3D laser scanner

Contents:

3D laser scanner recipe.

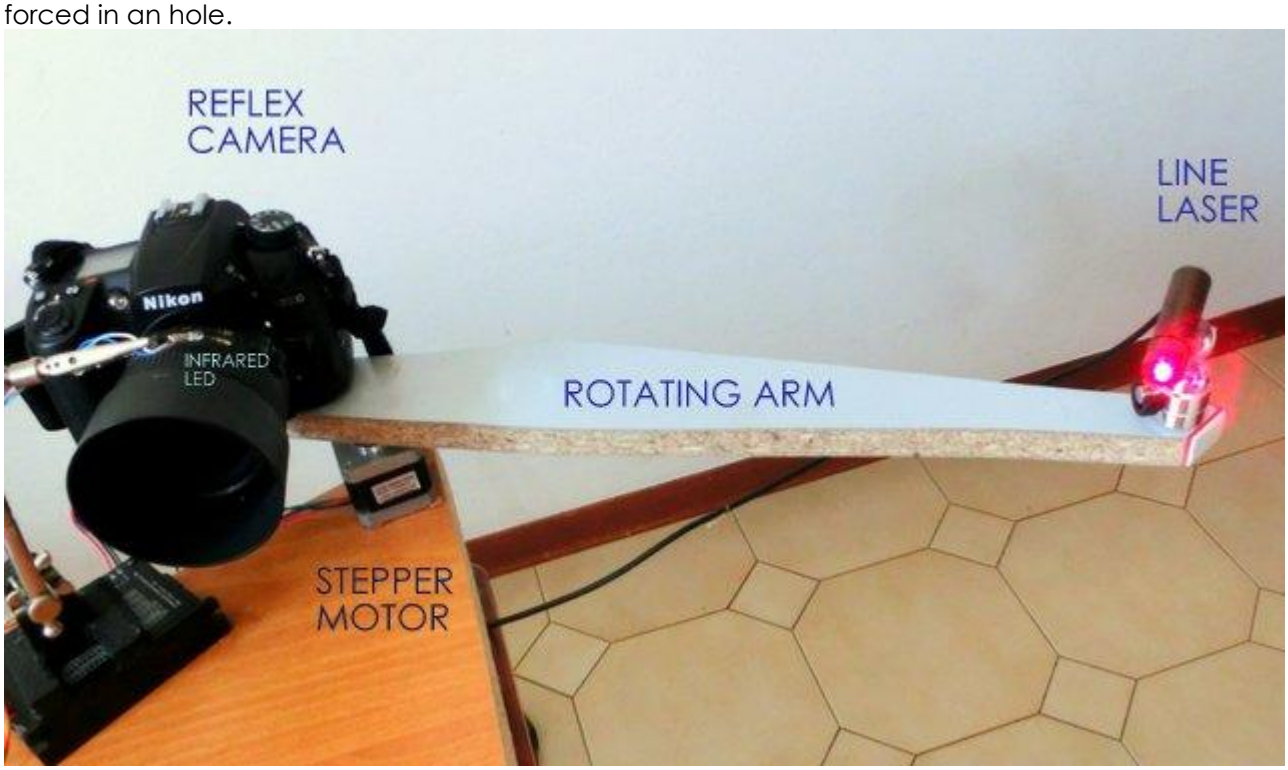
ingredients:

- 1 Line laser (ebay)
- 1 Arduino (ebay)
- 1 Stepper Motor (ebay longs-motor)
- 1 Stepper motor driver (up to 12800 steps)(ebay longs-motor)
- 1 12V power supply (from a chinese DVD player)
- 1 Infrared LED (from a remote control)
- 1 Digital SLR (Nikon D7000, Nikon SLR or a infrared remotely controlled SRL)
- 1 Piece of wood (plywood, plexiglass, metal, or whatever you prefer)
- 1 Ball head (get from a 5€ tripod bought from Chinese)
- Matalb (or Octave)

