3D Reconstruction of Reflective Surfaces

by

Sarah Fernandes-Pinto-Fachada, B.Eng.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2017
Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Sarah Fernandes-Pinto-Fachada

August 27, 2017
Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Sarah Fernandes-Pinto-Fachada

August 27, 2017
Acknowledgments

Firstly, I want to thank my supervisor, Rozenn Dahyot, for her advice, discuss, and time throughout the dissertation. Her guidance was immeasurable to help me complete this thesis.

I would like to thank my boyfriend Godeffroy for helping me proofreading again and again the report.

I am grateful to my mom and sister for their support and encouragement.

Last but not least, a special thanks goes to my friends in the IET master, who made this year be so wonderful.

Sarah Fernandes-Pinto-Fachada

University of Dublin, Trinity College

September 2017
3D Reconstruction of Reflective Surfaces

Sarah Fernandes-Pinto-Fachada,
University of Dublin, Trinity College, 2017

Supervisor: Rozenn Dahyot

3D reconstruction of reflective surfaces remains a challenging problem, as well as its applications for augmented reality. This dissertation attempts to design and implement a solution to reconstruct such surfaces in any environment, and use the results for animation. The environment is used as a prior to reconstruct both planar and curved mirrors. The surface of the mirror is deduced from correspondences between points and their reflection. The correspondences allow to find a symmetry plane, or to triangulate points of the specular surface. Eventually, the results are used and demonstrated in an animation, where a computer made object and its reflection are inserted in the scene. This could be a first step toward augmented reality in a scene with mirrors.
Contents

Acknowledgments iv

Abstract v

List of Figures ix

Chapter 1 Introduction 1
  1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
  1.2 This Dissertation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

Chapter 2 State of the art 3
  2.1 Standard 3D reconstruction . . . . . . . . . . . . . . . . . . . . . . . . . . 3
    2.1.1 Structure from motion . . . . . . . . . . . . . . . . . . . . . . . . . . 3
    2.1.2 Neural networks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    2.1.3 Visual hull . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  2.2 Specular surfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    2.2.1 Mathematics of specularities . . . . . . . . . . . . . . . . . . . . . . . 5
    2.2.2 Reconstructing specular surfaces . . . . . . . . . . . . . . . . . . . . . 6
    2.2.3 3D reconstruction using a mirror . . . . . . . . . . . . . . . . . . . . . 8
    2.2.4 Rendering specular surfaces . . . . . . . . . . . . . . . . . . . . . . . . 8
  2.3 Camera parameters estimation . . . . . . . . . . . . . . . . . . . . . . . . . 10
    2.3.1 The pinhole camera . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.3.2 Matching 2D and 3D .................................................. 11
2.3.3 Pose from $n$ points .................................................. 11
2.3.4 Lens distortion ....................................................... 12
2.4 Software and tools ..................................................... 13
  2.4.1 OpenCV ............................................................ 13
  2.4.2 OpenMVG .......................................................... 13
  2.4.3 OpenMVS .......................................................... 13
  2.4.4 Meshlab ............................................................ 14
  2.4.5 Blender ............................................................. 14

Chapter 3 Standard 3D reconstruction ................................ 15
  3.1 Structure from motion .............................................. 15
      3.1.1 Acquisition of a standard scene ................................ 15
      3.1.2 Acquisition of specular objects ................................ 16
    3.2 Localising a camera in a 3D environment ....................... 18
  3.3 Pipeline ............................................................ 21

Chapter 4 Planar mirrors ................................................. 24
  4.1 Context ............................................................. 24
  4.2 Localising a mirror ............................................... 24
  4.3 Implementation ..................................................... 26
  4.4 Results and limitations ........................................... 27

Chapter 5 Mirrors of any type .......................................... 30
  5.1 Context ............................................................. 30
  5.2 Surface reconstruction ............................................ 30
  5.3 Implementation ..................................................... 34
      5.3.1 Correspondences 2D-3D ...................................... 34
      5.3.2 Finding the distortion ...................................... 35
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.3 Triangulation</td>
<td>35</td>
</tr>
<tr>
<td>5.4 Results and limitations</td>
<td>36</td>
</tr>
<tr>
<td>5.4.1 Planar surfaces</td>
<td>36</td>
</tr>
<tr>
<td>5.4.2 Surfaces of any type</td>
<td>37</td>
</tr>
<tr>
<td>Chapter 6 Application: Animation and rendering</td>
<td>39</td>
</tr>
<tr>
<td>6.1 Mannequin animation</td>
<td>39</td>
</tr>
<tr>
<td>6.2 Implementation and video</td>
<td>41</td>
</tr>
<tr>
<td>6.3 Technical limitations</td>
<td>42</td>
</tr>
<tr>
<td>Chapter 7 Conclusions and future work</td>
<td>45</td>
</tr>
<tr>
<td>7.1 Conclusion</td>
<td>45</td>
</tr>
<tr>
<td>7.2 Point correspondences and segmentation</td>
<td>46</td>
</tr>
<tr>
<td>7.3 Planar mirrors</td>
<td>46</td>
</tr>
<tr>
<td>7.4 Mirrors of any type</td>
<td>47</td>
</tr>
<tr>
<td>Appendices</td>
<td>50</td>
</tr>
<tr>
<td>Appendix A Neural networks</td>
<td>51</td>
</tr>
<tr>
<td>Appendix B Links to code, 3D models and video</td>
<td>53</td>
</tr>
<tr>
<td>Bibliography</td>
<td>54</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Lewis Carroll, Through the looking glass, 1871, illustration by Sir John Tenniel</td>
<td>1</td>
</tr>
<tr>
<td>2.1</td>
<td>Reflection of a same point $P$ is observed on different points of the surface as the camera moves</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Point $P$ corresponds to infinitely many possible points and normals for the surface, all on the same line.</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>(a) For a camera far from the mirror, the scene and its reflection look really different. (b) For a close camera, they are almost symmetric.</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>Results of reconstruction of small scenes, (a) from 38 images (b) from 83 images</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>Two reconstructions of specular buildings. From top to bottom: photo of the place, sparse point cloud, dense point cloud, textured mesh.</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Correct points are mostly real feature points instead of reflections.</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Knowing where is the camera in the scene allows to locate the mirror in the same scene.</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Blender interface to select correspondences on (a) the 3D model and (b) on the photo.</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>(a) Photo of the scene (b) Rendering after locating the camera (c) Superposition of the two images.</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>Pipeline: from photos to an animation containing mirror surface.</td>
<td>23</td>
</tr>
</tbody>
</table>
4.1 (a) Photo of the scene. (b) Part of the photo corresponding to the actual scene. (c) Part of the photo corresponding to the reflection. 25
4.2 Symmetry in a configuration with a planar mirror 25
4.3 There does not exist symmetry plane between those two cameras. 26
4.4 (a) Photo of the scene. (b) Rendering after finding the plane. (c) Superposition of the two images. 28
4.5 (a) The two cameras located in the scene. (b) Rendering with an infinite plane. 28

