CEIG - Spanish Computer Graphics Conference (2012)
Isabel Navazo and Gustavo Patow (Editors)

3D Digitization using Structure from Motion

J.C. Torres[1], G. Arroyo[1], C. Romo[2], J. De Haro[2]

[1] Virtual Reality Lab, University of Granada, Spain. http://lrv.ugr.es
[2] Virtum Graphics. http://www.virtumgraphics.com

Abstract

Computational methods to generate three dimensional models from photographies have received increasing attention in the last years. Nowadays, although they are not a substitutive to laser scanners, they are a viable alternative in cases were using a laser scanner is not possible. These methods are also attractive because they do not need the acquisition of expensive equipments.
This tutorial introduces reconstruction methods focusing on the structure from motion approach, giving a brief explanation of its theory, explaining some basic algorithms, and the installation and use of Visual Structure from Motion *software.*

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Graphics Systems—

1. Introduction

Three dimensional reconstruction is the automatic or assisted generation of a 3D model that is a precise copy of a real object. Any 3D reconstruction method has some kind of capturing device, that is, a device that gets shapes and measures of the object. The information retrieved from the device must be processed in order to generate a 3D model using specific software. The complexity of this software process depend on the capturing device used. Capturing devices also determine the precision of the models.

One of the simpler capturing devices is the digitizing arm, that is able to measure the position of the points on the object surface that are touched by the user using the arm. Processing software for this device must generate a 3D triangle mesh using these points as vertexes.

This tutorial addresses 3D reconstruction using photographic cameras as capturing devices (see figure 1). That is, the information retrieved from the devices are photographs. There are many approach to process photographs generating 3D models, depending on the number of photographies, the relation existing between them and how deep information is obtained. One drawback of these methods is that they can not capture scale.

Section 2 gives a short review of the most important methods. The remainder of the tutorial focuses on *structure from motion* methods.

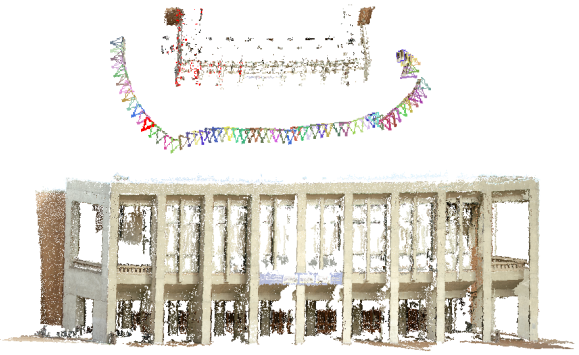

Figure 1: *3D model generation (down) from a set of images (up)*

2. Techniques

Human vision captures depth information using different sensory cues [RHFL10]. The most important way to acquire 3D information for humans is by means of stereoscopic vision. Stereoscopic vision uses the information obtained by the two eyes. The image of any observed object is projected from different views on both retinas, and their displacement can be used to triangulate the position of the object. Many people think that this is the only source of depth information, but this is false. The human vision system can obtain

depth information from the convergence of both eyes, that is, their angular position when looking to a fixed point can be also used to compute its depth using triangulation.

Moreover, we can retrieve depth information using only one eye, that is, using monocular cues. The most important monocular cues are:

Accommodation. The ciliary muscles stretch the eye lens to change the focal length. The state of the muscle is used to interpret depth.

Motion parallax. When the observer moves, the apparent displacement of objects is proportional to its depth.

Motion. The displacement of moving object projection depends on its velocity and distance. If we can infer its velocity we can estimate its distance.

Size. The size of the projection depends on the object size and distance. For known objects the brain infers the distance from the apparent size.

Perspective. Projections of parallel lines converge at vanishing points. The apparent distance of points on the two parallel lines gives depth information. This was the base of perspective drawing that was used since the 15^{th} century to produce drawing and painting.

Interposition. Objects occlusion gives information of relative object depth.

Lighting and shading. Reflections, illumination and shadows give frequently enough information to infer the position and shape of objects.

Distance fog. Objects that are far away have lower luminance contrast and lower color saturation.

Defocus blur. The focus depth of human eye is limited. Objects behind or ahead the focus distance appear blurred.

Some of these cues have been used to obtain information from photographs. The remainder of this section comments the most relevant ones.

2.1. Vanishing points

For scenes containing a large number of parallel lines, it is possible to identify a set of them and estimate the position of vanishing points. Once vanishing points have been computed, it is possible to compute 3D coordinates of any image pixel just using the pixel information and its ground projection on the image. This technique, that was introduced by Liebowitz in 1999 [LCZ99], can be used for a single photograph, as it only uses perspective cue (figure 2 shows a reconstruction example). It is quite useful for architectural models. It has the advantage of generating directly a quad mesh model, which has a small number of textured polygons.

