SORGATE: Extracting Geometry and Texture from Images of Solids of Revolution

Antonio Ledesma and Robert R. Lewis

Washington State University, Tri-Cities
contact: bobl@wsu.edu

Abstract. We describe SORGATE, a procedure for extracting geometry and texture from images of solids of revolution (SORs). It uses multivariate optimization to determine the parameters of the camera in order to build the viewing transform, as well as to reconstruct the geometry of the SOR using the silhouette. In addition to individual image analyses, it can use the data extracted from the same SOR viewed from different directions to produce a single, composite texture which can be combined with their blended geometries to produce a reconstructed 3D model. No prior knowledge other than the object's rotational symmetry is required. Camera viewing parameters are derived directly from the image.
SORGATE is useful when 3D modeling of SORs is needed yet direct measurement the physical objects is infeasible. As it does not require camera calibration, it is also fast, inexpensive, and practical. One use case might be for researchers and curators who wish to display and/or analyze the art on historical vases; metric reconstruction and proper texturing of the objects would allow this without requiring viewing in person.

Keywords: Image-based modeling and rendering · Solids of revolution · Geometry extraction · Texture extraction

1 Introduction

The task of extracting 3D information about objects appearing in 2D images and videos has wide application and is a fundamental research area in image-based modeling and rendering. To simplify the problem, methods often focus on certain classes of objects, such as those that exhibit symmetric properties. Accepting such properties as given makes it easier to extract the information. A common symmetry found in many everyday objects is rotational symmetry about an axis; objects with this symmetry are known as solids of revolution (SORs).

We present here a method which we refer to as "SORGATE" (Solid of Revolution Geometry And Texture Extraction) for doing this, as shown in Figure 1. If we have multiple images of the same SOR from different viewpoints, we can create a single composite texture that applies to as much of the SOR as the individual images cover. We can then combine this with the merged geometry information to, for example, produce a textured model of the SOR. We have incorporated this procedure into an application that can be used by people who have minimal knowledge of it.

Intended for submission to ISVC 2021, please do *not* distribute.

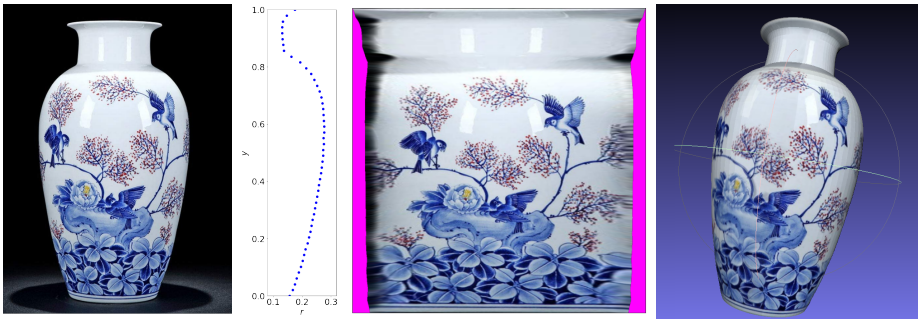

Fig. 1. Overview: Starting with one or more uncalibrated photographs (left) of an SOR, we can extract geometry and textures (center), and apply them to 3D models (right). (In this paper, pixels with no data (i.e. transparent) are coded in magenta.)

After this introduction, Section 2 highlights some similar work that has been done in the past using different approaches. Section 3 describes our overall design for the project, detailing how we apply the primary concepts and algorithms to the SOR reconstruction and texture extraction problem. Section 4 focuses on how the SORGATE application embodies the design and how a user should use the application. Section 5 describes how we evaluated each piece of the design separately, as well as discusses the results of the full process on real images. Finally, Section 6 contains concluding remarks and possible avenues of future work for the project.

This application is useful, for example, to analyze objects created on a pottery wheel or lathe which may not be viewable or measurable in person, perhaps because the object is inaccessible or no longer exists. It could also be used commercially to quickly and inexpensively generate 3D textured models of these objects for viewing on the web or for 3D printing duplicates to serve the purposes of education or advertisement.

2 Previous Work

Image-based modeling and rendering, the extraction of both geometric and texture information from images for the purpose of rendering, dates back to the work of Debevec, Taylor, and Malik [7]. Like texturing itself, its goal is to reduce the need for detailed modelling of both geometry and light interaction to improve realism.

