names of files order.

Fuji dataset is taken sequentially, so sequential feature matching could be used, but exhaustive is chosen for its better results.

Geometric Verification. This block verifies the feature matching, provided that a feature with a high similarity index with another one might not be representing the same point in the space. The output of this block is a scene-graph [2]

3.2.2. Incremental reconstruction

Taking the correspondence search output as its input. The sections of this block are able to generate a sparse reconstruction of a 3D object.

Incremental Reconstruction Initialization [5] . This process reads the input scene-graphs and selects a suitable two-image pair. The reconstruction of the 3D object begins.

Image Registration[6]. New images can be registered to the model by finding the estimation of the pose of the camera from point correspondence (3D to 2D).

Triangulation[7]. The images of the dataset are now integrated in the initialized 3D model. New points emerge and already featured points in previous images are strengthen. New points can be added to the scene –triangulating- if at least another camera sees it.

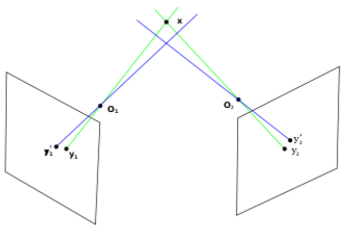


Figure 10. *Triangulation*

Bundle Adjustment[8]. Bundle adjustment is responsible for minimizing projection errors related to the previous stages of registration and triangulation..

Sparse reconstruction. The output of the incremental structure is a set of camera pose estimates and points generating the reconstructed scene.

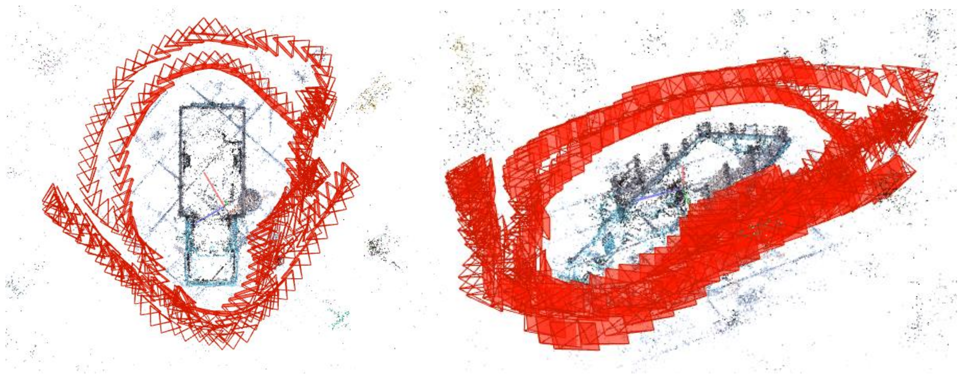


Figure 11. *Sparse reconstruction of Truck dataset.*

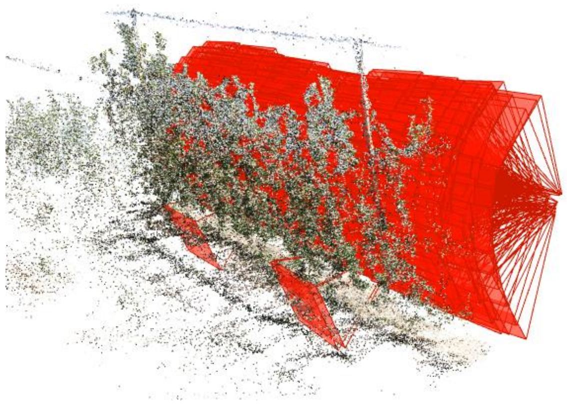


Figure 12. *Sparse reconstruction of Fuji dataset. Big % overlap between cameras. Sequential motion of the cameras can also be appreciated.*

Sparse reconstruction is not the last step of the process but the basis to refine and get a much better resolution object.

Dense reconstruction. Is responsible to generate a higher resolution scene. From the sparse reconstruction, colmap produces normal and depth maps and fuses them to generate the dense scene.