5.1 (a) Photo of the scene. (b) Part of the photo corresponding to the actual scene. (c) Part of the photo corresponding to the reflection. 31
5.2 Reflexion of point $P$. 31
5.3 Possible points and normals for the surface. 32
5.4 A second camera allows to triangulate the surface points and to compute normals. 32
5.5 For surface of any type, there is no stigmatism. 33
5.6 Correspondence between image seen by camera $C''$ (left) and image seen by camera $C$ (right). 34
5.7 (a) Photo of the scene. (b) Rendering after reconstructing the surface. (c) Superposition of the two images. 36
5.8 (a) Photo of the scene. (b) Rendering after reconstructing the surface. 37
5.9 (a) Photo of the scene. (b) Rendering after reconstructing the surface. 37
5.10 (a) Photo of the scene. (b) Undistortion of the photo. (c) Rendering after the same distortion. (d) Rendering undistorted. 38

6.1 Skeleton of mannequin model. 40
6.2 (a) Model of the mannequin obtained by 3D reconstruction. (b) Computer created model. 40
6.3 Hybrid image: half photo, half rendered 41
Chapter 1

Introduction

1.1 Motivation

Looking-glasses reflect a certain aspect of reality in literature and art. They reveal secrets and wishes, contain much more than the mere visible reality. Vampires cannot be seen in a mirror, Alice crosses the looking-glass to find a fascinating world (see figure 1.1). Cinematographic industry sometimes uses ‘magic mirrors’ in scenes, where a character differs from their reflection: they can be absent, shown in the future,... It is always challenging to create the impression of symmetry, while both sides are different.

Figure 1.1: Lewis Carroll, Through the looking glass, 1871, illustration by Sir John Tenniel
Computer graphics could help to render such mirrors, provided some information is given, such as its location and its shape. Moreover, to render the mirror, not only its own structure has to be known, but also what it reflects. So a 3D model of the mirror is not enough to obtain a realistic scene with magic mirrors.

While many approaches have been developed to reconstruct a reflective surface in 3D, they often make assumptions on the nature of the surface, or they use patterns such as chessboard to recover the shape of the mirror from the pattern distortion. The output is then only the shape of the mirror and cannot be directly rendered in the environment they have been placed for animation.

1.2 This Dissertation

This dissertation attempts to provide a pipeline for reconstruction of a specular surface placed in a scene. Given that scene with the specular object, modifications or animation will be done, demonstrating the technique. Such an animation is a first step toward a augmented reality application with mirrors.

The main objective of this work is to create and implement a solution to the problem of any specular surface 3D reconstruction, which can eventually be used in augmented reality (real-time) or animation (off-line).

The next chapter presents the literature of 3D reconstruction and specular surface acquisition. Chapter 3 proposes a pipeline for specular surface reconstruction, and its two first steps, which sit in the domain of standard 3D reconstruction. In chapter 4, we provides a simple solution to recover the position of a planar mirror, in a scene given as a prior. In chapter 5, we then try to modify and adapt this technique to surfaces of any type, such a curved mirrors. Eventually, the obtained results are reused in an animation, where the mirrors can be seen in their context with additional objects, in chapter 6.
Chapter 2

State of the art

In this chapter, the state of the art of specular surfaces is discussed. First, the standard techniques of 3D reconstruction are reviewed. The next section provides more details about mathematics and techniques used in the case of specular surfaces. Eventually, used tools and software are presented.

2.1 Standard 3D reconstruction

Before reviewing the particular case of specular surfaces, a brief presentation of more standard techniques is made. Some of those ideas are indeed relevant for specular 3D reconstruction.

2.1.1 Structure from motion

Structure from motion (SfM) is a technique to get a point cloud, camera positions and parameters from a dataset of photographs of a same scene taken from various points of view [1, 2].

1. Find points of interest (eg. SIFT [3]) in each view.

2. Match those points across views.
3. Compute every camera relative pose.

4. Triangulate points position.

5. Bundle adjustment.

This pipeline generates a sparse point cloud that can be densified, then a mesh can be computed and textured.

This technique has proven its efficiency and is now widely used. Nevertheless, as points of interest are needed, transparent or specular surfaces are cases of failure for standard structure from motion.

2.1.2 Neural networks

While SfM needs at least two photos to give a result, there exists solutions to estimate a 3D shape from a single view. With training data, neural network can recognise a type of object and estimate its shape. Faces can be reconstructed accurately thanks to a deep neural network [4]. Huge variety of different objects can also be deduced with a single view and a trained neural network [5].

As neural networks do not rely directly on feature points, it can be more robust to recognise and reconstruct specular, or transparent, objects.

2.1.3 Visual hull

The data of an object contour under different points of view allow to deduce a 3D shape called visual hull. The contour often needs to be manually given but the knowledge of the visual hull can be taken as a prior for a more precise reconstruction, as in [6] and [7].

2.2 Specular surfaces

In this section, more specific techniques about specular surfaces will be presented. After an overview of the mathematics behind specular surfaces, reconstruction of those surfaces
as well as 3D reconstruction of standard objects using mirror are discussed. The last part explains how to render such surfaces.

2.2.1 Mathematics of specularities

In usual structure from motion, tracked features are real, while in reconstruction of specular surfaces, features can be either real (marking on the surface) or virtual (reflection of a real feature). In the following, we will only consider virtual features, which are reflections.

Contrary to real features, tracking virtual features on a mirror’s surface does not allow to deduce the 3D position of the surface point by triangulation (see figure 2.1). Indeed, on regularly curved surface, the triangulation of such a point would give a caustic instead of a well localised point [8].