2.2. Epipolar geometry

Epipolar geometry is the classic 3D reconstruction method. It uses stereoscopic information, and so, it need two

Figure 2: *3D model generated using vanishing points computation [RG04].*

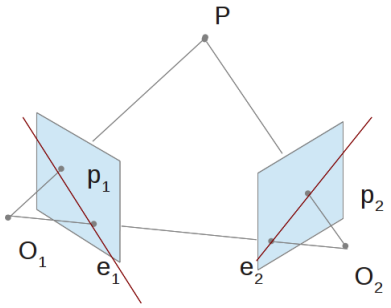

Figure 3: *Epipolar geometry.*

images obtained from two different cameras (or the same one at slightly different positions). We define two cameras located as shown on figure 3, where O_1 and O_2 are the projection centers and the blue quads are the projection planes of both cameras. The image of a point P on both cameras is P_1 and P_2. If the point P moves on the line O_1-P its image on the first camera will be the same, that is, all the line is projected on a point for this camera. For the second camera the image will move on the line e_2-P_2, which is called epipolar line. Points e_1 and e_2 are called epipoles, they can be computed intersecting the line O_1-O_2 and the two projection planes [SS01].

Generating the 3D models is straightforward when the two images of a point, P_1 and P_2, and projection centers, O_1 and O_2, are know. It can be done computing the intersection of lines O_1-P_1 and O_2-P_2. The only practical problem is that due to precision errors it is possible that lines do not intersect, and so minimum distance point must be computed instead of intersection point [MSK*03].

The main problems are to get the projection centers and to find the corresponding pixel on the second image for any given pixel on the first one. Projection centers depend on the camera parameters and location, so if cameras are mounted on a fixed structure they can be computed once, as a calibration process. Anyway, the calibration can be done computing the fundamental matrix from nine known corresponding point pairs [FLP01].

In order to find out corresponding pixels we must look for similar pixels on the epipolar line.

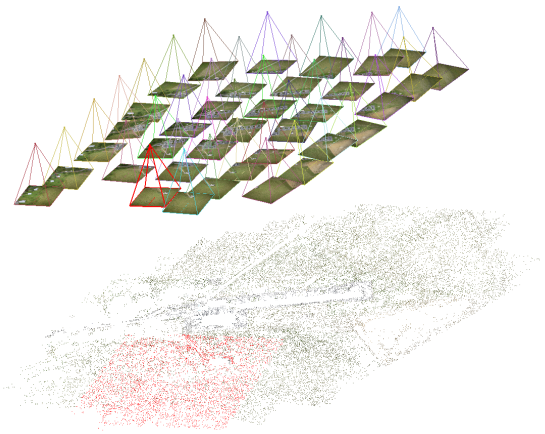

Figure 4: *Structure from motion result: sparse point cloud and cameras parameters*

2.3. Shape from shading

Shape from shading is a technique that tries to exploit lighting and shading cues. The underlying idea is to find out the surface that generates a shading similar to the shape of the input image [ZTCS99]. This problem can be solved as an optimization problem using different approaches. Nevertheless, it has some important drawbacks: shading cue gives ambiguous information (it is possible to obtain the same image for different geometries changing the position of the light); and it can be used only for diffuse materials without texture and with a single light. Despite these problems, shape from shading can be used to enrich the model generated using other techniques.

2.4. Structure from motion

In the fields of computer vision and visual perception, structure from motion (SfM) refers to the process of finding the three-dimensional structure of an object by analyzing local motion signals over time.

Structure from motion generates the 3D model from the motion cue. Assuming all object in the scene are static, its relative displacement depends only on their depth. This technique generates the model from a set of photographs or a video. More precisely, SfM systems take as input a set of images and generate a 3D point cloud of the scene, the position from which each photo has been taken, and optical parameters of the cameras [HZ03, FLP01, DSTT00, MSK*03]. Figure 4 shows the result obtained running a SfM algorithm on the set of aerial photographs shown on figure 9. It shows the position an orientation of the cameras used to capture the scene, this information has been computed by the SfM algorithm.

The remainder of this tutorial focuses on 3D reconstruction using structure from motion.

3. Structure from motion algorithms

Finding structure from motion presents a similar problem as finding structure from stereo vision. In both instances, the correspondence between images and the reconstruction of the 3D object need to be found.

First step in the algorithm is usually to find correspondences between images even when algorithms that do not use these correspondences [DSTT00] are used. In order to find these elements, features such as corner points need to be found and tracked from one image to the next. Features of an image are relevant areas of the image that remain stable in the next image, such as corners (edges with gradients in multiple directions).

The feature trajectories over time are then used to reconstruct their 3D positions and the camera's motion. This task is challenging because the image formation process is not generally invertible, hence additional information is needed to solve the reconstruction.

One possibility is to exploit prior knowledge about the scene to reduce the number of degrees of freedom [BRLT03]. Another possibility is to use corresponding image points in multiple views to extract 3D information [UHS07]. In this tutorial we focus on the last option to do a generic 3D reconstruction from any kind of images without prior knowledge.