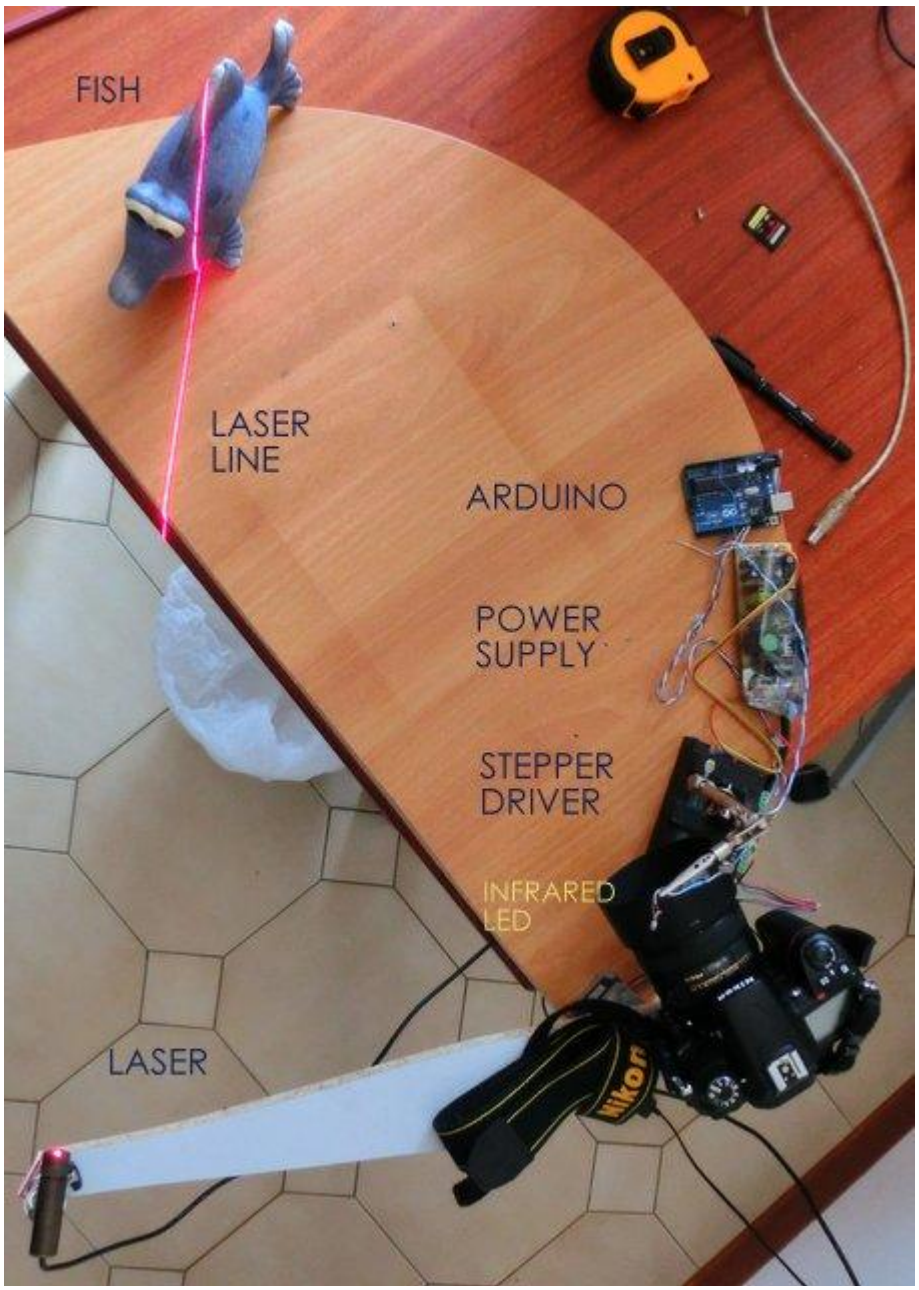
A line laser is a laser equipped with a cylindrical lens, in this case I used a red laser, but other colors can be used, like green or blue. The lens opens the beam shaping it into a plane. The one I used in this experiment is characterized by a nominal power of 100mW (not up to standard in Italy, I think the max allowed power is 5mW) but the beam does not remain focused in a single ray, it is extended in a plane, so the intensity quickly drops below dangerous levels.
The principle behind this scanner is the typical of a line scanner. A laser beam intercepts the object to be measured and a camera, positioned at a known angle and distance shoots a series of images. With some trigonometry considerations and optic laws it is relatively easy to reconstruct the Zeta dimension, the measurement of the distance between the object and the camera.
Commercial systems based on this principle, named line scanner or profilometers, are used in industries and are typically characterized by a high frame rate (even 200 or more frames per second) and a strength suitable for industrial uses. An industrial profilometer price ranges from a few thousand Euros to several tens of thousands of Euros. The cost of the components for the system I propose here does not exceed 100/150 €, if you already have an SLR.

Step 1: Holder arm for laser and SRL.

Using hot glue, welding or the method you prefer, connect the camera and the laser ball head at an arm whose length is about 40/50 cm. When the three parts are connected (laser head, arm and camera) locate the assembly center of gravity (approximately). In that position drill an hole for the stepper motor shaft. It would be geometrically easier to use a panoramic head, which grants the ability to turn the system around the lens nodal point, but that would require a more rigid system, certainly not achievable with a simple piece of chipboard, and a shaft simply forced in an hole.