Figure 2.1: Reflection of a same point $P$ is observed on different points of the surface as the camera moves

However, Zisserman and al. prove that convexity and concavity of a reflective surface can be determined by the tracking of the virtual features [9]. Savarese and al. analyse how a known curve is transformed through reflection, and how to deduce the surface thank to differential analysis for a parametric surface, locally approximated by paraboloid [10]. They also show that for a given virtual feature, for a known camera position, the corresponding surface point and normal are related, and the point is on a straight line (see figure 2.2).
2.2.2 Reconstructing specular surfaces

The problem of reconstructing complex surfaces can be studied from various angles. The number of views varies from 1 to multiple, and the camera can have a known or unknown movement. The environment can be totally unknown, supposed infinitely far away, or made of known patterns. It can also be static or changing. Ihrke et al. have done a review of these techniques [11].

Using prior knowledge on the surface

To help the reconstruction algorithm, assumptions can be done over the specular surface. Assuming that the surface is locally parabolic allows to simply parametrise it, and can still cover many cases. However other assumptions, stronger, can lead to very specific pipelines.

Assuming that the surface is spherical [12] gives a lot of information. Triangulation of virtual points results in a smaller sphere with the same center as the reflective sphere. A sphere detection algorithm allows to find the center of the specular sphere. The radius can hence be deduced thanks to the normals, to create the surface with correct reflections.

Another idea would be to manually segment the silhouette of the specular object.

Figure 2.2: Point $P$ corresponds to infinitely many possible points and normals for the surface, all on the same line.
and use the visual hull to initialise the reconstruction. The following steps are then reproduced in a loop until the mesh is accurate enough or that no more significant progress is made:

- The colors of reflections are looked up in a known environment map (obtained for example with a 360 camera);
- Probable normals are computed thanks to the reflections;
- Surface is refined according to the new normals and to the visual hull.

Matching feature points

Even if applying directly structure from motion on feature points of a mirror surface does not work, feature points can be used and matched across several views of a same moving object [14]. Assuming the environment is infinitely far away makes the location of the features depend only on the normals. This gives constraints, and assuming the object is described by a quartic equation, they can be solved to build a depth map of the surface.

Patterns

The most explored technique for reconstruction of mirrors is to use known structured patterns or light. Patterns such as chessboards provide straight lines and feature points that can be easily tracked.

A pattern can have a known motion [15] to disambiguate the possible positions of the points of the surface. The camera can be calibrated in the same process than acquisition of the data for 3D reconstruction [16].

Single view analysis is also possible: Savarese et al. show that given correspondences between aligned points on a pattern both seen directly and in the reflection, they are able to compute the surface up to 3rd order [17]. A similar differential analysis show that dense correspondences between 3D points of the pattern and 2D location of their reflection permit to find the shape of a specular surface smooth enough [18].
With a fixed camera, patterns made with a known light are distorted, which provides enough information to retrieve the surface’s shape accurately [19]. Moreover similar techniques exist for refractive surfaces [20].

2.2.3 3D reconstruction using a mirror

Another problem than can be studied with specular surfaces is the 3D reconstruction using a mirror. Indeed, a single view of an scene and its reflection can be separated in two different views, allowing stereo vision [21, 22]. Multiple mirrors can be used to obtain multiview [23]. We can note that a single view of a symmetric object and the symmetric of that view provide stereo as well [24].

The main difficulty in such an approach is to segment the reflection from the direct view in the image. If the camera is far away from the mirror, both differ greatly, and detecting symmetry is not enough (see figure 2.3). If the camera is close to the mirror plane, symmetry detection becomes possible. A common case of such configuration is water reflection [25].

Eventually, the mirror used for 3D reconstruction can be spherical [26]. The surface shape and size of the mirror are known. A spherical mirror captures a wider angle of view, and the shape of the sphere can be reversed to undistort the view. Then, with multi-view the 3D reconstruction is done.

2.2.4 Rendering specular surfaces

Rendering mirrors is a well studied problem [27]. The classical OpenGL pipeline does not render mirrors simply. Most often, ray tracing needs to be used to capture all the complexity of the scene. However for some simple cases, faster techniques exist.

For an infinitely far away environments, the color of the surface can be looked up in an environment map. Other objects cannot be seen in the reflection that way, so the technique is not suitable for rich scene with many objects.
Figure 2.3: (a) For a camera far from the mirror, the scene and its reflection look really different. (b) For a close camera, they are almost symmetric

For a single planar mirror, two direct renderings are enough: one from the camera and one from the symmetric of the camera. The views are then combined to the final result.

For glossy objects, not necessarily planar, the scene can be rendered in several passes, one for the colors, the normals and to corresponding 3D position of every pixel, but no reflection. Then, thanks to ray marching from every point of the specular surface, reflections can be retrieved.

Specular objects also have glints, due to imperfection. Those glints help human eye to distinguish the surface from a scene inside the mirror. There exists solutions to render that photo-realistically [28] using complex normal-maps.

Being able to render specular surface is useful to compare the result of a 3D reconstruction to a ground truth image, if no 3D model of the surface is known.
2.3 Camera parameters estimation

Given a 3D reconstruction of a scene and a photo of that scene, one may want to compute the position of that camera, and its intrinsics. This section presents the material that we are going to use to recover the information of location of a camera.

2.3.1 The pinhole camera

The most used camera model is the pinhole camera, and it is the one used in this work. A pinhole camera is defined by its intrinsic parameters (focal length, optical center) and extrinsic parameters (position and rotation).

Camera coordinates $X$ and $Y$ correspond to the $x$ and $y$ axis of the photograph, while $Z$ is the front direction.

We define the intrinsic matrix by

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$f_x$ and $f_y$ are the sensor size divided by focal length. $s$ is the skew but is almost always supposed to be 0. $c_x$, $c_y$ are the optical centre’s coordinates.

The extrinsic matrix is

$$\begin{bmatrix} R | t \end{bmatrix} = \begin{bmatrix} t_x \\ R_{3 \times 3} \\ t_y \\ t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $R_{3 \times 3}^T$ is the rotation, and $-R_{3 \times 3}^T t$ the camera position.

The camera matrix is

$$P = K \begin{bmatrix} I_3 | 0 \end{bmatrix} \begin{bmatrix} R | t \end{bmatrix}$$
Hence, for a 3D point \( X = (x, y, z, 1)^T \), \( X_{\text{cam}} = \begin{bmatrix} R & t \end{bmatrix} X \) is the point in camera coordinates.