3.1. Detecting features

In computer vision and image processing the concept of feature is used to denote a piece of information which is relevant for solving the computational task related to a certain application.

There are various algorithms for feature detection, but the most popular are which are based on Scale-invariant feature transform (SIFT) [Low99].

3.2. Matching features

Matching features is a simple algorithm that tries to identify a feature in the rest of available images. This is a necessary process due to the necessity of compute the triangulation in a later step.

One way to avoid search for all pixels in every image is to use epipolar lines. Assuming we have a pinhole model of camera, when two cameras view the same 3D scene from two distinct view-points, there are a number of geometric relations between the 3D points and their projections onto the 2D images. This leads to constraint between the image points.

In computer vision, specifically computational stereo, we can easily write an algorithm to find corresponding points for the two camera images. The algorithm can be written in pseudo-code like this:

```
Repeat                                                          1
  for each feature point in the left image {                    2
    compute the epipolar line in the right image                3
    If the epipolar line intersects only one feature point      4
    Then match those points and remove them from the lists      5
  }                                                             6
  Until no feature point can be matched uniquely                7
```

If we have more than two images, we can modify the algorithm to match features as follows:

```
Repeat                                                          1
  for each feature point in the first image {                   2
    compute the epipolar line in the second image               3
    compute the epipolar line in the third image                4
    If the epipolar line in either image intersects only        5
         one feature point {
      compute epipolar line for matching feature point in       6
           the other image.
    Intersection with epipolar line from first image gives      7
         the third point.
    Remove triplet from the List                                8
  }                                                             9
  Until no feature point can be matched uniquely                10
```

The process is repeated processing the features in the second image and so on [IG98].

This algorithm can be improved if more information is available or if some mathematical property can be obtained, such as in [LM04].

3.3. Reconstruction from different cameras

Given two or more views, a 3D point can be reconstructed by triangulation. An important prerequisite is the determination of camera calibration and pose, which may be expressed by a projection matrix. The geometrical theory of structure from motion allows projection matrices and 3D points to be computed simultaneously using only corresponding points in each view. When only two images are available, algorithms may have errors that make impossible the 3D reconstruction. In order to obtain a better accuracy we have to use more images.

The problem now is how to dynamically adjust the camera according to the different viewpoints. Bundle Adjustment can be defined as the problem of refining simultaneously the 3D coordinates describing the scene geometry as well as the parameters of the relative motion and the optical characteristics of the camera(s) employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all points. This problem is usually named *Bundle Adjustment*.

Bundle Adjustment amounts to jointly refining a set of initial camera and structure parameter estimates for finding the set of parameters that most accurately predict the locations of the observed points in the set of available images. Bundle Adjustment boils down to minimizing the reprojection error between the image locations of observed and predicted image points, which is expressed as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms.

One of the most used algorithm to solve this least-square problem is Levenberg–Marquardt algorithm (LMA) [Lev44], also known as the damped least-squares (DLS) method. It provides a numerical solution to the problem of minimizing a function, generally nonlinear, over a space of parameters of the function.

Lot of different variations of the basic algorithm has been used along the time, but most of them try to improve the performance of the algorithms by using clustering computation and Graphical Process Units [ASSS10] or Data Bases [IZKB10] to drastically reduce the time of computation. Some of them try to solve the problem when discontinuities appear in the images or there are hidden objects [THWS10].

There are also a lot of different libraries and software that realize Bundle Adjustment, such as SBA [Lou], SSBA [Zac11]. MCBA [Wu11], libdogleg [Kog11], or ceres-solver [AM12], among others.

4. Software

This section reviews some structure from motion software, focusing on open source implementations. Tools can be grouped into three categories:

- Individual algorithms, that solve specific tasks within the reconstruction process.
- Standalone tools, that solve the whole reconstruction process generating either a 3D mesh or a dense point cloud.
- Web services, to which images can be uploaded returning to the user the 3D model.

Following subsection introduces some of the most popular software.

4.1. Bundler

Bundler is an *open source* structure from motion system that works incrementally. It takes as input unordered images generating an sparse 3D points set and parameters of the photographs as output. It was used in the Photo Tourism project [SSS06, SSS07].

4.2. PMVS2

PMVS2 is a multi-view stereo software that takes a set of images and camera parameters as input and reconstructs the 3D structure of the scene as a dense points set. The software

obtains a set of oriented points instead of a triangle mesh, where both 3D coordinate and surface normal are estimated at each oriented point.

4.3. CMVS

CMVS takes, as input, the output of a Structure from Motion (SfM) software, then decomposes the input images into a set of image clusters of small size, allowing an MVS software to be used to process each cluster independently and parallely, ensuring that the union of reconstructions from all the clusters should not miss any details that can be otherwise obtained from the whole image set. CMVS should be used in conjunction with an SfM software (Bundler) and an MVS software (PMVS2).

