Martina Ballarin[*], Paolo Vernier[**]

Digitization of Maps Belonging to Cultural Heritage

Keywords: historical maps; 3D recording; photogrammetry; Arduino

Summary: Historical maps are an essential documentary heritage for the knowledge of territory. Digitization, in the case of old maps, cannot be a simple high-resolution two-dimensional scan, but it has to detect the geometry of the supporting material.
This work describes part of an on-going project, developed by the Laboratory of Photogrammetry of IUAV University, concerning the study of different methodologies and tools for the 3D digitization of old maps. In particular, high-resolution cameras, combined with a mechanical positioning system and appropriate software for the rigorous construction of "mosaics", will be explored as a potential means of replacing the expensive large format flatbed scanners. The research aims to digitally record the shape of maps, not only considering their semantic content (printed image), but also their three-dimensional surfaces, and their subjection to time and deformations. Through the development of new algorithms, procedures and mechanical properties, it will be possible to obtain a rigorous three-dimensional geometric memory of the original cartographic heritage.

Introduction

Increasing technical demands have stimulated the development of methods and instruments for a better and fuller use of cartography. At the same time, multi-media technology has demonstrated its importance in the exploitation of our cultural and environmental heritage. The ever-greater proliferation of computers and the digital visualisation of data make it possible to develop models that enable us to define just how much has been achieved by research into or intervention upon that heritage.
These new technologies have also led to a development in the way antique cartography is made available, offering immediate interfaces with the object of study and a wide variety of functions that are much simpler to operate than those in earlier computer technology.
All ancient cartographic documents, today preserved in many institutions, are a great historical and cultural heritage. Unfortunately, the aging of the analogic support often causes many problems in the preservation of ancient maps. "Recovery of historical maps" is the whole amount of operations that is possible to perform in order to preserve the cartographic document, meant both as its analogue support and as its content. In particular, modern geomatic techniques allow the preservation of the content by means of metric recovery in digital form and digital processing of historical cartography. Moreover, these digital processes give us new chances of using historical information, which would be unachievable on analogue supports.
However, the digitization of historical maps raises many issues, especially because the map has three different contents (semantic, symbolic and metric) that need to be taken into account and well preserved during the whole process. Too often the digitization of maps concentrates more on their textual and pictorial content, neglecting their geometric value.

[*] Laboratorio di Cartografia e Gis, Università IUAV di Venezia, Venice, Italy [martinaballarin@hotmail.it]
[**] Dr., Laboratorio di Cartografia e Gis, Università IUAV di Venezia, Italy [vernier@iuav.it]

Recently, several companies have started producing high quality scanners for the digitization of ancient maps in atlases or old books. These machines guarantee a good accuracy both from the radiometric and the geometric points of view, but nonetheless their cost is still prohibitive for the majority of public and private institutions.

Several studies have been made over the years to obtain a digitization process that could also register in digital shape both the three-dimensional surface of the supporting material (e.g. an undulated / deformed map surface) and its two-dimensional high-resolution image. The systems presented aimed to acquire an accurate memory of the map as it was, and then to correct its deformation on the digital copy, bringing it back to the initial shape without damaging the original map.

An interesting study concerned the use of triangulation-based laser scanners for the acquisition of three-dimensional data as point clouds, with an uncertainty of about 0.1 mm. These metrical data were associated to their radiometric values acquired by high-resolution cameras thanks to an external reference system. This was materialised as a grid of targets placed over the map, so that it would not touch its fragile surface (Adami et al, 2007).

Other applications used static systems for the acquisition of high-resolution stereo images in order to obtain two and three-dimensional products. The system could move along the X and Z axes, so that they could obtain images with a good overlap and suitable for the 1:1 scale, as the distance between the camera and the map could be changed according to the focal length of the camera used (Tsioukas et al, 2009). The same research group underlined the value of *Structure from Motion* software products for the digitization of old maps in atlases and bounded books, in comparison with dedicated book scanning devices (Tsioukas et al, 2012).

The solution proposed here intends to learn from these studies and automate the process of image acquisition. In this way, we can obtain sequences of oriented images, as for each one the rotation angles of the camera (ω, ϕ, κ) and the coordinates of its projection centres (X_0, Y_0, Z_0) are known. These images can be processed with *Structure from Motion* software products in a semi-automatic way or with more traditional photogrammetric software products. The results of the process will not just be two-dimensional images, but we will obtain three-dimensional surfaces that will enable us to analyse the deformation of the maps over time.