Let

\[
x = PX = \begin{pmatrix} u \\ v \\ w \end{pmatrix}
\]

The corresponding point on the photo is

\[
x_{\text{photo}} = x/w = \begin{pmatrix} u/w \\ v/w \\ 1 \end{pmatrix}
\]

2.3.2 Matching 2D and 3D

The first step to locate an image in a 3D environment is to get correspondences between some image points and their location in the scene.

That can be done indirectly, by matching feature points of the image to feature points of other known images, that have been used for reconstruction. Their feature points have already been triangulated, hence they have a corresponding 3D location.

Nevertheless, the 3D models are often manipulated, scaled or cut, so that the data available from the images used to build it are not usable any more. In that case, a direct matching from the 2D points to the 3D model could be desired. A solution exists using a 3D descriptor made of a combination of the descriptors seen from several points of view [29]. The descriptors of the 3D points can computed and stored in the same time than the 3D model is build, but than can required some specific file format.

2.3.3 Pose from \( n \) points

Pose from \( n \) points, or PNP, is the name of the problem of recovering the position of a camera given the correspondences between \( n \) 2D points on the image and \( n \) 3D points.
Let \( \{p_i\}_{i \in [1,n]} \) be the 2D points of the photograph and \( \{P_i\}_{i \in [1,n]} \) the corresponding 3D points of the scene. When \( n \geq 6 \) we can solve the problem by a singular value decomposition (SVD) that minimise the error \([30,31]\).

For simplicity, let assume the camera matrix \( K = I_3 \).

Let
\[
A_{1i} = \begin{bmatrix}
P_i.x, P_i.y, P_i.z, 1, 0, 0, 0, 0, -P_i.x p_i.x, -P_i.y p_i.x, -P_i.z p_i.x, -p_i.x
\end{bmatrix}
\]
and
\[
A_{2i} = \begin{bmatrix}
0, 0, 0, 0, P_i.x, P_i.y, P_i.z, 1, -P_i.x p_i.y, -P_i.y p_i.y, -P_i.z p_i.y, -p_i.y
\end{bmatrix}
\]
.

We pose \( A = [A_{11}; A_{21}; \ldots; A_{1n}; A_{2n}] \).

We have \( AX = 0 \) for \( X = [r_1, t_x, r_2, t_y, r_3, t_z]^T \). We can solve for \( X \) thanks to the SVD \( A = USV^T \).

\( X \) is indeed the last column of \( V \), to a scale factor. As the rotation matrix should be orthogonal, we take \( R = U'V'^T \) where \( U'SV'^T \) is the SVD of \( [r_1; r_2; r_3] \). Then we get the new \( t \) by scaling by the quotient of \( R \)'s coefficients and \( [r_1; r_2; r_3] \) coefficients.

### 2.3.4 Lens distortion

Sometimes, camera lens is not perfect and distortion can appear, changing straight lines into curved lines. Distortion is given by distortion coefficients \( K_i, i \in \mathbb{N} \) and \( P_i, i \in \mathbb{N} \).

The most often, only \( K_1, K_2, K_3, P_1 \) and \( P_2 \) are used.

A distorted pixel \((x_d, y_d)\) can be computed from the undistorted pixel \((x_u, y_u)\) by the following:

\[
x_d = x_u(1 + K_1 r^2 + K_2 r^4 + \ldots) + (P_2 (r^2 + 2x_u^2) + 2P_1 x_u y_u)(1 + P_3 r^2 + P_4 r^4 + \ldots)
\]
\[ y_d = y_u (1 + K_1 r^2 + K_2 r^4 + \ldots) + (P_1 (r^2 + 2y_u^2) + 2P_2 x_u y_u) (1 + P_3 r^2 + P_4 r^4 + \ldots) \]

Distortion can have many aspects such as barrel, pincushion or moustache.

\section*{2.4 Software and tools}

This section describes the libraries, scripts and software used for this work.

\subsection*{2.4.1 OpenCV}

OpenCV \cite{opencv} is an open-source library for computer vision. It is available in python and C++.

It has complete functionalities for image manipulation, camera calibration and linear algebra solvers.

\subsection*{2.4.2 OpenMVG}

OpenMVG \cite{openmvg} is an open library for multi-view geometry. It includes a pipeline for structure from motion:

1. Image listing,
2. Computation of features,
3. Features matching,
4. SFM, global or incremental.

It also contains various tools for localization and geodesy.

\subsection*{2.4.3 OpenMVS}

OpenMVS \cite{opemvs} is an open library for multi-view stereo vision. It completes the reconstruction done with OpenMVG by:
1. Densification of the sparse point cloud

2. Mesh computation

3. Mesh texturing

### 2.4.4 Meshlab

Meshlab [35] is an open source software for viewing and editing meshes. It is especially very convenient to check point clouds issued from 3D reconstruction, and to quickly edit meshes.

### 2.4.5 Blender

Blender [36] is an open source 3D creation software. It supports various 3D file formats for editing. It contains tools for animation and rendering, and provides a Python API. Video editing is also possible.

On top of the Blender default package, we have used a point cloud skinner script [37], to turn dense point clouds to surface, and two tutorials about character animation and skinning [38, 39].
Chapter 3

Standard 3D reconstruction

This chapter describes the work done about 3D reconstruction, which is not directly meant for specular surface reconstruction. The first section shows how classical structure from motion works for standard scenes but fails to reconstruct mirror surfaces. Then, an implementation of a solution to the PNP problem is described. The last section of this chapter presents our full pipeline for mirror reconstruction, which first steps consist in the work presented in this chapter.

3.1 Structure from motion

3.1.1 Acquisition of a standard scene

In the next chapters, the 3D environment of the mirror is supposed to be known, given as a prior for the mirror reconstruction. In most of the state of the art, the environment is a planar pattern, easy to represent. For example, a chessboard can have its corners on the $z = 0$ plane, at every integer $x$ and $y$. But for uncontrolled environments, a 3D representation is needed. For an infinitely far away environment, however, an environment map is a good enough representation. In any other case, standard 3D reconstruction is a fundamental requirement.

A scene built with small distinguishable objects such as coloured books and papers
provides a good environment to place a mirror object and be able to find correspondences between reflection and reality. Moreover, such a scene is easy to reconstruct, for the same reason: feature points are easily distinguishable.

We have built such scenes to work with. Examples of reconstruction are shown in figure 3.1. The reconstruction was done thanks to OpenMVG and OpenMVS.

![Figure 3.1: Results of reconstruction of small scenes, (a) from 38 images (b) from 83 images](image)

Two failure cases occurred. One was due to images with too few overlap, and therefore wrong camera relative position. The second failure case could happen during point cloud texturing process, a too high number of images made the program run out of memory. This last case is due to the implementation rather than data.