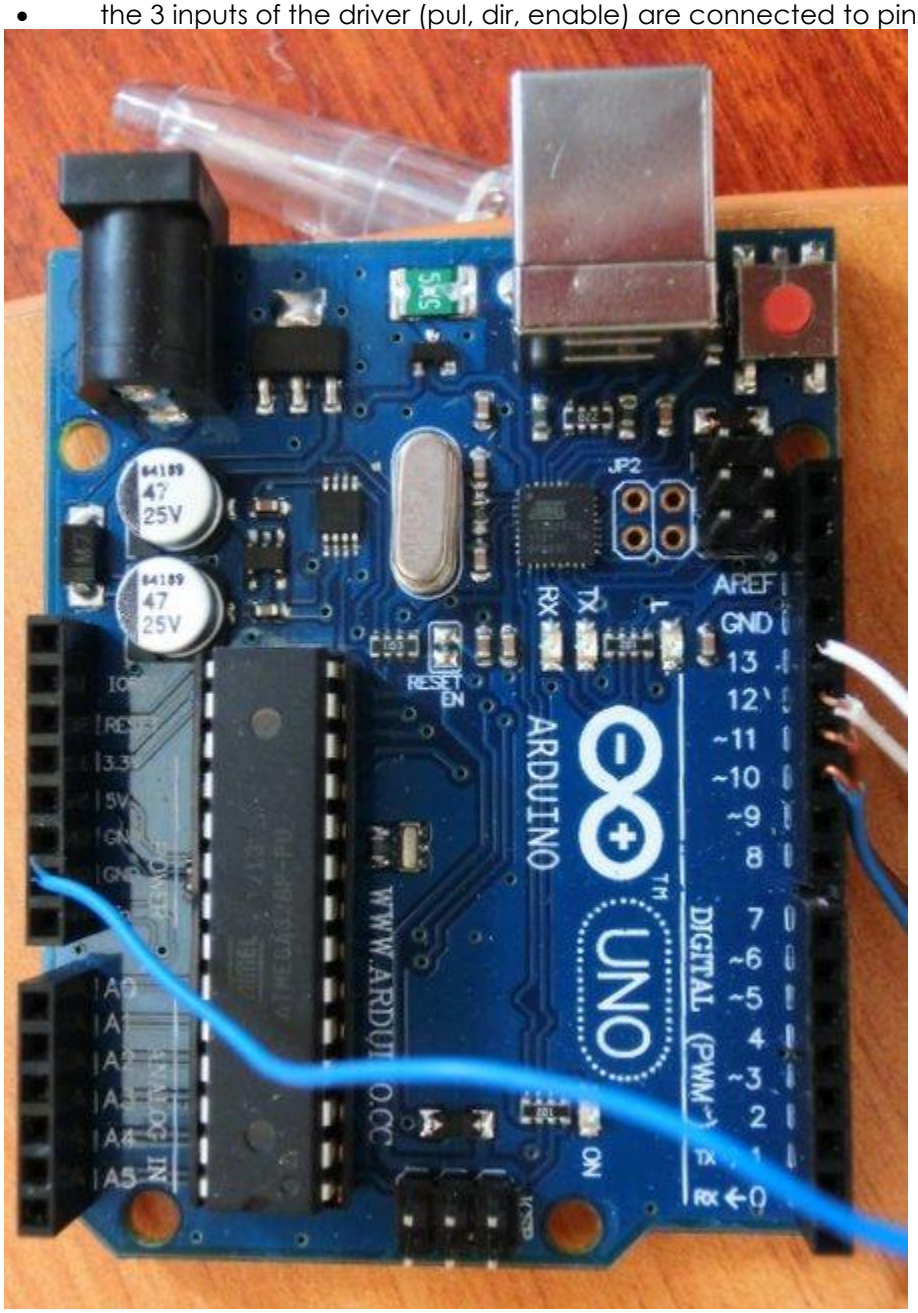
Pay attention to obtain the parallelism between the hole axis, the vertical axis of the image captured by the camera and the laser plane! I suggest you make the hole that houses the motor shaft using a drill press, or, if you have access to a lathe, you could realize a collar to ensures the orthogonality between the motor axis and the arm. Moreover, the laser plane will be, in turn, perpendicular to the plane of the arm and the camera must be perfectly horizontal and leveled. Assembling everything you should get something that looks like the system I show in the picture. I'm waiting for pictures of your own solutions! mail me at scanner@metaingegneria.com


FISH
LASER
LINE
ARDUINO
POWER
SUPPLY
STEPPER
DRIVER
INFRARED
LED
LASER
Nikon

Step 2: Rotation and camera synchronization

The second step is to put everything in rotation and to synchronize the camera. The fastest and easiest way I've found to achieve this synchronization is to use an Arduino to mimic the infrared remote control and to control the stepper motor.

- the infrared led is connected to pin 13 (and ground)
- the 3 inputs of the driver (pul, dir, enable) are connected to pins 10,11 and 12