The system

The system created is a numerically controlled machine composed of two main parts: a high-resolution camera for the acquisition of images and a mechanical structure for its movement along the two axes (X, Y). The mechanical part is composed of a metallic structure with a camera mount, an open-source platform (Arduino UNO), two stepper motors for the movement of the camera mount along the two axes and a servomotor for the image acquisition.

Arduino

First of all, identifying a platform able to control the mechanism and receive the instructions given by the operators was a very important step. The system we wanted to realise had to be adaptable, to respond to different applications' needs: it had to be able to receive basic information given by the operator and to properly transform it to obtain the practical result desired.

During the past decades, technology has exponentially evolved, allowing men to reach amazing goals. The decreasing of production costs allowed everyone to approach robotics, with interesting results both in terms of quantity, and in terms of quality and variety of products realised by non-

experts. The “Arduino project” is the most renowned example of this process, because it is low-cost, completely open-source and user-friendly: even users without any experience on electronics and programming can approach it.
The project is composed of three parts: the hardware (the platform with an Atmel microcontroller), the software (IDE, Integrated Development Environment) and, most importantly, the community of users.
The hardware platform is based on a printed circuit board, which integrates a microcontroller with PIN connected to I/O serial ports, a voltage regulator and a USB interface that enables communication with the computer. The software is an integrated development environment (IDE) available on different platforms (Linux, Apple Macintosh and Windows). This software allows even beginners to program in a simple and intuitive language derived from C and C++ called Wiring, which is freely downloadable and editable (URL 1). Finally, the community is very vast, mainly for the reasons described before: Arduino is economic; it is open-source (not only the hardware and software information, but also the projects developed by users are available for anyone); it is easy to use and it can be connected to almost every electronic device, which allows it to be used in every sort of situation (Fig. 1).

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

The Arduino Playground

Welcome to the Arduino Playground, a wiki where all the users of Arduino can contribute and benefit from their collective research.

This is the place to post and share your own code, circuit diagrams, tutorials, DIY instructions, tips and tricks, and after all the hard work, to show off your projects! Anyone can edit and add to the pages here.

Arduino Playground is a work in progress. We can use all the help you can give, so please read the Participate section and get your fingers typing!

NOTE (20081002): Solved the problem for adding content in non-western Latin languages, now it is possible to start typing in any language on the internet test here

Figure 1: The “Arduino project”: hardware, software, community.

The only two limits of Arduino are its physical structure on one side – there are a limited number of ports for input and output data – and the amount of memory available to store the program (32Kb). The user has to carefully evaluate the length of the program and the number of ports considering the purposes of the project.

Motors

In order to be able to properly plan the acquisition process and to simplify the post-processing phase, it is necessary to track the paths the robot follows and to determine the sensor’s exact position at the time of the acquisition. This is how we guarantee the automatic reconstruction of the digital data in a single reference system. Since these movements are very small, we need to reach high precisions. Therefore, the only useful method to track the position of the system is to record the motors movements.
We decided to use stepper motors, because they guarantee a precise control of their rotation: they can rotate, on average, between 0.9 and 1.8 degrees at every step in both directions and they can reach high speed. The linear movement corresponding to every angular step is declared on the tech sheet of every motor. Thanks to this system, in order to know the exact camera positions we just need to multiply this movement to the number of rotations assigned before every acquisition: the position tracking is ensured by the motor itself. For this project we used two “Mercury Motor”

SM-42BYG011-25, bipolar motors that have a step angle of 1.8 degrees, a rated voltage of 12V and a rated current of 0.33A.

Arduino cannot control these two motors by itself, it needs extra hardware to manage the high current necessary to move the stepper motors: a motor shield has to be connected to the main board. The shield chosen for this project is a board with three drivers that can control three bipolar stepper motors, governing their speed and direction. For this project we just needed two motors, but we wanted to keep open the option of integrating the system with another movement along the Z axis, to modify the distance between the camera and the map. Moreover, the shield gives the possibility of reducing the single step of each motor to 1/2, 1/4, 1/8, up to 1/16, in order to control smaller movement and increase the accuracy of the system (Fig 2).