### 3.1.2 Acquisition of specular objects

Before developing a specific algorithm for reflective surfaces, it is worth to test what type of results the same reconstruction software can provide on specular objects. The pictures have been taken with the same camera than in the previous section.

When the specular object takes a too important part in the photographs, it is a total failure case. After matching feature points, the camera positions cannot be found, because there are too many outliers.
Figure 3.2: Two reconstructions of specular buildings. From top to bottom: photo of the place, sparse point cloud, dense point cloud, textured mesh.
When the object is a bit less specular, such as in building, a few points are found on the surface. Then during the densification of the point cloud, a few other points can be triangulated on the surface, but the shape can vary a lot from the expected result.

We can observe on figure 3.2 than very few points are triangulated on the specular surface during the sparse point cloud computation. Most of them appear after dense point cloud has been calculated. It is interesting to note than the correct points on the specular surface are mostly features of the building near the reflection of the sky (uniform blue texture), while incorrect and not found points correspond to reflection of other buildings and landscape (see figure 3.3).

Figure 3.3: Correct points are mostly real feature points instead of reflections.

This technique can be a good start for specular surface reconstruction, when the surface has real features and when the environment is uniform, like the sky. However, its limitations quickly invite to look for more specific and accurate techniques.

### 3.2 Localising a camera in a 3D environment

In the objective of building a pipeline for reflective surface reconstruction using the environment, we will work with a photograph of a specular object placed in that environment.
Locating the camera which took the photo in that scene can give a lot of information on how the reflection occurs (see figure 3.4).

Given a photograph and a 3D model of the scene, we will explain how to find the extrinsic parameters (position and rotation) of the camera which has taken the image. This new photograph can contain objects which are not present in the scene; it just needs to contain enough feature points corresponding to the scene to be located. These new objects can be a mirror of course, but for other applications, this algorithm can be used to enrich an existing 3D model.

We have implemented a solution to the PNP problem following the algorithm described in [30], in python. The 2D-3D correspondences are given manually, via the Blender interface (see figure 3.5):

- The 3D points are empties (Blender 3D object corresponding to a single vertex) located on the 3D model;

- The 2D points are trackers placed on the photograph.

We have not tried to implement or use an automatic way to find these correspondences,
to be sure not to have outliers, the camera location needing to be precise in the following.

Then, we solve the system of equation presented in 2.3.3. This algorithm needs at least 6 points, the width and height of the photograph (up to a scale). Eventually, we compute the focal length of the camera.

The focal length $f$ is computed after the rotation and location are found. It is set such as to minimise the reprojection error (assuming the camera is perfectly placed).

$$f = \frac{1}{2n} \sum_{i=1}^{n} \left| \frac{w}{2} \frac{p_i.x}{P_i.z} \right| + \left| \frac{h}{2} \frac{p_i.y}{P_i.z} \right|$$

where $w$ and $h$ are the image dimensions, $n$ the number of correspondences, $p_i \in [-1, 1]^2$ the pixel coordinate and $P_i \in \mathbb{R}^3$ the corresponding 3D point in the camera coordinate system. Indeed, reprojection error is given by

$$\sum_i \left( \frac{w}{2} p_i.x - f \frac{P_i.x}{P_i.z} \right)^2 + \left( \frac{h}{2} p_i.y - f \frac{P_i.y}{P_i.z} \right)^2$$

Example of results are shown on figure 3.6. Small errors can be observed on the books corners and on the Rubik’s cube. Note that the pencil has been moved, between the scene acquisition and the moment the photo has been taken.
3.3 Pipeline

We can now reconstruct a standard scene, and use the obtained 3D model as a prior for reflective surfaces reconstruction. We will propose a pipeline to place a mirror in that scene and reconstruct it. The output is not only the mirror shape, but the mirror in its 3D environment. Two cases are studied:

- Planar surfaces. This case is more simple, but the proposed solution is more accurate.

- Surfaces of any type. The results obtained for surfaces of any type is subject to more errors but still approximate reasonably the surface.

The first steps are the same for both cases, and were presented in this chapter. A 3D reconstruction of the scene is needed. Camera position for the images with specular surfaces have to be known or deduced via correspondences between the scene and the photograph. The other steps differ for planar surface and other type of surface.

In the case of a planar mirror, we first compute the position of a second camera, thanks to the PNP algorithm we have just described. Then, we find a symmetry plane between the direct view and the reflected view.

In the case of surface of any shape, we compute a distortion which fits a second camera. Then we compute the position a rotation of that camera, to triangulate the points of the
surface.

Once the specular surface is found, it can be rendered, and various animations can be
designed, such as inserting virtual character, making different reality and reflection...

An overview of the pipeline is shown of figure [3.7]
Figure 3.7: Pipeline: from photos to an animation containing mirror surface.
Chapter 4

Planar mirrors

In this chapter, a technique to localize a mirror plane in a 3D scene is presented.

4.1 Context

In this section, the assumptions of the algorithm are detailed. First, the mirror is supposed perfectly planar.

We assume that the scene is known (except of course the mirror) and that we dispose of a 3D representation of it. Some objects can be added or moved in the scene, but most of it should stay the same than in the 3D model we have.

The algorithm works with a single image containing both a part in the real scene and a part in the mirror (reflection). The objects seen in the reflected and in the real scene need to occupy a large enough proportion in the photo to allow to solve the PNP problem for both sides of the image.

4.2 Localising a mirror

A perfectly planar mirror is a symmetry plane in a 3D scene. Hence a camera observing a reflection in a mirror is seeing the exact same thing (to a symmetry) that if it was observing the real scene from its symmetric position (see figure 4.1 and 4.2). In the
following, we call “real camera” the camera configuration in which the photo has been taken, and “virtual camera” its symmetric to the plane. In figure 4.1, the real camera observes the scene as in (b) while the virtual camera is located in a way such that it observes the scene as in image (c).

Figure 4.2: Symmetry in a configuration with a planar mirror

The idea of the algorithm developed in this chapter is to find the positions and rotations of the camera and its symmetric thanks to the information available in a single
photograph. The mirror is then located in the middle of the real and virtual cameras, and oriented such as their front directions (or any corresponding directions) intersect on the plane.

Eventually the contours of the mirror can be found thanks to the photograph. We project these contours from the photo to the scene, and remove the parts of the infinite plane that are not part of the actual mirror.