the Arduino code I used is:

```
/*
Stepper Motor Control +camera control
*/

int pinIRLED = 13;  // infrared led
int pul = 10;
int dir = 11;
int enbl = 12;
int passi = 4; // numero passi su 12800
int intervallo = 3; // intervallo tra i passi

void setup() {
pinMode(pinIRLED, OUTPUT);
pinMode(pul, OUTPUT);
pinMode(dir, OUTPUT);
pinMode(enbl, OUTPUT);
digitalWrite(enbl, HIGH);
}

 void pulseON(int pulseTime) {
unsigned long endPulse = micros() + pulseTime;       // create the microseconds to pulse for
while( micros() < endPulse) {
digitalWrite(pinIRLED, HIGH);                  // turn IR on
delayMicroseconds(13);                         // half the clock cycle for 38Khz (26.32Ã—10-6s) – e.g.
the 'on' part of our wave
digitalWrite(pinIRLED, LOW);                   // turn IR off
delayMicroseconds(13);                         // delay for the other half of the cycle to generate
wave/ oscillation
}
}

 void pulseOFF(unsigned long startDelay) {
unsigned long endDelay = micros() + startDelay;      // create the microseconds to delay for
while(micros() < endDelay);
}

void takePicture() {
for (int i=0; i < 2; i++) {
pulseON(2000);                          // pulse for 2000 uS (Microseconds)
pulseOFF(27850);                        // turn pulse off for 27850 us
pulseON(390);                          // and so on
pulseOFF(1580);
pulseON(410);
pulseOFF(3580);
pulseON(400);
pulseOFF(63200);
```

```
}                                   // loop the signal twice.
}

void loop() {
// ciclo per dare gli step al driver dello stepper
for (int ii=0; ii<passi; ii++){
digitalWrite(pul, HIGH);
delay (intervallo);
digitalWrite(pul, LOW);
delay (intervallo);
}
delay(1600);                        // questo intervallo è necessario per garantire che il
braccio abbia terminato le oscillazioni prima di riprendere l'immagine
takePicture();                      // take the picture
delay(200);
}
```

Now the assembly is ready and it should work more or less like this:

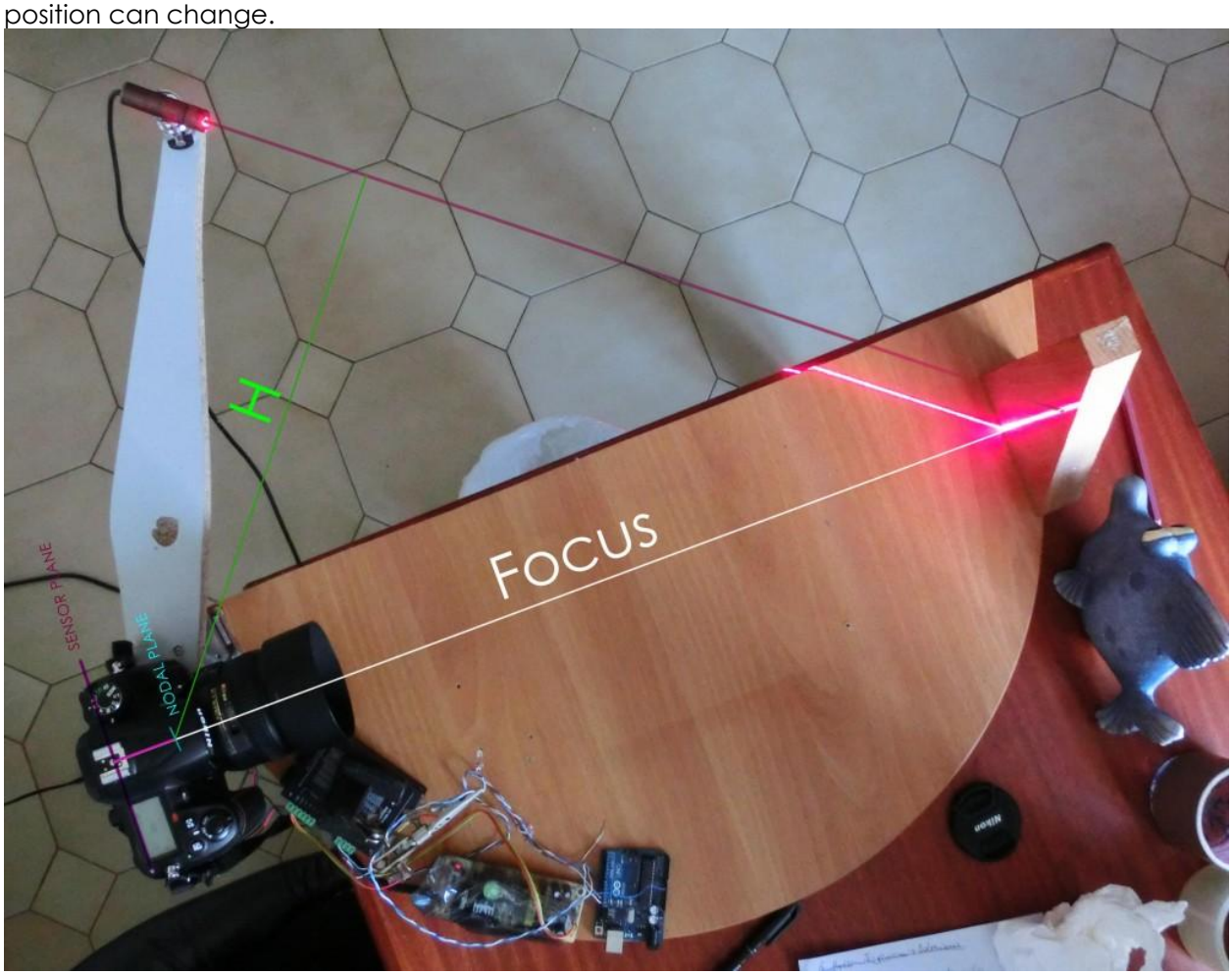
Between a photograph and the next arm rotates at an angle defined by the number of steps. In our case the angle is 360/12800*step = 0.1125 °, where step is equal to 4. When using small rotations like this one, positioning accuracy may drop down, so it would be better to use a motor with integrated gear reduction.
I repeat: take care of the alignment between the laser and the camera. Both the roll and pitch axes must be orthogonal to the motor rotation axis .
The images should represent only the laser beam (and the brightness of the laser does not lead to saturation of the cmos). However, to prevent any lights or hidden objects being visible in the image and make it easier for the software to find the center of the laser track, I coded a filter, based on the parameter referred to as "taglio", which filters the pixels with red channel value lower than the cut, placing them equal to zero.
Use the lowest ISO setting that your camera allows (in order to obtain an image as clean as possible) and manual focus (to avoid that the machine focusing between a photo and the next, modifying the effective focal), for my object, my distance between object and laser (about 600mm) and my laser (a 100mW), good exposure settings are t = 1/250 and f11. For less brighter laser probably 1/125 f8 should be better. Obviously, the diaphragm must be very close, due to the short distance to the subject and the need for a relatively large depth of field.
Take the photos in a partial darkness. Images that you will get should look like this one:

Step 3: Geometry definition

I used a fixed focal length lens (50mm f1.4). In case you are using a zoom lens make sure that the zoom does not change after you have fixed it.
Block the zoom rotation with some tape.
At this point you need to determine four measurements:

- the distance "H" between the laser plane and the lens nodal point
- the focal distance (focus), or the distance at which the laser passes through the vertical plane of symmetry of the image
- the distances between the arm axis of rotation and the nodal point. (X and Y shift)

Note: To obtain the nodal point of your lens you can use this method.
If you use, as I did, a 50mm fixed lens, a first approximation can be assumed that the nodal point is just 50mm from the sensor plane (which is identified on the camera by the symbol with a line through a circle). On the web you can find many tables with the nodal point position for various optics. You have to mind that with the variation of the focus length s the focal point position can change.




Nikon
AF-S NIKKOR 50mm 1:1.4G
NODAL POINT
Y SHIFT
X SHIFT
SENSOR PLANE
OFF ON

Step 4: Image processing

For the images processing I wrote a script, rather brutal, in Matlab. I like Matlab, it allows you to focus on the essence of the algorithm, leaving the semantic subtleties of C + + to those who know how to manage it. Yes, it is slow, but with a few lines of code you can get what you need. I think that this script is usable with little effort even with Octave , but I have not checked.
The script has to be executed in the same directory of the images. hanging around in the code, there are various parameters to be set:

- passo_angolare = angolar step. The pitch, in degrees, between one image and the next (linked to the parameter set on Arduino)
- taglio= cut. The intensity value below which all pixels are set to zero (in my images 10 is a good value)
- parametro= parameter of the smoothing process (1 = highest noise level, 0.0001 smooth)
- H = distance between the laser beam and the nodal point (see images above)
- distanza_focale= focus (see images above)
- focale=focal length of the lens
- shift = distance between the nodal point and axis of the motor (I have not integrated the shift in Y because with my lens is equal to zero)
- larg_sensore= sensor width, in mm (in this case 15.6mm)
- alt_sensore= sensor height, in mm (in this case 23.5mm)