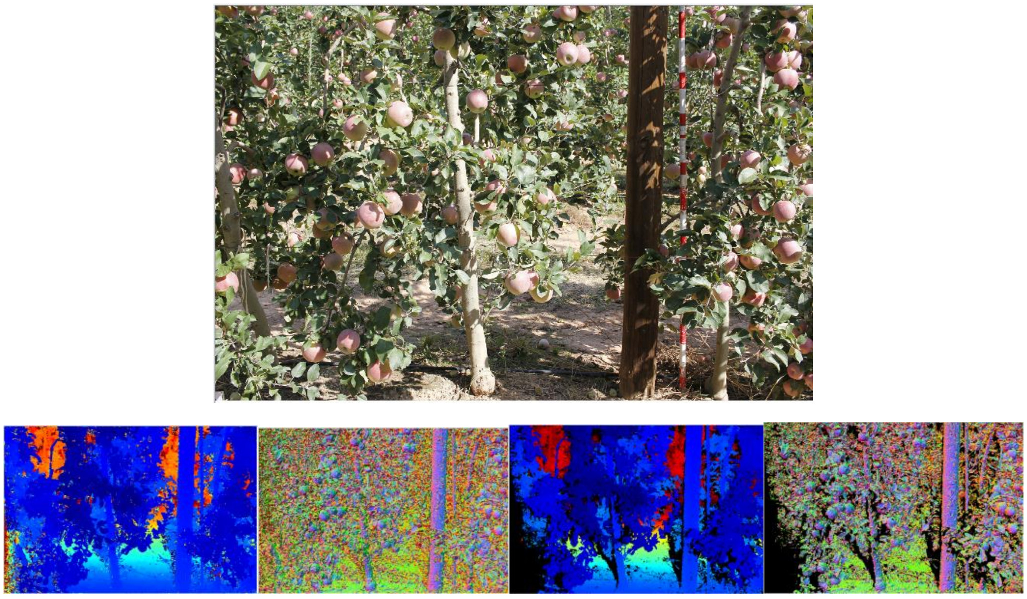


Figure 13. *Top picture is the source. Center are depth maps (photometric left – geometric right). Bottom, normal maps (photometric left - geometric right)*

Fusion results in an improved version in terms of resolution of the reconstruction.

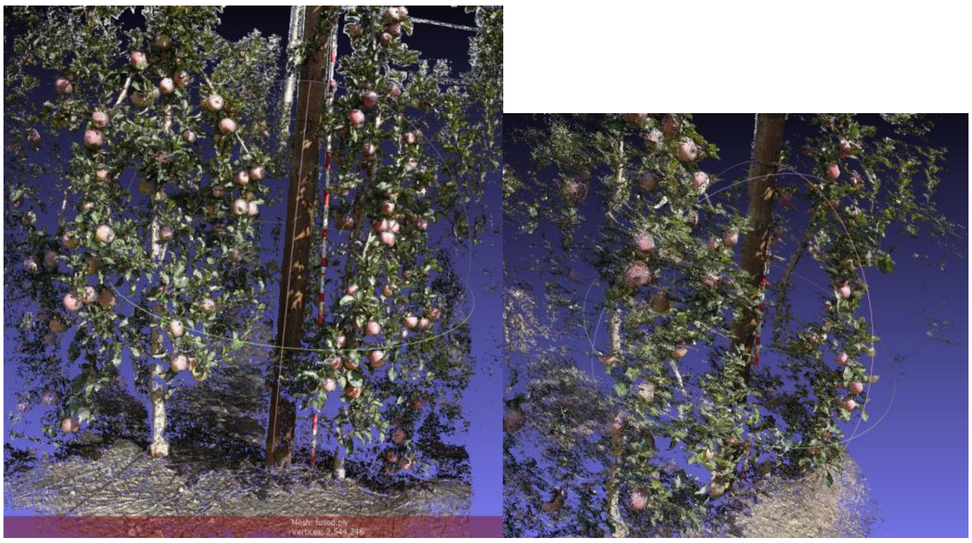


Figure 14. *Dense reconstruction of Fuji dataset.*

3.3. Metadata reconstruction pipeline

This section unveils the process taken to insert metadata to the 3D space once a first reconstruction is done.

The implemented solution lives within the colmap reconstruction pipeline, taking advantage of its processes and adding value to the system. In order to add labels to a 3D object, a colmap reconstruction is previously executed, so that access to the generated information is enabled.

The key of this process is to be able to select or find the proper input to the subsystem and generate an output integrated with the general system.

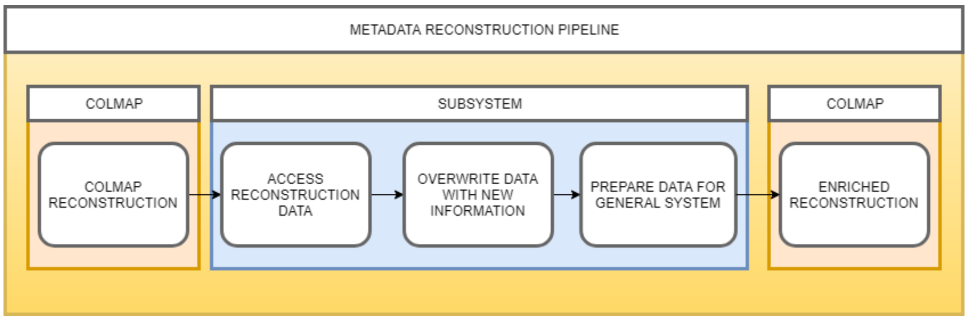

METADATA RECONSTRUCTION PIPELINE
COLMAP
SUBSYSTEM
COLMAP
COLMAP RECONSTRUCTION
ACCESS RECONSTRUCTION DATA
OVERWRITE DATA WITH NEW INFORMATION
PREPARE DATA FOR GENERAL SYSTEM
ENRICHED RECONSTRUCTION