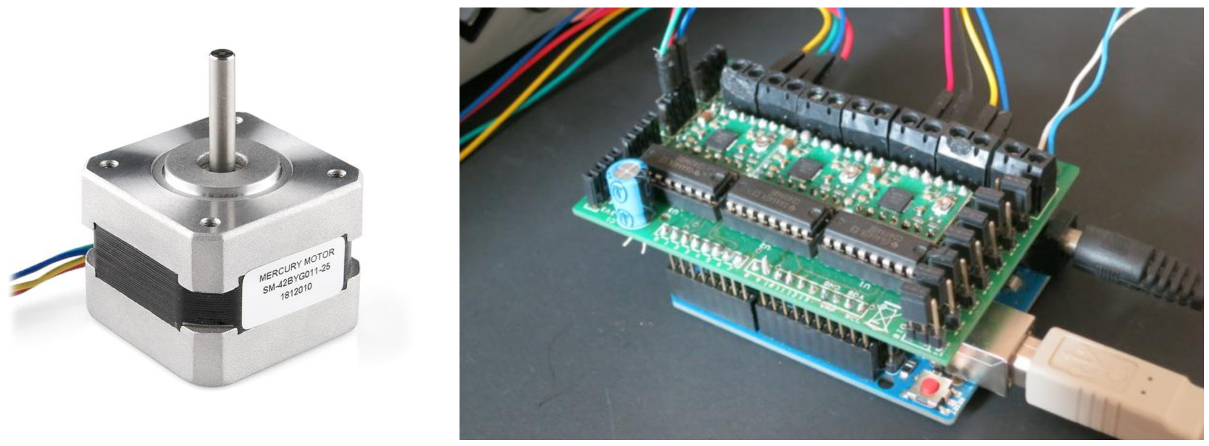


Figure 2: The stepper motor and the motorshield connected to Arduino.

As stated before, the machine also uses a servomotor attached to the camera to push the shutter button and take the pictures. These motors can accurately control a movement, because generally they move between two fixed positions (usually 0° and 180°), instead of continuously rotating. Moreover, they are easy to connect and control because they have a dedicated “library”, a specific program – already written – that provides extra functionality and that the user can call when it is needed. The servomotor used in this project is a GoTeck RC V2 Micro Servo (GS-9018) that has a rated voltage between 4.8V and 6V, so it can be powered directly by Arduino (Fig. 3).

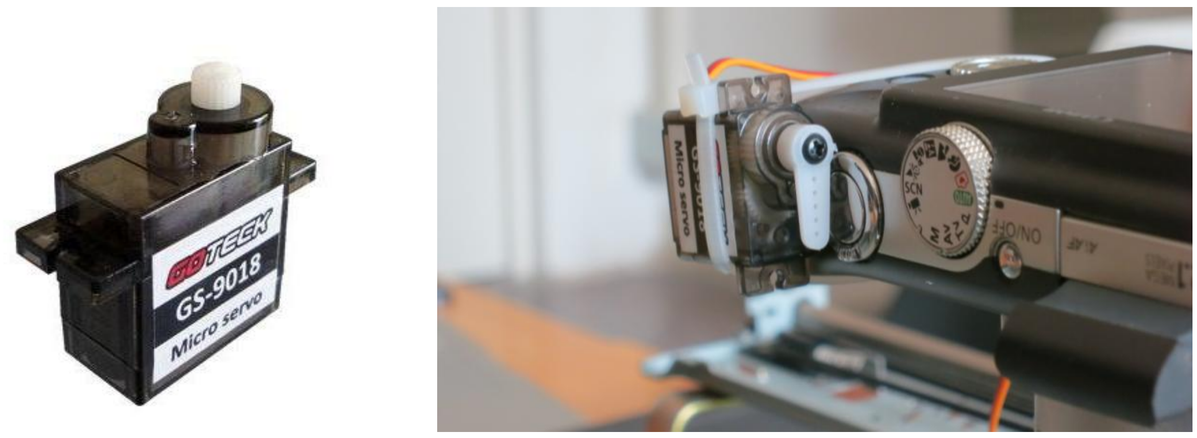


Figure 3: The servo motor connected to the camera.

Cameras

In order to obtain raster data suitable for a 2D and 3D high-resolution reproduction of historical maps, we will need to use last generation digital reflex cameras with some fundamental characteristics, such as a full frame sensor (36x24 mm) and high-resolution photos. Moreover, we will need to use a good calibrated lens, of which focal length, position of the principal point and parameters for the correction of distortions are known. Software products for camera calibration, and also the digital photogrammetry software used for the first tests, automatically compute all the necessary information for the inner orientation of the camera.