scanner.m

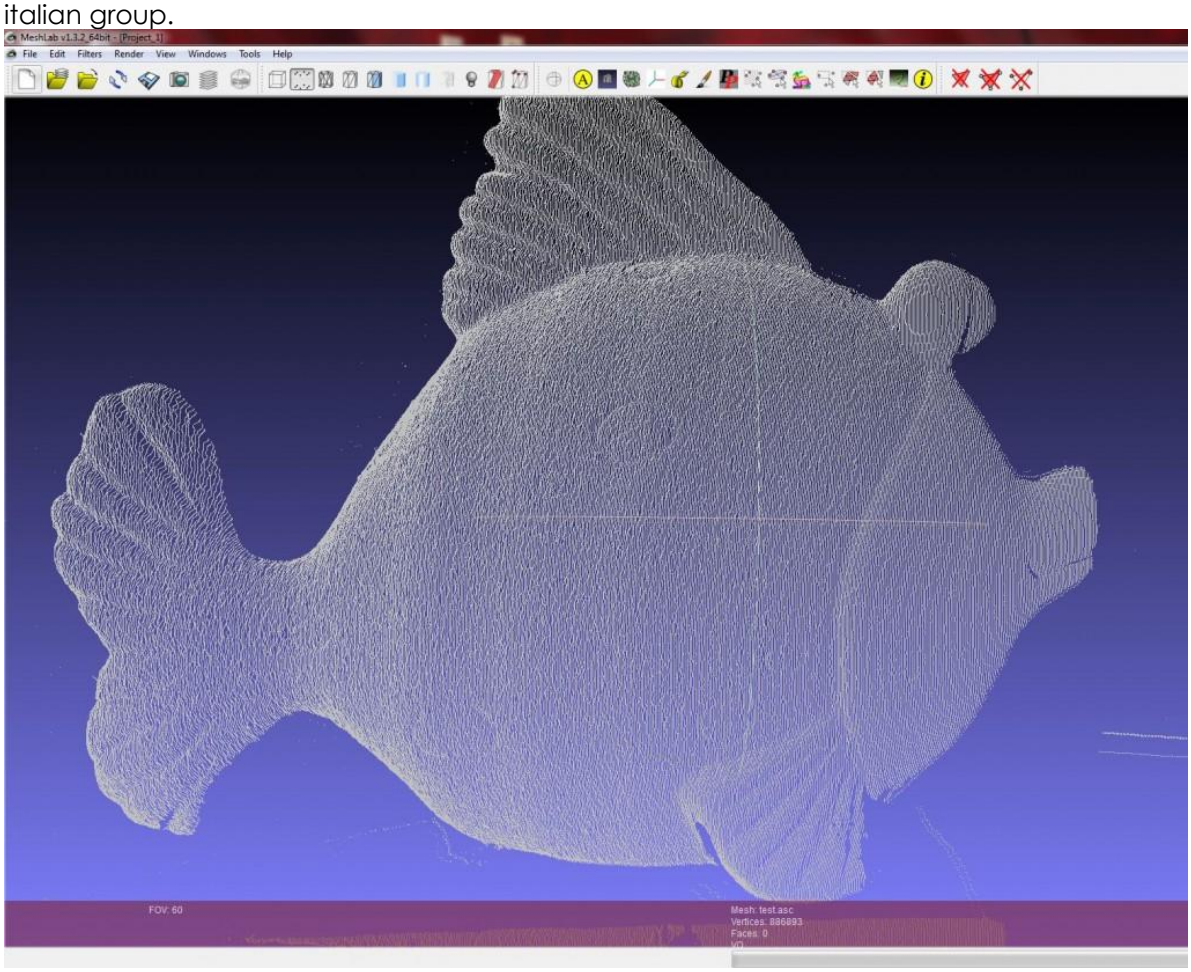
```
clear;
framedir=dir('*.jpg');
riga=size(framedir);
righe=riga(1);
passo_angola=-0.03222; %gradi
passo_angolare=degtorad(passo_angola);
iniz=[0,0,0];
save('test.asc', 'iniz','-ASCII');
for step =1:righe
% leggo immagine
RGB = imread(framedir(step).name);
% la ruoto per cercare i massimi lungo le colonne
RGB=imrotate(RGB,90);
% trasformo gli interi in valori float a singola precisione
rgb = single(RGB);
% determino dimensione immagine
a = size (rgb);
rgb = imresize(rgb, [a(1)/1 a(2)]);
a = size (rgb);
passo=0.25;
%produco vettore di 10x elementi rispetto alle colonne di rgb
fine= a(1)-passo;
corto=0:1:fine;
indicecolonna=1:1:a(2);
```

```
lungo=0:(passo):(fine);
slun= size (lungo);
taglio=10; %valore di taglio del nero
dist=1:a(2);
%ciclo per creare l'interpolazione sulla colonna a
% e trovare la posizione del massimofor i = 1:a(2)
colonna=single(rgb(:,i,1));
%creo interpolazione a decimo di pixel
ys=interp1(corto,colonna,lungo,'spline');
[c,pos]=max (ys);
% ruoto la posizione dello zero
poss=-pos+slun(2);
% se l'intensità del massimo è inferiore al taglio –> nel vett dist
% metto NaN oppure la posizione
if (c<taglio)
dist(i) = 0;
else
dist(i) = pos;
end;
end;parametro= 0.01; %parametro di smooth
% faccio smooth del vett dist, usando il parametro
smooth= csaps(indicecolonna,dist,parametro,indicecolonna);
% se il valore smoothato esce dall'immagine lo faccio appoggiare sul bordo
for i = 1:a(2)
if (smooth(i)<0)
smooth(i)=0;
else if (smooth(i)>slun(2))
smooth(i)=slun(2);
end;
end;
if (i>2)
if (abs(smooth(i)-smooth(i-1))>5)
smooth(i-1)= 0;
end;
end;
if dist(i)>0
else
smooth(i)= 0;
end;
end;centraggio=0.5/passo;
H=-378; %mm
distanza_focale= 630; %mm
shift=45;
focale=50; %mm
alfa=asind(H/distanza_focale);
beta =0;
sigma =0;
h=0;
```