Figure 15. *Metadata reconstruction pipeline.*

Two variants for the subsystem are proposed, although only the second one, with external scripts, has been fully tested and validated.

Full integration with colmap. A full integration with colmap means extending the source code and rewriting the c++ classes to get a total integration of our subsystem in the colmap solution. This variant is better in terms of having a full end to end solution within the same software.

External scripts. External scripts provide more flexibility to our subsystem, although colmap is also the base of our process, the parsing of the data is done outside the source code. The endpoints of the block have to be strictly analysed for an adaptation without loss of information. This variant is better for changing and playing with different configurations without having to adapt all of its dependencies. Python is the chosen language for the scripts.

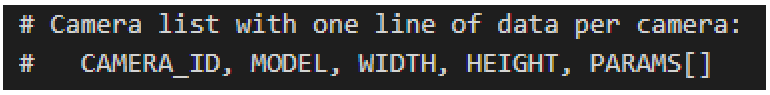
Five system blocks integrate the process followed to achieve the desired output:

3.3.1. Colmap reconstruction

The input of our subsystem is the information containing relations between images or cameras, its 2D points and the generated 3D points. It is not the 3D object in the ".ply" format that is needed, but the data from which it originates.

Reconstruction data containing needed relations is generated at the sparse reconstruction step of the colmap pipeline. It is necessary to specify where this data is going to be stored and in which format. Colmap is able to generate three files containing the desired information in .bin or .txt format. Although .bin format provides a better performance in terms of efficiency for the system read and write operations, .txt provides the ability to understand which information each file contains.

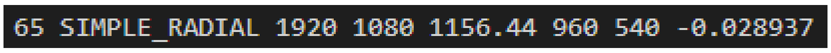
- **cameras.txt.** This file contains information about the intrinsic parameters of the cameras or views of the reconstruction, each camera is identified by an id and has the following list of fields:



```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
```

Figure 16. *"cameras.txt" structure*

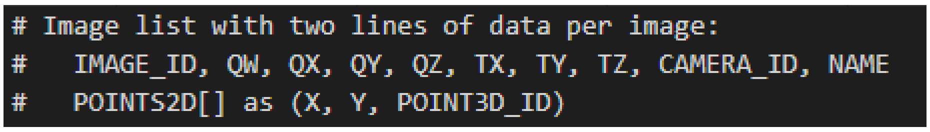
  - ❑ CAMERA_ID. The id of the camera. Represented by an integer.
  - ❑ TYPE OF MODEL. From within a preconfigured list of camera models of varying complexity or a custom-made model. Camera models vary depending on which set of parameters adapts better for each case scenario. Focal length, principal point or distortion parameters are among these parameters.
  - ❑ WIDTH. Width of the camera sensor (image resolution)
  - ❑ HEIGHT. Height of the camera sensor (image resolution)
  - ❑ PARAMS[]. The list of parameters, depends on each camera model.



```
65 SIMPLE_RADIAL 1920 1080 1156.44 960 540 -0.028937
```

Figure 17. *"cameras.txt" data example. In this example, 65 would be the id of the camera, SIMPLE_RADIAL the type of model, 1920x1080 the camera sensor size and the following values represent the parameters of the SIMPLE_RADIAL camera model. Those being focal length (1156.44), principal point (960, 540) and one radial distortion parameter (-0.0289)*

- **images.txt.** This file contains information about the images of the dataset that originate the 3D object. Each image is identified by an id and contains the following of fields:



```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
```

***Figure 18**. "image.txt" structure*

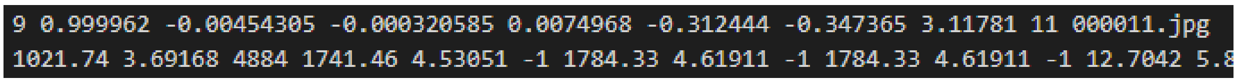
□ IMAGE_ID. Id of the image. Represented by an integer.

□RECONSTRUCTED POSE. The reconstructed pose of the image is specified as a quaternion[19] (qw,qx,qy,qz) and a translation vector (tx,ty,tz), representing the projection from world to camera coordinate system.