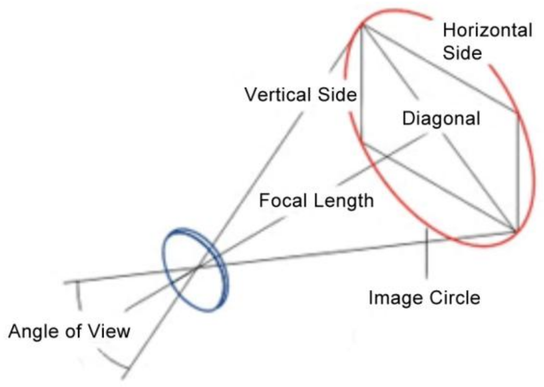


Figure 4: The angle of view depends on the focal length and the sensor dimension.

The sensor dimension allows us to use the angle of view of the camera lens without any data loss (Fig. 4). In fact, the same focal length can have different angles of view if it is used together with sensors of different dimensions. For example, a 20 mm camera lens with a 36x24 mm has an angle of view of 94°; on a 23,7x15,6 mm its angle of view will be 70° (URL 2).
The table below (Table 1) shows different examples of combinations of sensors and camera lenses.

Nikkor Camera lens	ANGLE OF VIEW		Equivalent focal length. Format 135-35 mm
	SRL 35 mm (24x36)	D100 (15,6x23,7)	
	Diagonal 43,27 mm	Diagonal 28,37 mm	
AF 16 f/2.8D Fisheye	180°	107°	16 (non fisheye)
AF 14 f/2.8D ED	114°	90°	21
AF 18 f/2.8D	100°	76°	28
AF 20 f/2.8D	94°	70°	30
AF-S 17-35 f/2.8D IF-ED	104°-62°	77°-44°	26-53
AF 60 f/2.8D	39°40′	26°30′	92
AF 105 f/2.8D	23°20′	15°20′	160
AF 200 f/4D IF-ED	12°20′	8°	305
AF 70-180 f/4.5-5.6D ED	34°20′-13°40′	22°50′-9°	107-275

Table 1: Angles of view.

Program

The program we wrote on the Arduino software (IDE) allows the movement of the system just described. Its logic is quite simple. The camera mount has to move at a number of steps defined by the user. Therefore, the program has to tell the X motor to move at that specific number of steps along the X axis, then it has to stop and tell the servomotor to click the shutter button and take the picture. This process is repeated *n* times, according to the number of pictures we want to acquire along that axis. Once it has finished the first *n* movements, the system moves along the Y axis, going on until it completes the cycle defined by the user. As shown in the scheme below, in order to simplify the movements and speed the acquisition process, the camera always moves forward: it first moves to the right along X, then it moves forward along Y and then it goes back to the left along X again, going on accordingly to the dimension of the map that needs to be digitized (Fig. 5).

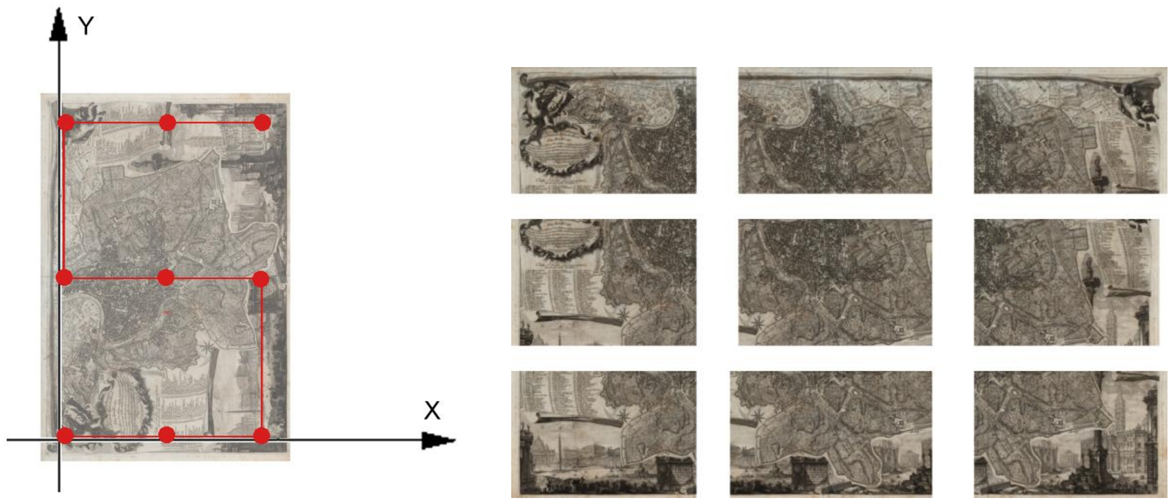


Figure 5: Scheme of a possible acquisition process and images acquired.