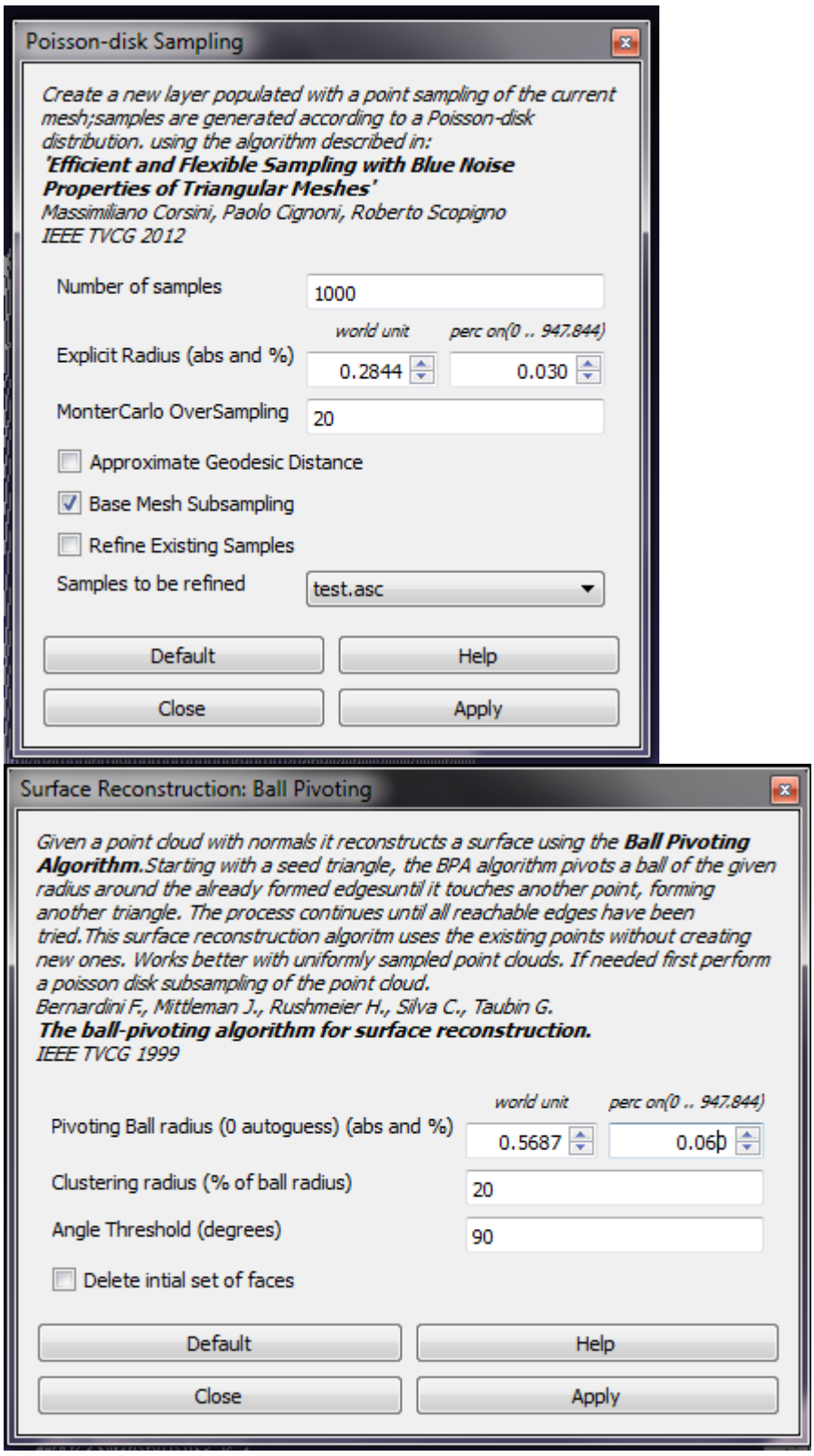
```
l=0;
larg_sensore=15.6; %larghezza sensore, dopo la eventuale rotazione
alt_sensore=23.5; %altezza sensore, dopo la eventuale rotazione
h2=slun(2)/2; % posizione della metà altezza sensore
l2=a(2)/2; %posizione metà larghezza sensore
dpm_oriz=larg_sensore/a(2); %larghezza pixel orizz
dpm_vert=alt_sensore/slun(2); %larghezza pixel vert
angoli=double (zeros(2,1));
angola=double (zeros(2,1));
nonzeri=1;
nonzero=1;
for i = 1:a(2) %ciclo per il numero delle colonne
if smooth(i)>0 % conto i punti validi, diversi da zero
nonzero=nonzero+1;
end;
end;ZETA=zeros(nonzero,3); %inizializzo vettore distanze
ZETAcart=zeros(nonzero,3);for i = 1:a(2) %ciclo per il numero delle colonne
if smooth(i)>0 %solo per i punti validi
h=(h2-smooth(i)+centraggio)*dpm_vert; %trasformo in mm l'altezza rispetto al centro
beta=atand(h/focale); %calcolo beta
l=(l2-i)*dpm_oriz; %trasformo in mm la larghezza
sigma=atand(l/focale); %calcolo sigma
zeta=(H/sind(alfa+beta))/cosd(sigma); %calcolo zeta
angoli(1,1)=90-beta;
angoli(2,1)=90-sigma;
angola= phitheta2azel(angoli)/360*2*pi; %trasformo in coordinate azimut elevation
%trasformato in coordinate cartesiane
[ZETAcart(nonzeri,1),ZETAcart(nonzeri,2),ZETAcart(nonzeri,3)]=sph2cart(angola(1,1),angola(2,1),z
eta); %trasformato in coordinate cartesiane
nonzeri=nonzeri+1;
end;
end;%sposto sistema di riferimento
ZETA=ZETAcart;
ZETA(:,1)=ZETAcart(:,1)-shift;
passo_totale=step*passo_angolare;
%creo matrice di rotazione
ruota=angle2dcm(0,0,passo_totale);%aggiungo rotazione di step angolare
for t=1:nonzeri
ZETA(t,:)=(ruota*(ZETA(t,:))')';
end;
save('test.asc', 'ZETA','-append','-ASCII');
end;
```