□ CAMERA_ID. Camera id as an integer.

□ NAME. Source image file.

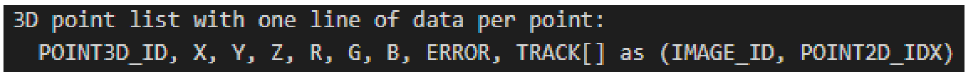
□POINTS2D[]. For each feature position, specifies if a 3D point is generated from it (point3d_id) or not (-1).



```
9 0.999962 -0.00454305 -0.000320585 0.0074968 -0.312444 -0.347365 3.11781 11 000011.jpg
1021.74 3.69168 4884 1741.46 4.53051 -1 1784.33 4.61911 -1 1784.33 4.61911 -1 12.7042 5.8
```

***Figure 19.** "image.txt" data example. The image with id 9 that originates from source file 000011.jpg generates a point 3D from feature coordinates (1021.74 , 3.69168)*

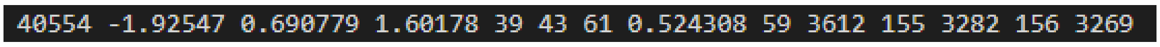
- **points3D.txt.** This file contains information about the 3D points of the sparse reconstruction of the 3D object. A point is also identified by an id and contains the following fields:



```
3D point list with one line of data per point:
  POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
```

***Figure 20.** "points3D.txt" structure*

- ❑ POINT3D_ID. Unique integer identifier of the 3D point.
- ❑ COORDINATES. XYZ coordinates of the point int the 3D object.
- ❑ COLOR. Color of each point. The process of attaching color to the 3D point is mimicked for the label information.
- ❑ ERROR. Reprojection error.
- ❑ TRACK[ ]. Relevant information to find the sources of the 3D point generation.



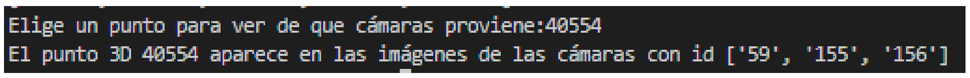
```
40554 -1.92547 0.690779 1.60178 39 43 61 0.524308 59 3612 155 3282 156 3269
```

***Figure 21.** "points3D" example. The point with id 40554 at (-1.93, 0.69, 1.6) originates from images 59, 155 and 156*

3.3.2. Access and parse reconstruction data

In order to access and parse the reconstruction data, a python script is used. The input files are located by the user in the .txt format instead of the .bin format. The script is able to, depending on each file, parse its data and locate each field value.

For efficiency, not all the variables are stored but only the required ones for each operation. If we are looking to locate the images from which a 3D point generates, we will identify the specified point, and look for and store only the image_id field.



```
Elige un punto para ver de que cámaras proviene:40554
El punto 3D 40554 aparece en las imágenes de las cámaras con id ['59', '155', '156']
```

Figure 22. *Code snippet. Point 40554 is selected and origin images are successfully identified (59, 155, 156).*

Storing only the required data fields will make a huge impact in the system efficiency since the input files can become inconveniently large.

3.3.3. Overwrite 3D Point with new information

As label information can be understood to be in the same level as color data, the same structure is used. Since color is specified with RGB values in the points3D file, the format for the labels needs to be also stated.

For easiness of use and understanding, the format of the labels will be directly a string, without codification of any type. Changing the format will be a key factor for optimizing the solution.

To attach the information where it matters, the script will follow this procedure:

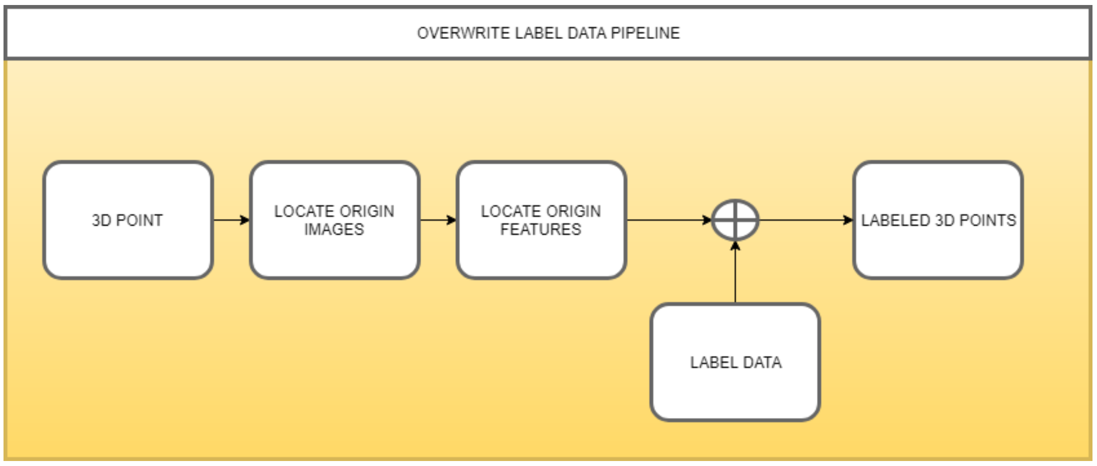


Figure 23. *System diagram for attaching labelled information to the 3D point.*

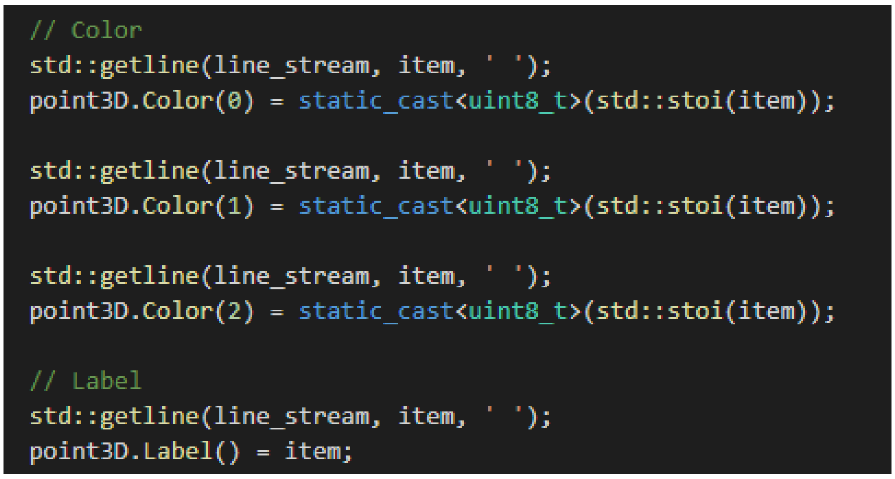
Labels will be only looked for in the located origin features coordinates. If a label exists for that position, it will be inserted into the attributes for a point3D in the points3D file.

3.3.4. Prepare data for general system

The following process is a proposition, it could not be fully verified since issues at the build stages of the solution were raised.

Although information is already attached at this point and it can be seen externally, we need to reinsert the new information in the colmap pipeline in order to be able to reconstruct the 3D object with the new data.

To achieve this, modifications of the point3D C++ object need to be done. First, the system needs to be able to read the new attached data, to do so, the C++ Read methods from the "`reconstruction.cc`" source file are rewritten.



```
// Color
std::getline(line_stream, item, ' ');
point3D.Color(0) = static_cast<uint8_t>(std::stoi(item));

std::getline(line_stream, item, ' ');
point3D.Color(1) = static_cast<uint8_t>(std::stoi(item));

std::getline(line_stream, item, ' ');
point3D.Color(2) = static_cast<uint8_t>(std::stoi(item));

// Label
std::getline(line_stream, item, ' ');
point3D.Label() = item;
```

Figure 24. *Code snippet of "reconstruction.cc" used to parse the "points3D" file. Information is stored as point3D attributes in the point3D object. line_stream is a string stream for each line. For each space, each value is stored in the point3D object attribute.*

Since a point3D is generated from multiple sources, we need to specify a method for when different labels are attached to the origin features that generate the 3D point. The merging solution simply decides the output label by checking the one with most appearances. In case of doubt, the first found label will remain in the selected position.

3.3.5. Enriched reconstruction

The following process is a proposition, it could not be fully verified since issues at the build stages of the solution were raised.

The enriched reconstruction represents the full integration of our subsystem in the colmap pipeline. The user is able to explore the point metadata from a user-friendly interface and even has the ability to select new bounding boxes or masks.

The result of the previous steps results in successfully attaching the information to the 3D point of the sparse reconstruction, but not to the dense reconstruction. An additional adaptation is needed to expand the capabilities of the pipeline.

4. Results

This section presents the achieved results of the system by showing a step by step process from a 2D bounding box with labelled information to its reconstruction in the 3D space.

It is not a usual numeric evaluation of the system since there is a lack of objective methods to do so. So instead we focus on each step of the process to expose a qualitative evaluation at the end.

2D bounding box. A bounding box is selected representing a labelled region for an image (of the truck dataset). All the features inside of the bounding box will be adding extra data for its reconstruction.

A rectangular bounding box of vertices (670, 774), (890,774), (370,888) and (890, 888) is selected.