Moreover, Arduino gives back the X, Y coordinates of every camera position when each picture is taken. As stated, this is quite easy, as we know the linear movement for every single step taken – thanks to the technical sheet of each motor – and Arduino just needs to multiply it to the number of steps assigned for the movement between each picture.
The logical scheme of the program we wrote is quite simple:
- first of all, there is the declaration of the variables and the connection between motors and pin ports on Arduino;
- then, the “void set up” initializes the motors and the motor shield and sets the pin ports on Arduino as output ports;
- finally, there is the main body of the programme, the acquisition cycle, which goes on loop after the set up. It is divided into two parts that work in the same way: at the beginning the system takes the picture, than it moves and then it computes the X, Y coordinates.

Preliminary results

The system realised is still an on-going project. At present, we have built a prototype version using pieces of an A3 scanner and an A4 printer soldered together in order to allow the movement along the two axes. This allowed us to check if the system worked and to run the first tests. We used three levels to ensure the orthogonality between the optical axis of the camera and the platform on which the map lies and we performed tests with different settings and configurations (Fig. 6). The camera used was a Canon Powershot SX200 IS, because the structure realised cannot lift a reflex camera yet. Anyway, it is a good compact camera, with characteristics more than suitable for this first stage. The camera has a 6.17x4.55 mm CCD sensor that allows us to obtain 12 Mpx images (4000x3000) and a 12x (28-336 mm) optical zoom. Controlling the focus and fixing the zoom at a given length, we obtained good images for the tests.

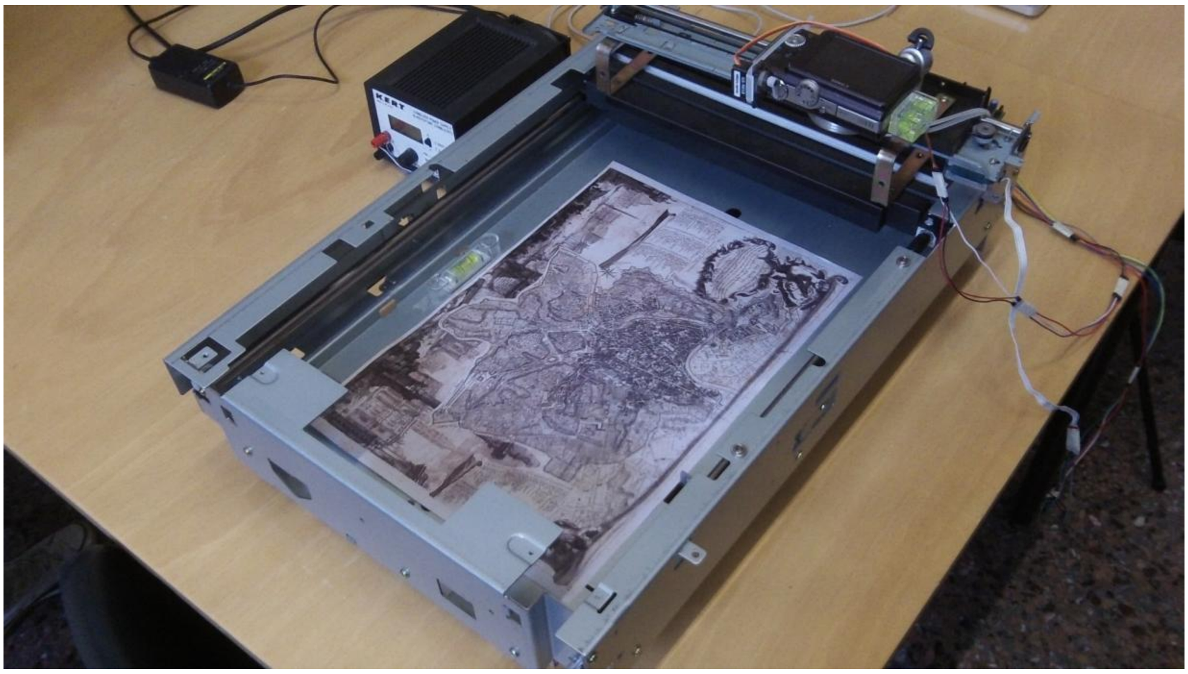


Figure 6: The prototype realized.