Step 5: Meshing

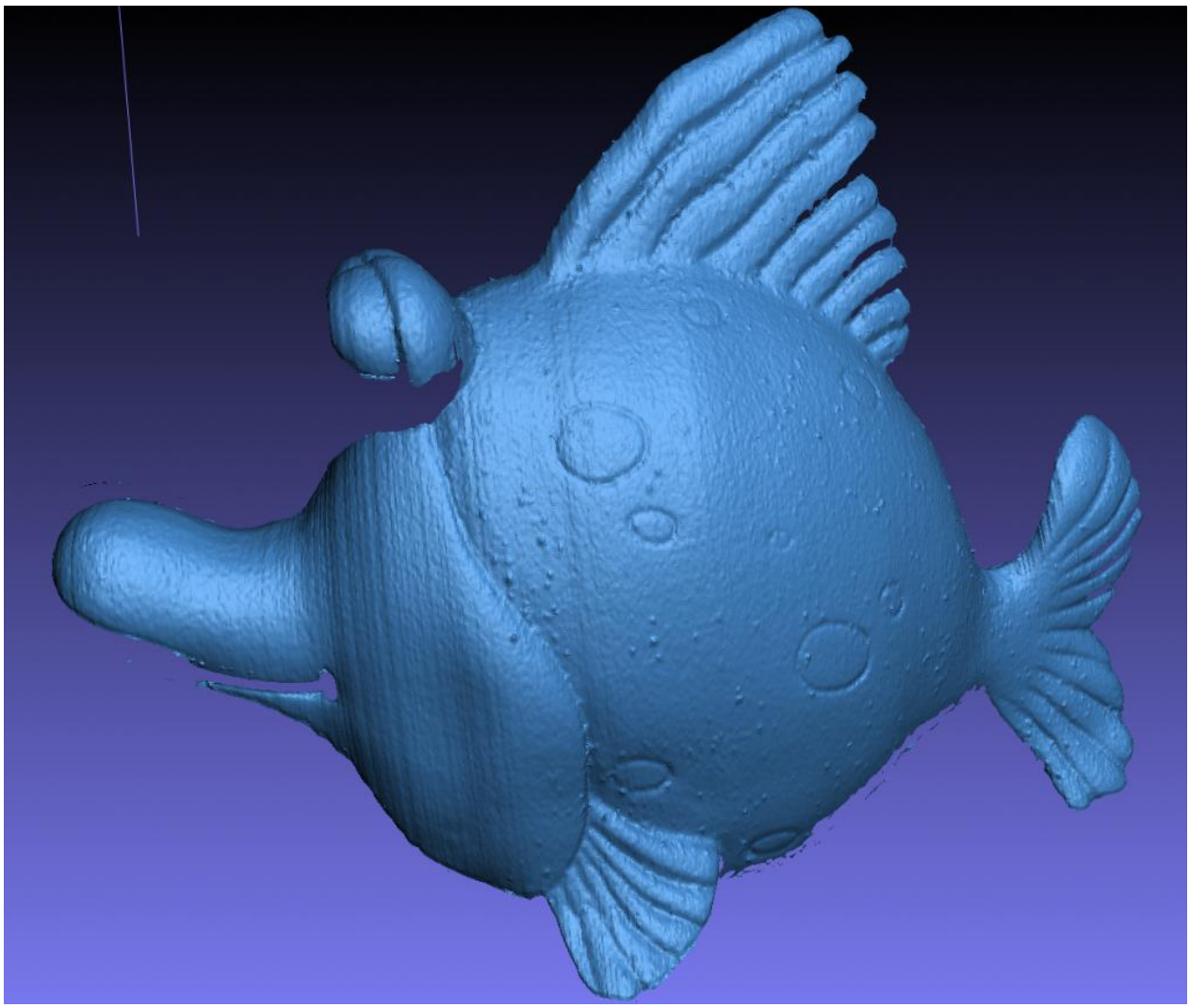
The matlab scritp save an ASCII file named test.asc, which contains the point coordinates. To import the point cloud and meshing it I use MeshLab , great tool, powerful and free, written by VCG at CNR, Italy. NB. obviously also the Arduino is the result of the work of another great italian group.



Meshing phase requires two steps, the first is the points decimation, using the poisson disk sampling, followed by the mesh generation, using the pivoting ball algorithm.



I suggest to set the poisson disk sampling radius at a little more than the half of the typical distance between two lines of points, and the pivoting ball radius at twice of the previous one.
The quality of the results depends on many factors: the thickness of the laser beam, surface scattering, arm oscillations, axes alignment, ecc.
The mesh should look something like this:



obviously the complete model reconstruction requires several scans that had to be aligned with each other in MeshLab in order to create a complete mesh representing the volume.
This is the original fish:

Good job!

For info, questions and suggestions mail me at: scanner@metaingegneria.com.