4.4. VisualSFM

VisualSFM is an *open source* software GUI that integrates and combines diverse SfM software. It is described in section 5.

4.5. RunSFM

RunSFM is another *open source* software package integrating Bundler and CMVS [Ho12].

4.6. Insight3D

Insight3D is an *open source* 3D reconstruction and modeling software which combines structure from motion and reconstruction from vanishing points. It can be used to create 3D models from photographs, and includes interactive modeling tools to create textured polygonal model [Mac].

4.7. AgiSoft PhotoScan

This is a commercial software (http://www.agisoft.ru) running on Windows operating system. The cost for a professional version is over 3000 euros. It generates good quality dense point cloud [NK11].

4.8. Pix4D

Pix4D is a commercial software system (http://pix4d.com/). It has been specially designed for aereal photographs generated from UAVS. Its web page contains also sample data.

4.9. Microsoft Photosynth

Microsoft Photosynth is a web service that compute Bundler Adjustment. It can be accessed at http://photosynth.net at no charge.

4.10. ARC3D

Arc3D is a web based reconstruction service for automatic matching of image features that compute a depth map for each photograph [VG06].

4.11. Autodesk 123D

123D is the Autodesk structure from motion software. It is a commercial tool, although at the time of written this tutorial it can be downloaded for free from Autodesk web page (http://www.123dapp.com/). It computes dense points clouds.

4.12. Comparing tools

Some studies have been carried out comparing different SfM methods. We describe some of the comparison from Wang and Neitzel reports. Wang compares Bundler, Bundler with PMVS2, ARC3D and their own SfM method that uses Delaunay triangulation to stabilize the computation of stereo correspondence [CST*08]. His conclusion is that ARC3D generates an outputof a size between 5% and 20% of the points cloud returned by Bundler using PMVS2. His method performs between 2 and 5 times more accurate than Bundler [Wan11].

Neitzel compares Bundler, Bundler configured to use PMVS2, Microsoft Photosynth, ARC3D and AgiSoft PhotoScan. They use 99 aerial photographs for a small area [NK11]. The worst result is reported again for ARC3D which generates a coverage of only half of the scene. They also conclude that Bundler and Photosynt performance is quite similar, and Bundler with PMVS2 gives a similar result to PhotoScan, although the first one generates some small holes that do not appear on the second one.

5. VisualSFM

As it has been previously mentioned, VisualSFM is a GUI application for Structure from Motion (SfM) that integrates and improves the following algorithms: an incremental SfM system, SIFT on the GPU, and the Multicore Bundle Adjustment. In addition to the sparse reconstruction, VisualSFM provides an interface to run Yasutaka Furukawa's PMVS/CMVS tool that is used for 3D dense reconstruction. This software also partially integrates the Viewpoint Invariant Patch and Repetition Analysis projects.

VisualSFM needs other packages: Gtk, SiftGPU, CUDA, GLEW, GLUT (freeglut is also a valid option), DevIL, and PBA, between others. We describe all these packages briefly:

Gtk is an very well known user-interface library. It has been use for example in Gnome, Firefox, and many other well known software. VisualSFM uses gtk+2.18r version or higher to work properly. This library is huge but it is usually included as precompiled packages

in the most popular distributions. Unfortunately, if we want to use gtk+3 and higher versions, we need some hard work (http://developer.gnome.org/gtk3/3.5/gtk-migrating-2-to-3.html). Most of new distributions still maintain both versions, but it may be removed soon as long as get older.

Additional libraries are necessary to detect features and do the matching process. When all images are similar in nature (same scale, orientation, etc) simple corner detectors can work to match features across different images. But when you have images of different scales and rotations, you need to use the Scale Invariant Feature Transform (SIFT). *SiftGPU* is the ChangChang Wu's GPU implementation of SIFT. This algorithm can solve the problem of matching very quickly by simply using standard computers. This algorithm works very well, but the number of known objects is usually small. SiftGPU requires specific hardware, but it is extremely useful.

With a NVIDIA 9800 GT we can obtain interest points and descriptors at 25 fps for 640x480 images, depending on the level of detail in the scene.

Compute Unified Device Architecture (*CUDA*) is a parallel computing architecture developed by Nvidia for graphics processing using graphics processing units (GPUs) that is accessible to software developers through variants of industry standard programming languages. This library is needed by SiftGPU.

The OpenGL Extension Wrangler Library (*GLEW*) is a cross-platform C/C++ library that helps in querying and loading OpenGL extensions. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. Glew and OpenGL library are used by SiftGPU and VisualSFM.

DevIL is a cross-platform image library. Although this is a very powerful library, it is unmaintained since 2010, so it may become impossible to compile in the recent future. Fortunately, it is included in most of modern distributions as a precompiled package, and it can be removed as dependency if we do not want to use SiftGPU for any other purpose than installing VisualSFM.