From the first test we were able to acquire twelve pictures, because we planned two movements along the X axis and three along the Y axis. For each movement on X we assigned 500 steps; knowing that the motor has an angular step of 0.217mm, the distance between each projection centre was 10.8 cm. On Y, we divided the length into three movements of 2800 steps each. Considering that the angular step of that motor is 0.0319 mm, the distance between each photogram on Y was 8.9 cm.
The photograms were imported and processed with Photoscan Agisoft (Fig. 7), a *Structure from Motion* software product, for the first results analysis (URL 3). Through this method, a texturized mesh of more than 5.000.000 polygons was created (Fig. 8) from which we could export both a surface model and a point cloud. On the point cloud we performed the first analysis on the map deformation: on the picture below we can see a depth map, on which deformations of some millimetres are clearly visible (Fig. 9).

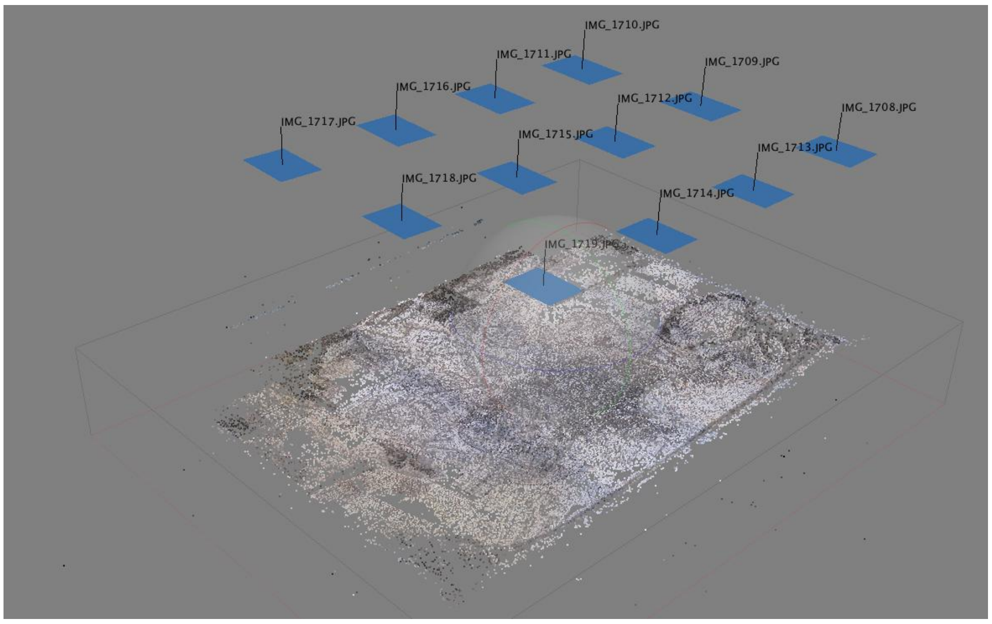


Figure 7: Alignment of the images in Photoscan Agisoft.

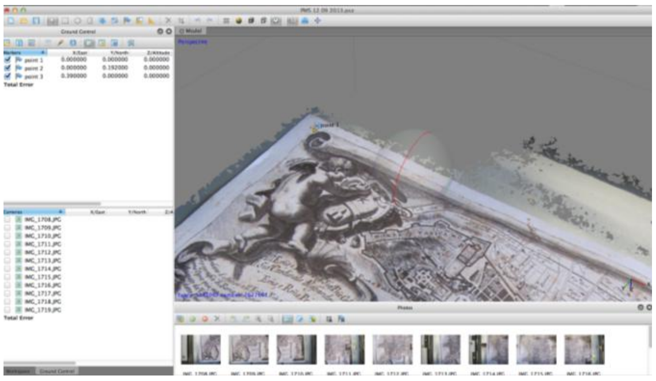


Figure 8: The texturized mesh obtained from PhotoScan Agisoft.