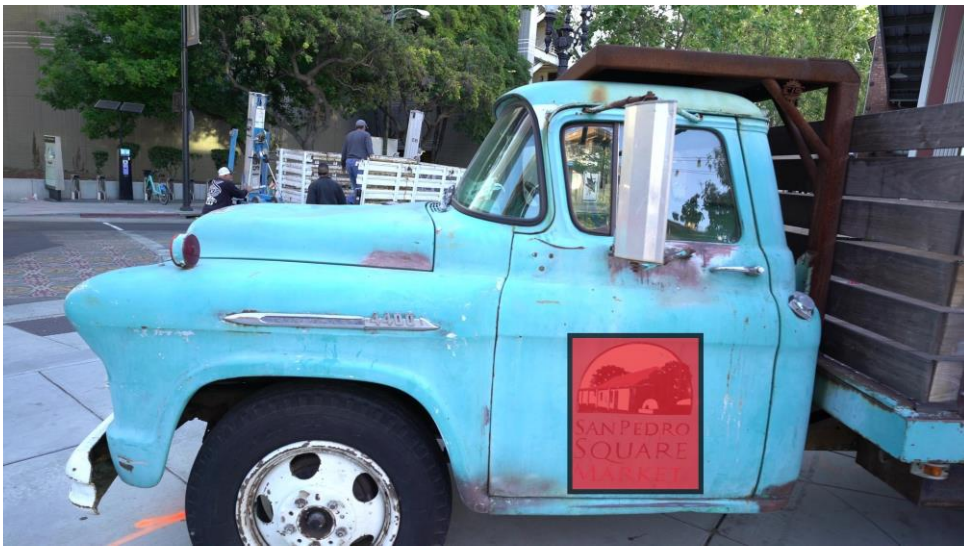
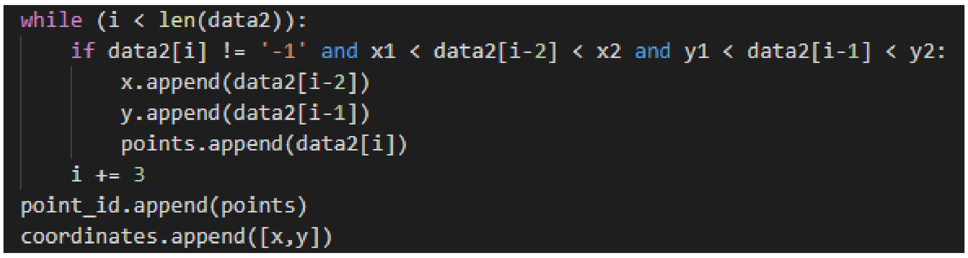


***Figure 25**. 2D Bounding box* (in red)

Feature information. Since there already is information of the reconstructed 3D object and the features from which it originates, now a validation check of the existence of any features that generate 3D points in the bounding box is performed.

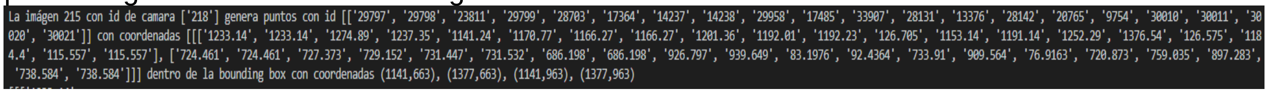
To do so, access to the "images.txt" files is granted followed by the check if any of the features that generate a point exist inside the bounding box.



```
while (i < len(data2)):
    if data2[i] != '-1' and x1 < data2[i-2] < x2 and y1 < data2[i-1] < y2:
        x.append(data2[i-2])
        y.append(data2[i-1])
        points.append(data2[i])
    i += 3
point_id.append(points)
coordinates.append([x,y])
```

***Figure 26.** Checking features inside the bounding box.*

For the example image, the script is able to find a total of 20 features from which 3D points originate inside the bounding box.



```
La imágen 215 con id de camara ['218'] genera puntos con id [['29797', '29798', '23811', '29799', '28703', '17364', '14237', '14238', '29958', '17485', '33907', '28131', '13376', '28142', '20765', '9754', '30010', '30011', '30
020', '30021']] con coordenadas [[['1233.14', '1233.14', '1274.89', '1237.35', '1141.24', '1170.77', '1166.27', '1166.27', '1201.36', '1192.01', '1192.23', '126.705', '1153.14', '1191.14', '1252.29', '1376.54', '126.575', '118
4.4', '115.557', '115.557'], ['724.461', '724.461', '727.373', '729.152', '731.447', '731.532', '686.198', '686.198', '926.797', '939.649', '83.1976', '92.4364', '733.91', '909.564', '76.9163', '720.873', '759.035', '897.283',
 '738.584', '738.584']]] dentro de la bounding box con coordenadas (1141,663), (1377,663), (1141,963), (1377,963)
```