The emergence of multicore computers represents a fundamental shift, with major implications for the design of computer vision algorithms. Most computers sold today have a multi-core CPU with 2-16 cores and a GPU with anywhere from 4 to 128 cores. Exploiting this hardware parallelism will be the key to the success and scalability of computer vision algorithms in the future. *PBA* implements new inexact Newton type Bundle Adjustment algorithms that exploit hardware parallelism for efficiently solving large scale 3D scene reconstruction problems. In this software the use of multi-core CPU as well as multi-core GPUs are explored.

Graclus is a fast graph clustering software that computes normalized cut and ratio association for a given undirected graph without any eigenvector computation. This is possible because of the mathematical equivalence between general cut or association objectives (including normalized cut and ratio association) and the weighted kernel k-means objective. One important implication of this equivalence is that we can run a k-means type of iterative algorithm to minimize general cut or association objectives. Therefore unlike spectral methods, our algorithm totally avoids time-consuming eigenvector computation. We have embedded the weighted kernel k-means algorithm in a multilevel framework to develop very fast software for graph clustering.

5.1. Instalation of VisualSFM and Bundler in Linux Ubuntu/Debian

In order to install VisualSFM, we firstly need to install a modern distribution of Linux Ubuntu Desktop or Linux Debian. New compilers of these distros (such as g++ 4.6) does not support all the code of VisualSFM and most of the needed packages used by this software, so we have to modify parts of code in order to make it work (it is explained later in this tutorial).

VisualSFM uses the multiple libraries by dynamic loading the modules. That means that it loads needed libraries into memory at run time, retrieving the addresses of functions and variables contained in the library, and then, executing those functions from memory. So the first step may be compile VisualSFM without any additional library but Gtk. We need to install the gtk-devel package, then we can use the make command for VisualSFM. We can define the variable vsfm with the path to the directory we installed VisualSFM:

```
export vsfm=/path_to_vfsm
```

Now we must download and install SiftGPU.

5.1.1. Installing SiftGPU

Before installing SiftGPU we need to install a proper driver for our Nvidia Graphics Card in the system (the process is explained at https://help.ubuntu.com/community/BinaryDriverHowto/Nvidia for Ubuntu, alternatively we can install SIFT, but we may have problems with patents in USA and other countries along the world (http://www.google.com/patents?vid=6711293).

Once the driver has been installed, the next step is to install CUDA. Take into account that CUDA can be installed for 32 and 64 bits Operative Systems. We shouldn't have any problem if drivers are previously installed.

We define the variable CUDA as the path where we installed CUDA in our system:

```
export CUDA_INSTALL_PATH=/path_to_CUDA
```

We can now install the libdevil-devel package or change the code of SiftGPU to place this line in some header:

```
#define SIFTGPU_NO_DEVIL
```

And to remove -lIL in the file named makefile.

Then a simple make command should work with both options. We have to copy now the file libsiftgpu.so as $vsfm/bin/libsiftgpu.so.

We define the path to the Siftgpu code:

```
export SiftGPU=/path_to_SiftGPU
```

5.1.2. Installing PBA

We can download and compile PBA now. Problem is that PBA doesn't compile directly using a modern compiler, so we have to add the following lines in the file named pba.h:

```
// ------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <cstddef>
using namespace std;
// ------------------------------------------------
```

This happens due to the new C++ standard forgives the use of variables, classes and functions without *namespaces*.

If we want to use CUDA, some lines have to be modified according to the architecture of our machine in the file named makefile:

```
CUDA_LIB_PATH = $(CUDA_INSTALL_PATH)/lib64
```

or

```
CUDA_LIB_PATH = $(CUDA_INSTALL_PATH)/lib32
```

then we can use the command:

```
make -f makefile
```

and then we have to copy the file `libpba.so` to `$vsfm/bin/libpba.so`

If we do not want to use CUDA, we must use the command:

```
make -f makefile_no_gpu
```

and then to copy the file `libpba_no_gpu.so` to `$vsfm/bin/libpba.so`

5.1.3. CMVS/PMVS2

CMVS is the implementation of Yasutaka Furukawa's patch-based dense reconstruction algorithms. CVMS depends on three different libraries that are not usually in the repositories of the system: Bundler, PMVS2 and Graclus. We also need the following libraries found in the repositories of our system: libGSL, blas, fortran, libgslcblas, and liblapack.