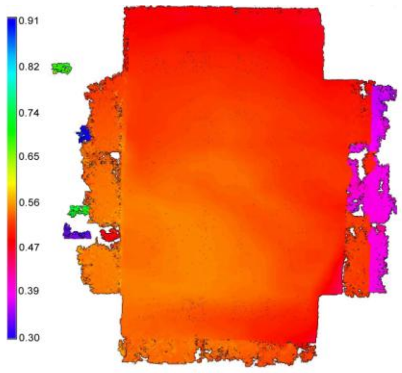


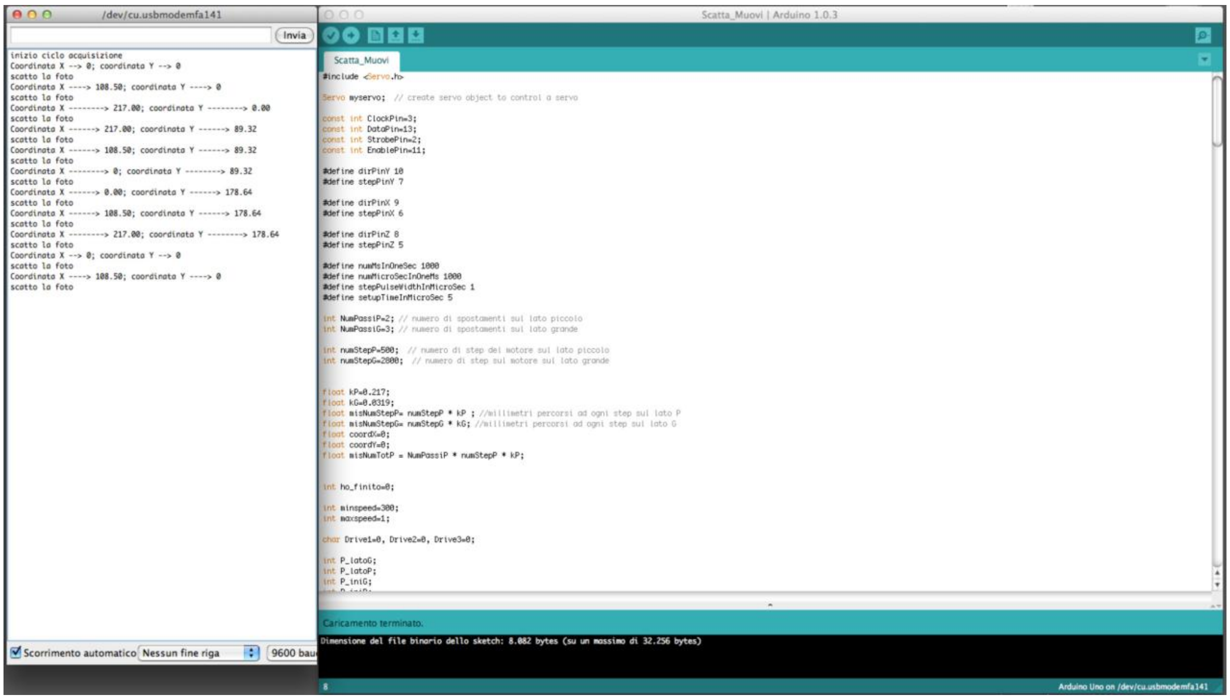
Figure 9: The depth map realized in Pointools.

From Photoscan we also exported an orthophoto, which combines the accuracy of the photo-grammetric survey with the quality of the radiometric value obtained from the same high-resolution pictures. As the distance between camera and map was small (more or less 0.20 m), the final orthophoto could be exported with a pixels size of 0.00006 m even if the focal length was just 5 mm.

Knowing the pixel coverage on the historical map is fundamental for the nominal scale that we want to obtain (Tsioukas et al, 2006). In traditional photogrammetry a lot of attention has been put on the film dimensions and its possible stretches in order to calculate the restitution scale, which is also related to the scale of the photogram and the distance from the object to be surveyed. In digital photogrammetry, pixels are the fundamental unity and therefore we need to take into account how much of the object is covered by a single photodiode of the sensor, together with the already mentioned dimension of the sensor itself, the number of pixels for each single image and the distance between camera and historical map. However, in the digitization of old maps we could live aside the concept of "representation scale", as sometimes there could be the need of exceeding the real dimensions of the map to reproduce it in a larger format and visualise the details without loss of quality and information.

We are now performing other tests to compare the coordinates given by Arduino with those calculated by the software (Fig. 10). The first results are encouraging and would allow us to reach our goal: obtaining series of oriented images that could be mosaicked without the need of identifying any reference point. Moreover, the acquisition of photograms following a regular grid would simplify the matching phase. In fact, when two images geometrically differ only by a translation, a cross-correlation coefficient is enough to compute the correspondences between the two images. Otherwise, the second image would need to be properly transformed (through a transformation comparable to the affine) in order to be similar to the first one. In this case, the system would not be linear and we would have to use a least squares matching.