In practice, there does not necessarily exist a symmetry plane between the real camera and the virtual camera, because of errors in their pose computation (see figure 4.3). Nevertheless, an approximation in still possible. The median plane between the two locations of the cameras is a good approximation.

![Figure 4.3: There does not exist symmetry plane between those two cameras.](image)

4.3 Implementation

Localising the two cameras is done through the PNP algorithm described in the previous chapter.

The mirror plane is created in Blender with the data of a position and a rotation matrix. Let \( C_r \) and \( C_v \) be the positions of the real and virtual cameras.
The position $P$ of the mirror is

$$P = \frac{C_r + C_v}{2}$$

and its normal is $\vec{n} = \frac{C_r - C_v}{\|C_r - C_v\|}$

So the rotation matrix is

$$R = \begin{bmatrix} \frac{\vec{x}}{\|\vec{x}\|} & \frac{\vec{y}}{\|\vec{y}\|} & \vec{n} \end{bmatrix}^T$$

where $\vec{x} = \vec{n} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\vec{y} = \vec{n} \times \vec{x}$. Only the last row of the matrix is important, because it gives the normal to the plan. The other rows just need to be chosen to obtain a rotation matrix.

Once the plan is created, it can be cut to have the same dimensions than the mirror on the photograph. We have not worked on that problem, and done that manually, by projecting the mirror image from the camera to the plane.

The obtained plane is then given a mirror material and can be rendered to compare with groundtruth photographs.

### 4.4 Results and limitations

Results are shown in figure 4.4 and 4.5.

The plane orientation is very similar to the input, however, small variations in that orientation lead to bigger variations in the rendered reflect.

We can observe that while the shape of the mirror has been segmented by projecting the photo on the obtained plane, it leads to an almost perfect rectangular shape, corresponding to the real mirror used in the photo.

This technique is mathematically accurate, but lacks of automatism. Finding the
Figure 4.4: (a) Photo of the scene. (b) Rendering after finding the plane. (c) Superposition of the two images.

Figure 4.5: (a) The two cameras located in the scene. (b) Rendering with an infinite plane.
correspondence points in the real scene and in the reflection is the longest step and could easily be automated. Segmenting the image in real and virtual parts is more difficult, or impossible when the knowledge of the environment is too weak.

This process can be applied for more complex mirror configuration: for each planar mirror, a new camera has to be computed to deduce the mirror position.
Chapter 5

Mirrors of any type

This chapter proposes a solution for the reconstruction of curved mirrors. It attempts to stay close to the pipeline used for planar mirror, and to adapt it for the case of any type of mirror.

5.1 Context

The assumptions made in this chapter are similar to the assumptions of the previous chapter, but this time the surface is not supposed to be planar. The surface should however show some regularity, to allow distortion computation.

We assume again that the scene is known and that we have a 3D representation of it.

The algorithm works with a single image containing both a part in the real scene and a part in the mirror. The objects seen in the reflection and in the real scene need to occupy a large enough proportion in the photo to allow to solve the PNP problem both for the reflection and for the scene.

5.2 Surface reconstruction

Similar to previous chapter, the photograph can be divided in two parts, the real and virtual part (see figure 5.1). This time, only the real part ((b) on figure 5.1) can be
Figure 5.1: (a) Photo of the scene. (b) Part of the photo corresponding to the actual scene. (c) Part of the photo corresponding to the reflection.

associated to a pinhole camera. The reflection part is distorted ((c) on figure 5.1): no pinhole camera can be found to observe the scene that way, even with a different focal length.

We will hence assume that the position of the real camera, its focal and rotation for a given picture are known. They can for example be found with the PNP solver presented in chapter 3.

Figure 5.2: Reflexion of point $P$.

Let $P$ be a point in the scene which is seen in the reflection on the specular surface at a pixel location $p \in \left[ -\frac{w}{2}, \frac{w}{2} \right] \times \left[ -\frac{h}{2}, \frac{h}{2} \right]$, where $w$ and $h$ are the image width and height (See figure 5.2). $P'$ is the point on the surface where the reflection occurs, and $\overrightarrow{n}$ its
normal. Let $f$ be the focal length of the camera. It is useful to note that in the camera’s coordinates system, $p$ can be seen as a 3D point: $p = (x, y, f)$.

![Figure 5.3: Possible points and normals for the surface.](image)

As shown on figure 5.3, given the point $P$ which is seen in reflection at pixel $p$, infinitely many couples location-normal ($P'$ and $\vec{n}$) make point $P$ be seen at the same pixel location. This ambiguity could be solved if the direction $PP'$ was known.

![Figure 5.4: A second camera allows to triangulate the surface points and to compute normals.](image)

The idea on this algorithm is to introduce a second camera $C'$ looking at the known points $P$ to triangulate the corresponding surface points $P'$ (see figure 5.4). Indeed, for
$P$ in the scene, the corresponding point $P'$ and normal $\vec{n}'$ verify:

$$
\begin{align*}
P' &= C + \lambda \vec{Cp} = C' + \mu \vec{C'p}' \\
\vec{n}' &= \frac{1}{2} \left( \frac{\vec{pC}^2}{\|\vec{pC}\|} + \frac{\vec{C'p}^2}{\|\vec{C'p}'\|} \right)
\end{align*}
$$

$\lambda$ and $\mu$ are real positive constants that be be determined by solving a linear system. As $p, p', C$ and $C'$ are known, the system can be solved.

Of course most of the mirrors do not provide exact stigmatism (only planar mirrors, and some quartic mirrors at their focus points)(see figure 5.5). The image in the mirror is indeed distorted. That distortion prevents from directly computing the second camera parameters, and will introduce errors. Nevertheless, this approximation can be correct for reconstructing small portions of the surface, where the field of view from $C'$ is not too wide.

![Figure 5.5: For surface of any type, there is no stigmatism.](image)

The possible distortions are functions of $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ and can be as various as the possible shapes of surface. We propose to compute it from a few correspondences between pixels of the reflection part of the image and 3D points. The result is hence approximated.

Once the image is undistorted, the extrinsic parameters of $C'$ can be computed. For
a pixel $p'$ seen by camera $C'$, the corresponding pixel seen by camera $C$ is $p = \phi \circ \text{dist}(p')$ where $\phi$ is an affine function to map the two images (see figure 5.6).

Figure 5.6: Correspondence between image seen by camera $C'$ (left) and image seen by camera $C$ (right).