We also need a package named CLAPACK that usually is not included as precompiled package in our system, so we have to download it and install it. CLAPACK uses cmake to compile, and it should compile without problem with the command:

```
cmake . && make
```

We export the variable with the path where we have installed this library:

```
export CLAPACK=/path_to_CLAPACK
```

After compiling this library we have to copy some files:

```
mkdir \$CLAPACK/INCLUDE 2> /dev/null;
cp \$CLAPACK/INCLUDE/* \$CLAPACK/INCLUDE/clapack/
```

Glaclus needs CLAPACK to compile. We download the library by sending an email to the author and receiving the software by means of this WEB site: http://www.cs.utexas.edu/users/dml/Software/graclus.html

Graclus is a C (not a C++) library, that means that if we try to compile it directly we will have problems and a lot of code to modify in CMVS. To prevent this we can directly compile Graclus (or any other C library) using g++ instead gcc. We can simply use this command in the path of the sources of Graclus:

```
CC=g++ make
```

We export the path to Graclus in a variable:

```
export GRACLUS=path_to_Graclus
```

Bundler should not be difficult to compile. Problems may appear with new compilers, and can be easily solved by adding some lines into the file bundle.cc:

```
// ------------------------------------------------
#include <vector>
#include <algorithm>
#include <numeric>
// ------------------------------------------------
```

Bundle needs the following packages (they are available in the repositories): slibann, minpack, f2c and gfortran.

One problem is that new versions of libraries of libann are lowcase where older versions are uppercase, we have to change the name to the libraries:

```
sudo ln -s /usr/lib/libANN_char.so /usr/lib/libann.so
```

If you are not the administrator of the system or if you want to do an elegant installation, you can always copy the library and redefine `LD_LIBRARY_PATH` to point to the directory where you have copied the file.

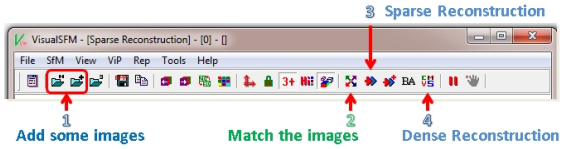

Figure 5: *VisualSFM interface. Numbers refer to steps on the reconstruction process.*

Bundler uses some non-conformant C++ code, what derives into errors by the new compilers. In order to prevent it we have to downgrade the diagnostics by using this command to compile:

```
make  OPTFLAGS='-O3 -Wall -fpermissive'
```

Once we have installed Bundler and Graclus, we can start to install CMVS, first we should define a variable to the path:

```
export CMVS=path_to_CMVS
```

Then we have to do some changes in the CMVS makefile, that is in $CMVS/program/main:

```
#Your INCLUDE path (e.g., -I/usr/include)
YOUR_INCLUDE_PATH = -I$CLAPACK/INCLUDE/ -I$GRACLUS/metisLib2
      / -I$CLAPACK/INCLUDE/clapack/
```

We should not have any problems to compile the library now.

If needed, we can install PMVS2 following the same instructions that for CMVS.

5.1.4. Setting up VisualSFM

In order to make VisualSFM works, we have to create an script that tells VisualSFM where to find the libraries and executables of the previous installed programs and libraries.

```
LANG=EN # Optional, define your language here.
LD_LIB_PATH=:$LD_PATH:$SiftGPU/bin/:$CUDA_INSTALL_PATH/lib/2
PATH=$PATH:./:$SiftGPU/bin/:$CUDA_INSTALL_PATH/bin/
echo $PATH
echo $LD_PATH
LANG=$LANG PATH=$PATH LD_LIBRARY_PATH=$LD_PATH ./VisualSFM 6
```

CMVS is optional, so we can select the executable directly from the interface every time VisualSFM need it.

6. Using VisualSFM

Runing VisualSFM is quite easy, and is well documented on its website [hW11]. A tutorial is also available [Lis].

The reconstruction process can be done following these steps (see figure 5):

1. Load images. You can select the image files or load a text file containing the image list.

Figure 6: *UAV quadrotor*

2. Perform matching. This step detects image features and find feature correspondeces. It generates a graph connecting images with common features. This process is carried out using SIFT or siftGPU.
3. Sparse reconstruction. This process runs bundler and generates an sparse point cloud (see figure 9).
4. Dense reconstruction using Yasutaka Furukawa's CMVS/PMVS. The result is a dense point cloud that can be exported to ply format.

It is possible to iterate steps one and two to add images incrementally. The resulting point cloud can be triangulated by means of MeshLab or any other tool for digitized models [CCD*08].

7. Practical evaluation

We present two real case studies solved using VisualSFM. For the first one we have tested different panoramic images taken with an Unmanaged Aerial Vehicle (UAV) and using VisualSFM to reconstruct the frontage of the building. Our UAV is a quadrotor with an installed panoramic Go-Pro video-camera, made by the Intelligenia Dynamics company (figure 6). The reconstructed frontage is the LRV (Laboratorio de Realidad Virtual) of the University of Granada (figure 7). All photographs were obtained by flying the UAV using different trajectories along the frontage of the building. We have tested a building with very plain areas and with dense textured areas in order to test the capabilities of the different algorithms.