The mosaic of high-resolution images could be used for archiving, divulgation and research purposes (i.e. to study the transformation of a territory over time). Moreover, as shown, through this method we can already create 3D maps that could be used for different purposes: to better understand and analyse deformations; to restore original geometry and maybe even to create 3D models through solid printing, allowing for example the visually impaired to approach the third dimension of cartographic heritage documents.



```
inizio ciclo acquisizione
Coordinata X --> 0; coordinata Y --> 0
scatto la foto
Coordinata X ----> 108.50; coordinata Y ----> 0
scatto la foto
Coordinata X --------> 217.00; coordinata Y --------> 0.00
scatto la foto
Coordinata X ------> 217.00; coordinata Y ------> 89.32
scatto la foto
Coordinata X ------> 108.50; coordinata Y ------> 89.32
scatto la foto
Coordinata X --------> 0; coordinata Y --------> 89.32
scatto la foto
Coordinata X ------> 0.00; coordinata Y ------> 178.64
scatto la foto
Coordinata X ------> 108.50; coordinata Y ------> 178.64
scatto la foto
Coordinata X --------> 217.00; coordinata Y --------> 178.64
scatto la foto
Coordinata X --> 0; coordinata Y --> 0
scatto la foto
Coordinata X ----> 108.50; coordinata Y ----> 0
scatto la foto
```

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

const int ClockPin=3;
const int DataPin=13;
const int StrobePin=2;
const int EnablePin=11;

#define dirPinY 10
#define stepPinY 7

#define dirPinX 9
#define stepPinX 6

#define dirPinZ 8
#define stepPinZ 5

#define numMsInOneSec 1000
#define numMicroSecInOneMs 1000
#define stepPulseWidthInMicroSec 1
#define setupTimeInMicroSec 5

int NumPassiP=2; // numero di spostamenti sul lato piccolo
int NumPassiG=3; // numero di spostamenti sul lato grande

int numStepP=500;  // numero di step del motore sul lato piccolo
int numStepG=2800;  // numero di step sul motore sul lato grande


float kP=0.217;
float kG=0.0319;
float misNumStepP= numStepP * kP ; //millimetri percorsi ad ogni step sul lato P
float misNumStepG= numStepG * kG; //millimetri percorsi ad ogni step sul lato G
float coordX=0;
float coordY=0;
float misNumTotP = NumPassiP * numStepP * kP;


int ho_finito=0;

int minspeed=300;
int maxspeed=1;

char Drive1=0, Drive2=0, Drive3=0;

int P_latoG;
int P_latoP;
int P_iniG;
```

```
Caricamento terminato.
Dimensione del file binario dello sketch: 8.082 bytes (su un massimo di 32.256 bytes)
```

Figure 10: Example of coordinates computed by Arduino.

Conclusions and future perspectives

The results reached until now are very encouraging, because they allowed us to obtain an accurate final product in an almost automatic way and through a user-friendly and low cost tool. However, the project is far from its conclusion. In the near future, we are going to build a proper CNC machine, with pieces made and designed for these purposes. This will allow us to use heavier cameras (i.e. reflex), but also to obtain a more reliable structure (i.e. a structure with axes that are surely orthogonal).

Moreover, other tests will be fundamental: not just on the comparison between the accuracy of the coordinates given by Arduino and those calculated by the software, but also on the comparison between this system and other data acquisition techniques. We will use triangulation-based laser scanners on the same maps digitized by our system and different high-resolution cameras with different optical lenses. These cameras will be both calibrated and not and we will use other software product normally used in digital photogrammetry, such as Photomodeler or Image Modeler. These comparisons will help us to check the accuracy of the system proposed, but also to better understand the best procedures for digitizing historical maps in different contexts and for different purposes.

References

Adami A., Fregonese L., Guerra F., Livieratos L., Tsioukas V., (2007). Digital Representations And Analysis Of Deformations Induced In Map Supporting Materials, *Proc. of CIPA-2007 International Symposium*, Athens, Greece, Oct. 1-6. 2007

Tsioukas V., Koussoulakou A., Pazarli M., Ploutoglou N., Daniil M., Stergiopoulou I. (2012). Scanning or digitizing in libraries? A test on the efficiency of dedicated book-scanning devices in digitizing bound atlases and maps, *e-Perimetron, Vol. 7, No. 4*: 163-169.

Tsioukas V., Daniil M., 2009. 3D digitization of historical maps, *e-Perimetron, Vol. 4, No. 1*: 45-52.

Tsioukas V., Daniil M., 2006. Possibilities and problems in close range non-contact 1:1 digitization of antique maps, *e-Perimetron, Vol. 1, No. 3*: 230-238.

URL1: http://www.arduino.cc

URL2: http://www.dpreview.com

URL 3: http://www.agisoft.ru