If the distortion function is explicitly known, every point of the surface seen by the camera $C$ can be triangulated, even if no 3D correspondence is given for that pixel. That means that the point cloud generated can be as dense as desired (even if that does not give more precision for the shape of the surface), depending on the resolution of the photograph.

Then it can be skinned to obtain a surface. Smoothing effects exist in Blender to avoid seeing the triangles of the surface. Indeed, while meshes with a low number of triangles are very suitable for standard objects, mirror with a low number of triangles rarely look realistic and appealing.

5.3 Implementation

5.3.1 Correspondences 2D-3D

Once again, correspondences between 2D image points and 3D points need to be given:

- To locate the photo;
- To compute the distortion in the surface reflection;
To locate the undistorted image of the surface reflection.

The location of the camera is done with the PNP solver script that runs directly with Blender. To compute the distortion, these correspondences are exported in a file, which is then processed through a script in C++ using OpenCV.

5.3.2 Finding the distortion

We have chosen to approximate the distortion the same way lens distortion can be approximated, because the camera calibration function provided in OpenCV computes it easily. We are however aware that this model is too restrictive for most of the surfaces. The distortion function used is

$$\text{dist} : (x_u, y_u) \mapsto (x_d, y_d) = \begin{pmatrix} 
    x_u(1 + K_1 r^2 + K_2 r^4 + K_3 r^6) + (P_2 (r^2 + 2 x_u^2) + 2 P_1 x_u y_u) \\
    y_u(1 + K_1 r^2 + K_2 r^4 + K_3 r^6) + (P_1 (r^2 + 2 y_u^2) + 2 P_2 x_u y_u)
\end{pmatrix}$$

where \((x_u, y_u)\) is the undistorted coordinate of the pixel, \(r\) the distance to the distortion center and \((x_d, y_d)\) the distorted pixel coordinates.

5.3.3 Triangulation

As the mirror is rarely exactly stigmatic, the second camera position cannot be exact and work perfectly for every point. In most of the cases, the equation \(P' = C + \lambda \vec{C}_{\vec{p}} = C' + \mu \vec{C}'_{\vec{p}}\) is wrong because the corresponding rays from \(C\) and \(C'\) does not intersect.

The system to solve for \(\lambda\) and \(\mu\) is

$$\begin{pmatrix}
    (\vec{C}_{\vec{p}})_x & -(\vec{C'}_{\vec{p}})_x \\
    (\vec{C}_{\vec{p}})_y & -(\vec{C'}_{\vec{p}})_y \\
    (\vec{C}_{\vec{p}})_z & -(\vec{C'}_{\vec{p}})_z
\end{pmatrix} \begin{pmatrix}
    \lambda \\
    \mu
\end{pmatrix} = \overrightarrow{CC'}$$

As it is overdetermined, a singular value decomposition give the best possible solution. To keep the surface point to the right pixel position, we take \(P' = C + \lambda \vec{C}_{\vec{p}}\) rather than
\[ P' = C' + \mu C' p \] or the average.

If \( \lambda \) or \( \mu \) is negative, it would mean that the surface point is behind one of the cameras, which is impossible. The point can then be discarded. Moreover, if the resulting normal \( \vec{n} \) verifies \( \vec{n} Cp > 0 \), it would mean it points to the opposite side of the camera, and the surface point cannot be seen.

The coordinates of the points which are not outliers are exported in a PLY file, then skinned and rendered in Blender.

### 5.4 Results and limitations

#### 5.4.1 Planar surfaces

As this pipeline works for any surface shape, it has been tested on the particular case of planar surfaces. In that case distortion is identity, and the two cameras have the same focal length. The results are similar to those of the previous section. We can observe that the error is similar, because the second camera is the same than the one used in the previous chapter (see figure 5.7).

![Figure 5.7: (a) Photo of the scene. (b) Rendering after reconstructing the surface. (c) Superposition of the two images.](image)

The obtained surface is not perfectly planar, because the two cameras are not perfectly symmetric, so the triangulation of some points is approximated.
5.4.2 Surfaces of any type

Figure 5.8: (a) Photo of the scene. (b) Rendering after reconstructing the surface.

Figure 5.9: (a) Photo of the scene. (b) Rendering after reconstructing the surface.

The results obtained for any type of surfaces are very incorrect (see figure 5.8 and 5.9). The surface has the good convexity and looks alike the expected result, but after rendering, the reflection is very different from the groundtruth photograph. This variation is mostly explained by the choice of the distortion model, which is inaccurate. We can observe that this model is not correct on the undistorted image of figure 5.10. That inaccuracy makes the localisation of the second camera difficult, or even impossible. However the obtained results are encouraging to look for a better distortion model.
Figure 5.10: (a) Photo of the scene. (b) Undistortion of the photo. (c) Rendering after the same distortion. (d) Rendering undistorted.
Chapter 6

Application: Animation and rendering

This chapter shows an example application of specular surfaces reconstruction. While photograph, stop motion and filming can only render exact reflections and rigid surfaces, the introduction of the presented techniques in animation allows special effects, such as different actions in the reflection than in the scene, crossing the surface, destroying the surface...

6.1 Mannequin animation

The film we have realized represents an animated wooden mannequin whose reflection acts differently, and who crosses planar mirror and curved mirror surface.

The mannequin 3D model was found on turbosquid[40], a website providing 3D models. The model was then skinned with Blender as on figure 6.1.

We have not used the mannequin model obtained by 3D reconstruction, because of lack of accuracy and symmetry (see figure 6.2). The model obtained by 3D reconstruction had the advantage of being textured and having the exact proportions. However, the animation would have suffered because of the difficulty to skin a character with so many
The model found online was made of simple geometries, which allowed an efficient skinning. The wooden texture has been simply obtained with uniform diffuse color and small specular coefficient.
6.2 Implementation and video

The video has been made with both stop motion images and computer rendered images. It gathers all the techniques presented in this dissertation.

Here are the steps done to the result:

1. 3D reconstruction of the scene, without the moving character and without mirror.
2. Placing a mirror in the scene and taking a photo.
3. Mirror reconstruction.
4. Stop motion video with the character.
   Animation and rendering of the character.

![Figure 6.3: Hybrid image: half photo, half rendered](image)

In the video, the planar mirror appears three different ways:

- Only photo.
• Rendering the computed plane.

• Rendering from the symmetric of the camera, while the real part of the scene is a photo (see figure 6.3).