Figure 27. *Features inside bounding box check output for image 215*

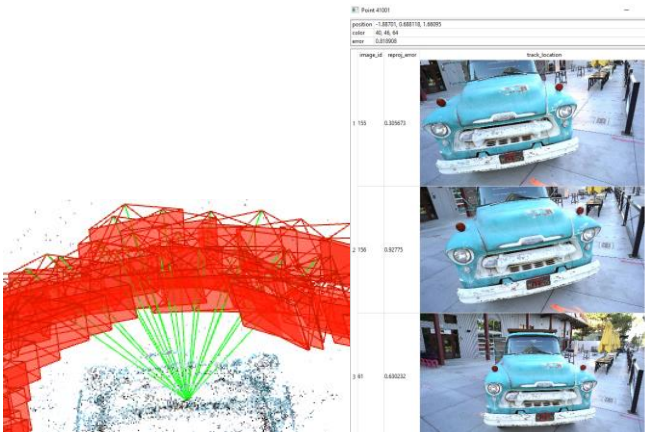


Figure 28. *Camera sources for a given point.*

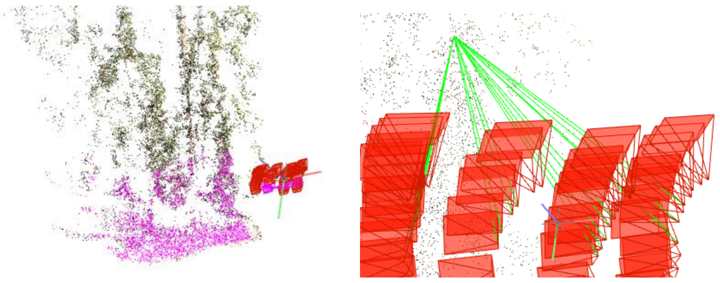


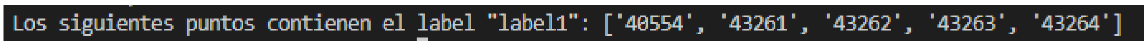
Figure 29. *Points generated from a given camera(left). Cameras that generate a given point(right)*

Overwrite 3D point with new information. Once the feature is checked so that it belongs to the bounding box, the value of the 3D point is stored.

Now access to the “points3D.txt” is required. All the 3D points found in the previous step are overwritten with the new information in its metadata. To do so we parse the “points3D.txt” file to find the saved points so that we can attach the bounding box information to each data line. Although not optimal for efficiency purposes, each bounding box is represented by a string label, so that a string object with the selected value is written in the point3D data.

As stated in the methodology, the merging of the labels is also taken into consideration when multiple labels are assigned for the same 3D point.

Prepare data to general system. Now that the data is attached to the 3D point, label information for a specific 3D point, not a region, is available. To access this information, parsing of the new “points3D.txt” is required. Checking if a certain label is present in the points3D file will return the list of points containing a certain label.



```
Los siguientes puntos contienen el label "label1": ['40554', '43261', '43262', '43263', '43264']
```

Figure 30. *Listed points with a certain label.*

Following this procedure we are now able to proceed to the usual reconstruction of the object with the extra parameters in its structure, although it is not immediately seen in the interface.

Subjective evaluation. This pipeline is able to attach all of the data to the 3D sparse reconstruction points but fails to extrapolate the information to the more interesting dense reconstruction. We could evaluate the efficiency of the system in terms of the reconstruction quality from the ground truth, but since the current state of the system works up to the sparse reconstruction, it doesn't seem fair. Thus, space for future work exists to expand the capabilities of the software from the built basis in order to achieve better resolution results.

5. Budget

The object of the thesis is not a prototype, so the budget is not related to production costs of materials, chain of production or any other related costs.

The budget is generated as follows:

PEOPLE

POSITION	WAGE	HOURS	TOTAL
Intern engineer	8€ /hr	720h (24 ECTS)	5760 €

Table 1. *People budget disclosure*

DEVELOPMENT

COLMAP		0€ (BSD Licensing)	
SERVER*	PRICE /hour	HOURS	TOTAL
AWS instance m5.4xlarge	0,74 € / hr[9]	800 **	592€

Table 2. *Development budget disclosure*

An m5.4xlarge instance has 16 virtual CPUs + 64GB of RAM available and limited volume storage.

TOTAL BUDGET	6352 €

Table 3. *Total budget disclosure*

*Although access to the UPC imatge research group was granted, it was only accessed for data retrieval for convenience. Calculations have been made taken into account an AWS instance.