We have tested different number of photographs for similar points of view. The result is different according to the number of images: when the size of the set of images is smaller than 100, the algorithm reconstructs easily textured areas, but it is not possible to obtain any 3D information from plain areas, this is due to the resolution of the camera. As long as we increase the number of photographs from different point views, we obtain better results.

Figure 7: *LRV facade used for testing*

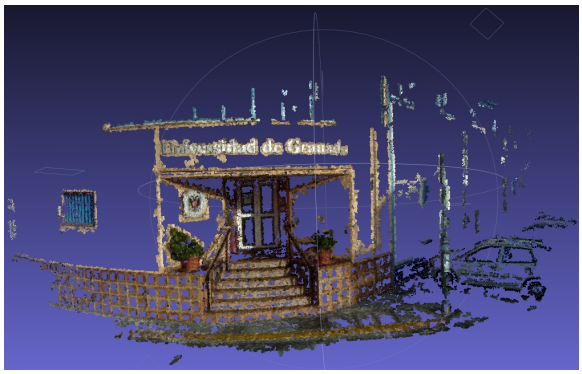

Figure 8: *LRV facade used for testing*

When the set of images is bigger than 200, algorithm of SfM fails. Explanation is that SfM try to match the same pixel in too many images, that results in a failed test and produces an empty match. Result is that only very textured area (such as the title of the University in the building) appears, and the rest is omitted in the model.

In any case, the model has a lot of holes because the object material on the scene (see figure 8). Glasses are difficult to capture due to brightness, as well as flat coloured areas like walls. The reason is that for this type of materials SfM finds too many correspondences in other images, so it is not able to obtain an unique valid match, returning an invalid point.

For the second case of study we have used aerial photographs from Pix4D (`http://pix4d.com/showcase.html`). We have used 36 images from the showcase three (see figure 9). This dataset is a field of 0.7 Km^2 covered with 525 images with an average ground sampling distance of 3.3cm / pixel.

Bundler runs on 30 second and dense reconstruction using CMVS/PMVS needs 45 minutes. Resulting cloud contains 2.8 million points, and it is shown on figure 10. It can be seen that coverage is quite good in most areas.

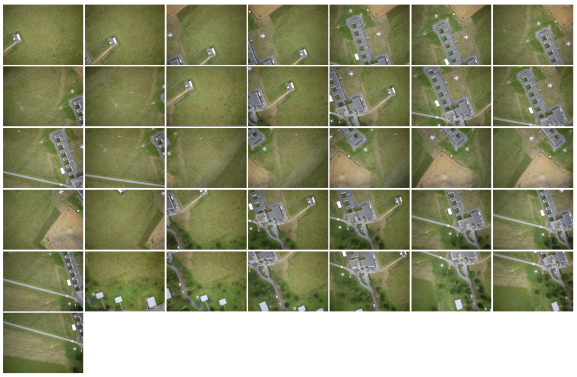

Figure 9: *Aerial photographs used for test case*

Figure 10: *Dense reconstruction from aerial photographs*

8. Conclusions

Structure from motion is a feasible and attractive approach to generate 3D models. It can generate good coverage dense point clouds that can be triangulated using standard software as MeshLab. It is also attractive as it does not need expensive capturing devices.

As conclusion, we can affirm that there is good open source software generating results similar to the best commercial software.

Acknowledgements

This work has been partially funded with FEDER fund by: the Consejería de Innovación Ciencia y Empresa de la Junta de Andalucía under grant PE09-TIC-5276; the Campus de Excelencia Internacional BIOTIC under grant 20F12/39; and the Agencia de Obras Públicas de Andalucía under grant IG-GI3000/IDIC

References

[AM12] AGARWAL S., MIERLE K.: Ceres-solver: A nonlinear least squares minimizer with BSD license, 2012. http://code.google.com/p/ceres-solver. 4

[ASSS10] AGARWAL S., SNAVELY N., SEITZ S., SZELISKI R.: Bundle adjustment in the large. In *Eleventh European Conference on Computer Vision (ECCV 2010)* (2010), pp. 29–42. 4

[BRLT03] BOSSE M., RIKOSKI R., LEONARD J., TELLER S.: Vanishing points and 3D lines from omnidirectional video. *The Visual Computer 19(6): 417-430* (2003). 3

[CCD*08] CIGNONI P., CORSINI M., DELLEPIANE M., RANZUGLIA G., VERGAUVEN M., GOOL L. V.: Meshlab and Arc3D: Photo-reconstruction and processing 3D meshes. In *EPOCH Conference on Open Digital Cultural Heritage Systems* (2008). 8

[CST*08] CHEN C.-I., SARGENT D., TSAI C.-M., WANG Y.-F., KOPPEL D.: Stabilizing stereo correspondence computation using delaunay triangulation and planar homography. In *Advances in Visual Computing*, vol. 5358 of *Lecture Notes in Computer Science*. Springer Berlin, 2008, pp. 836–845. 5