This last way of rendering, by compositing two images, allows a part of the reflection to differ from the scene, while the impression of a mirror is still there thanks to the rest of the scene. This also avoids the difficult work of segmenting the part of the scene that should be different. All the mirror is segmented instead, which is more easy, due to the geometric shape of the surface.

![Figure 6.4: Crossing the mirrors.](image)

Both curved and planar mirror can be crossed by the character, which would be impossible without the help of a computer (see figure 6.4).

Eventually, the surface can be deformed, which is impossible with usual stiff mirror (see figure 6.5). This has not been done in the final video, even if it is simple to obtain, because keeping visual track of the reflection is difficult.

### 6.3 Technical limitations

As the scene is often manipulated, small object moves often occur. Those moves prevent camera location and so surface reconstruction. It was important to fix the objects of
Figure 6.5: Deformation of the mirror.

Figure 6.6: Different positions for the Rubik’s cube (a) Photo after animation. (b) Initial position in 3D reconstruction.
the scene with tape, and to be sure that all the steps requiring to be in the exact same environment were done together, before risking to modify the scene. As recovering the initial position of an object is tedious, or even impossible, perturbing the scene can lead to starting the work again. On figure 6.6, we can observe for example that the Rubik’s cube has moved because of a fortuitous collision with the mannequin.

Moreover, the use of stop motion makes the camera slightly move each time the button is pressed. For the video, only one of all camera’s positions is used, which can sometimes create a strange impression.

However those limitation are more due to the material and the scene preparation than intrinsic problems.
Chapter 7

Conclusions and future work

7.1 Conclusion

In this work, two techniques for mirror reconstruction have been proposed, one for planar surfaces and one for any type of surface. Those technique do not involve complex computation or require special properties of the surface. No pattern or known light structure is needed, but the 3D environment has to be known as prior.

Some steps of the algorithm still need to be automated, but this is more an implementation problem than an open problem.

The major limitation is now the use of distortion models for curved mirrors. This should be the focus of future research.

We have demonstrated that this pipeline, with the prior of the known scene, can be eventually used to animate a computer object in the scene, and render its reflection. As this pipeline is not real-time (mostly because of the lack of automatism), it can only be used for animation, but it is a first step for augmented reality with mirrors.
7.2 Point correspondences and segmentation

Almost all the written scripts needed correspondences between points on a photo and in
the 3D model. Even if there exists automatic ways to directly or indirectly find those
correspondences, they often require a special file format, or libraries that do not work with
Blender. Moreover there could be too much outliers than manually given correspondences,
especially when working with mirror, where the same features can appear several times.

Nevertheless, the presented pipeline often asks such correspondences, which is a time
consuming and tedious task, particularly when repeated. Making this step automatic
would be a huge economy of time.

Finding in a photograph which part corresponds to the scene and which part is the
reflection is another problem that we have not treated. However, if the feature points are
automatically found, the computer needs a way to sort them. For a descriptor variant by
symmetry, matching with the 3D scene can be done once for the photograph and once for
its symmetric. The feature points that can be matched with the scene in the first case
should be in a cluster corresponding to the real part of the photo, while the points of the
symmetric of the photo that are matched belong to the reflection.

7.3 Planar mirrors

The algorithm for planar mirror reconstruction is quite complete. It could be applied to
some extend for other kinds of surface, provided enough correspondences can be found.
For example a multifaceted mirror such as in figure 7.1 could be reconstructed with several
iteration of this technique.

Two main difficulties can occur. The first is that some facet orientation may not
provide a reflection where feature point can be found (see figure 7.2). These facets would
stay not reconstructed. The other problem would be how to segment each facet of the
mirror. A solution can be to keep the part of the facet where the reflections occur, or to
manually select the contour of the facet.

7.4 Mirrors of any type

The principal cause of failure for the proposed mirror of any shape reconstruction technique is due to the chosen distortion model. The lens distortion model has been used, because a distortion followed by camera intrinsic and extrinsic parameters is easy to compute with OpenCV. However, if the given points do not match such a distortion, the best approximation done can be very erratic. The next step to improve this work would be to find a better way to automatically compute a good distortion, that ideally could be any function.

The other difficulty is due to astigmatism in the mirror surfaces. Even if it can be approximated, in some cases, it could be preferable to decompose the mirror in smaller surfaces, each with a different optical center (see figure 7.3).
Figure 7.2: Reflection of a mesh on a multifaceted sphere.
Figure 7.3: Example of very astigmatic mirror.
Appendices
Appendix A

Neural networks

This chapter shows tests done with a recent technique using neural networks for point cloud generation from a single image\textsuperscript{5}. This works does not fit in the pipeline presented, but is worth to be described.

The main idea of this chapter is to enrich a point cloud with a second view, taken with a mirror. Indeed, the insertion of a mirror in the view, would be the same as seeing two similar objects, or two sides of the same object. The two clusters of points could hence be merged to obtain a more precise one.

However the reconstructed object needs to occupy a precise portion of the image, and when it contains both the object and its reflection, at least one object fails to be reconstructed. The other solution would be to do the process twice, but the advantage of using a mirror would be lost. In that case, we could take two photographs of the object.

The software tested take as input a single photo and a mask of the silhouette of the object which has to be reconstructed.

As we can see on figure A.1, the mirror prevents to center correctly the image on the object and the reconstruction fails. It does not seem to detect that the two objects are distinct; the obtained point sets are not separated well.
Figure A.1: Result of neural network point cloud generation. (a) Input photograph. (b) Input mask. (c) Output. From top to bottom: sample provided with code, result with our own mug, result with a mirror.
Appendix B

Links to code, 3D models and video

All the code is on Github: https://github.com/Satrah/Reflective-surfaces

The reconstructed scenes are on sketchfab, a platform to upload and share 3D models:

- Specular building 1: https://skfb.ly/686Os
- Specular building 2: https://skfb.ly/686Nr
- Reconstructed scene with planar mirror: https://skfb.ly/6t7LA
- Reconstructed scene with teapot mirror: https://skfb.ly/6t7MA

Here is the link to the video: https://youtu.be/9Y201Ve0PqI
Bibliography


[37] Blender Online Community, “Point cloud skinner for blender.”

[38] A. Gonzalez, “Rigging and skinning a character model.”
https://www.youtube.com/watch?v=PPbSGUPj72w.

https://www.youtube.com/watch?v=kSDWfx6ib9k.