**Estimated time.

6. Conclusions and future development:

By implementing an integrated solution to the problem, the project makes sense in a mostly closed environment instead of having totally separated modules for each capability.

Success of annotation is related to the ability to extrapolate the information from the lower resolution sparse model to the dense model with higher resolution. If this gap is not ensured, the viability of the system can be questioned.

Although the final achieved objectives might differ from the original ones, an important work of adaptation has been done to provide a basis from which to build upon.

6.1. Future development

This project was not the start nor is the finish of the global project, in those terms, future developments is needed in this module.

- **Point selection**. Evolve from scripts to user-friendly interface to select points and see metadata.
- **Integration with colmap GUI**. Integrate the process of extraction and read of metadata available from colmap's GUI.
- **Integrate the contribution to the original project community**. Giving back to the community of open source projects is key for the expansion and share of knowledge.

Bibliography:

[1] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[2] Li, Yunpeng, Noah Snavely, and Daniel P. Huttenlocher. "Location recognition using prioritized feature matching." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2010.

[3] Snavely, Noah, Steven M. Seitz, and Richard Szeliski. "Skeletal graphs for efficient structure from motion." *CVPR*. Vol. 1. 2008.

[4] Knapitsch, Arno, et al. "Tanks and temples: Benchmarking large-scale scene reconstruction." *ACM Transactions on Graphics (ToG)* 36.4 (2017): 78.

[5] Beder, Christian, and Richard Steffen. "Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence." *Joint Pattern Recognition Symposium*. Springer, Berlin, Heidelberg, 2006.

[6] Lepetit, Vincent, Francesc Moreno-Noguer, and Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem." *International journal of computer vision* 81.2 (2009): 155.

[7] Hartley, Richard I., and Peter Sturm. "Triangulation." *Computer vision and image understanding* 68.2 (1997): 146-157.

[8] Triggs, Bill, et al. "Bundle adjustment—a modern synthesis." *International workshop on vision algorithms*. Springer, Berlin, Heidelberg, 1999.

[9] AWS pricing available at: https://aws.amazon.com/es/ec2/pricing/on-demand/

[10] VSFM software documentation available at: http://ccwu.me/vsfm/

[11] Bundler repository available at: https://github.com/snavely/bundler_sfm

[12] AgiSoft software available at: http://www.agisoft.com/features/professional-edition/

[13] Schönberger, Johannes L. Colmap repository available at: https://github.com/colmap/colmap.

[14] MicMac repository available at:https://github.com/micmacIGN/micmac

[15] Collins, Robert. CSE, Penn State. "Lecture 13: Camera projection II" available at http://www.cse.psu.edu/~rtc12/CSE486/lecture13.pdf

[16] Lcas repository available at: https://github.com/lcas/cloud_annotation_tool

[17] Furukawa, Yasutaka, and Jean Ponce. "Accurate, dense, and robust multiview stereopsis." *IEEE transactions on pattern analysis and machine intelligence* 32.8 (2010): 1362-1376.

[18] Westoby, Matthew J., et al. "'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications." *Geomorphology* 179 (2012): 300-314.

[19] https://www.geometrictools.com/Documentation/RotationIssues.pdf

[20] Washington University Computer Science Department "Lecture 5: Cameras, Projection, and Image Formation" available at https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect5.pdf

[21] Collins, Robert. CSE, Penn State. "Lecture 12: Camera projection" available at http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf

[22] Tensorflow object detection repository available at: https://github.com/tensorflow/models/tree/master/research/object_detection

[23] Cloud factory software available at: https://www.cloudfactory.com/computer-vision

[24] Schönberger, Johannes L., et al. "Pixelwise view selection for unstructured multi-view stereo." *European Conference on Computer Vision*. Springer, Cham, 2016.

[25] Schonberger, Johannes L., and Jan-Michael Frahm. "Structure-from-motion revisited." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[26] R. Monica, J. Aleotti, M. Zillich, M. Vincze, Multi-Label Point Cloud Annotation by Selection of Sparse Control Points, International Conference on 3D Vision (3DV), Qingdao, China, October 10th-12th 2017

[27] Tarsha-Kurdi, Fayez, Tania Landes, and Pierre Grussenmeyer. "Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data." *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*. Vol. 36. 2007.

Appendices (optional):

Colmap github repository: https://github.com/colmap/colmap.

Interesting to explore the /src/base files:

- Reconstruction
- Camera_models
- Camera
- Point3D
- Projection

Custom python scripts at: https://github.com/herlazp/3dannotation

Glossary

PC.	Point Cloud
SfM.	Structure from Motion
SIFT.	Scale Invariant Feature Transform
GUI.	Graphic User Interface
RANSAC.	Random Sample Consensus
MVS.	Multi View Stereo