[DSTT00] DELLAERT F., SEITZ S. M., THORPE C. E., THRUN S.: Structure from motion without correspondence. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR* (2000), pp. 557–564. 3

[FLP01] FAUGERAS O., LUONG Q.-T., PAPADOPOULO T.: *The Geometry of Multiple Images*. MIT Press, 2001. 2, 3

[Ho12] HO N.: Structure from motion: RunSFM, 2012. http://nghiaho.com/?page_id=253. 5

[hW11] HANGCHANG WU: Visualsfm - a visual structure from motion system, 2011. http://www.cs.washington.edu/homes/ccwu/vsfm. 8

[HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 3

[IG98] ISHIKAWA H., GEIGER D.: Occlusions, discontinuities, and epipolar lines in stereo. In *Fifth European Conference on Computer Vision (ECCV '98), 2-6 June 1998, Freiburg, Germany* (1998). 4

[IZKB10] IRSCHARA A., ZACH C., KLOPSCHITZ M., BISCHOF H.: Large-scale, dense city reconstruction from user-contributed photos. *Computer Vision and Image Understanding. 16 (1) January 2012, Pages 2-15* (2010). 4

[Kog11] KOGAN D.: libdogleg - a general purpose sparse optimizer to solve data fitting problems, such as sparse bundle adjustment, 2011. http://github.com/Oblong/libdogleg. 4

[LCZ99] LIEBOWITZ D., CRIMINISI A., ZISSERMAN A.: Creating architectural models from images. In *EUROGRAPHICS' 99* (1999). 2

[Lev44] LEVENBERG K.: A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics 2: 164–168* (1944). 4

[Lis] LISCIO E.: Open source tools for 3D forensic reconstructions. part 3. 8

[LM04] LU X., MANDUCHI R.: Wide baseline feature matching using the cross–epipolar ordering constraint. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR, pp. 16-23,* (2004). 4

[Lou] LOURAKIS M.: A generic sparse bundle adjustment C/C++ package based on the levenberg–marquardt algorithm. http://www.ics.forth.gr/~lourakis/sba. 4

[Low99] LOWE D. G.: Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157* (1999). 3

[Mac] MACH L.: insight3d. http://insight3d.sourceforge.net. 5

[MSK*03] MA Y., SASTRY S. S., KOSECKA J., SOATTO S., KOSECKA J.: *An Invitation to 3-D Vision: From Images to Geometric Models*, vol. 26 of *Interdisciplinary Applied Mathematics Series*. Springer-Verlag New York, LLC, 2003. 2, 3

[NK11] NEITZEL F., KLONOWSKI J.: Mobile 3D mapping with a low-cost UAV system. In *Conference on Unmanned Aerial Vehicle in Geomatics* (2011), International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 5

[RG04] RAMÍREZ A., GONZÁLEZ S.: Modelado a partir de fotografías. proyecto fin de carrera. Ing. Informática. Univ. de Granada, 2004. 2

[RHFL10] REICHELT S., HÄUSSLER R., FÜTTERER G., LEISTER N.: Depth cues in human visual perception and their realization in 3D displays. In *Three Dimensional Imaging Visualization, and Display* (2010). 1

[SS01] SHAPIRO L. G., STOCKMAN G. C.: *Computer Vision*. Prentice Hall, 2001. 2

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring image collections in 3D. *ACM Transactions on Graphics* (2006). 4

[SSS07] SNAVELY N., SEITZ S. M., SZELISKI R.: Modeling the world from internet photo collections. *International Journal of Computer Vision* (2007). 4

[THWS10] THORMÄHLEN T., HASLER N., WAND M., SEIDEL H.-P.: Registration of sub-sequence and multi-camera reconstructions for camera motion estimation. *Journal of Virtual Reality and Broadcasting, 7 (2)* (2010). 4

[UHS07] UNNO H., HAYASHIBE K., SAJI H.: Extraction of corresponding points from stereo images by using intersections of segments. In *MVA2007 IAPR Conference on Machine Vision Applications, Tokyo, JAPAN* (2007). 3

[VG06] VERGAUWEN M., GOOL L. V.: Web-based 3D reconstruction service. *Mach. Vision Appl. 17*, 6 (2006), 411–426. 5

[Wan11] WANG Y.-F.: *A Comparison Study of Five 3D Modeling Systems Based on the SfM Principles. Technical Report, TR 2011-01*. Tech. rep., Visualsize Inc., 2011. 5

[Wu11] WU C.: Multi-core bundle adjustment (CPU/GPU), 2011. http://grail.cs.washington.edu/projects/mcba. 4

[Zac11] ZACH C.: Simple sparse bundle adjustment (SSBA), 2011. http://www.inf.ethz.ch/personal/chzach/opensource.html. 4

[ZTCS99] ZHANG H., TSAI P., CRYER J. E., SHAH M.: Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 8 (1999), 690–706. 3