A common first step is to determine the camera pose from the image [2, 22]. Reducing the problem domain by assuming that the object is an SOR makes the problem more tractable. Several approaches make use of projective geometry. Chellali et al. use only object contours and projective geometry to build 3D geometric models from a single image [5]. They do, however, require a calibration object in image at the base of the SOR.

Circular cross sections of SORs project to ellipses with certain invariant properties and symmetries [17, 23]. Many techniques use this and therefore require accurate fitting of ellipses to silhouette data. Liu and Hu use minimal contour data to model spacecraft that exhibit SOR properties through accurate ellipse fitting [14]. Their technique shows much improvement over older techniques [8] when applied to multiple ellipses that share the same central axis. Wong, et al. are also able to recover the geometry of an SOR from a single image by determining the surface normals from the its silhouette and use this to recover depth information [24]. Puech et al. explore a technique which extracts the three Eulerian angles relating the camera and object coordinate systems, though their technique requires two well-defined cross sections to be visible [18]. Zhang et al. propose a technique that does not require any explicit cross sections be visible, but rather a well-defined silhouette. This is useful for SORs with smooth surfaces or which lack defined tops and/or bases [26].

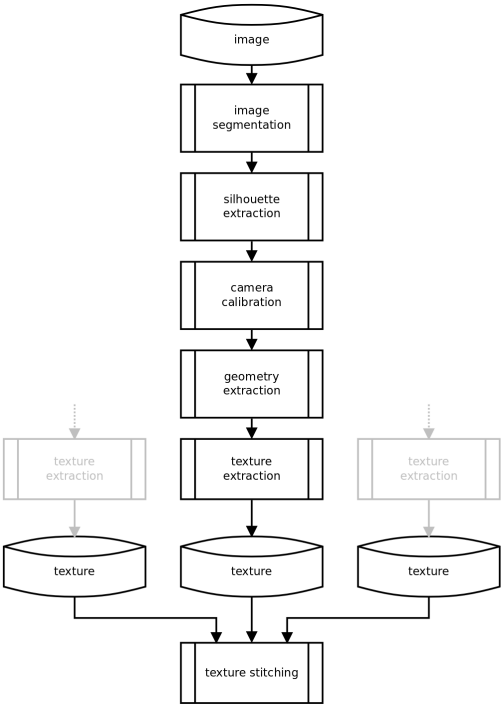

Fig. 2. The SORGATE processing pipeline.

The work done by Columbo et al. perhaps most closely relates to ours differing in that it constructs the viewing matrix through projective geometry rather than parameterized viewing transforms [6, 16]. This is a more elegant approach than ours, but our approach of finding the viewing parameters by optimization greatly simplifies the image stitching. Jain's work in [11] is also closely related, though the author imposes more constraints on the problem, such as assuming the camera is coplanar with the top cross section of the object.

3 Design

There are six major steps (Figure 2) in the processing pipeline for SORGATE. Starting with an image, we first segment the foreground (the SOR itself) from the background (any other objects/-surfaces). By applying filtering and edge detection techniques and analyzing the curvature of the silhouette we then determine the circular planar base and top cross sections of the SOR. Then we find the viewing parameters of the camera that generated the image, as well as the radii of the base and top cross sections.

Fig. 3. Image Segmentation (left) and Silhouette Extraction (right). *(Image source: [1])*

Next, we determine the geometry of the SOR: a set of (y, r) pairs estimating the SOR's radius as a function of height. In the following texture extraction step, we create a flattened version of as much of the SOR's texture as is visible in the image. Finally, if we have multiple images of the same SOR from differing viewpoints, we stitch them together to form a single texture, which we can then merge back with the geometry to create an interactive 3D representation of the SOR.

Space does not permit us to show how our initial design changed as a result of implementation, so we have incorporated those insights in this "design" section. Nor does it permit us to provide all of our formulae and algorithms and their derivations. We provide both of them in [13].

Image Segmentation An important part of image analysis is segmentation: isolating the pixels describing the object of interest from those that are irrelevant [21].

As Figure 2 shows, the image is the only data input to SORGATE. Segmentation is the first step. In our case, we use the very popular *GrabCut* algorithm [3, 19] to segment the image. We used *GrabCut* primarily because it allowed for adjustments to the segmentation at the pixel level, though one of the more recent deep learning techniques as described in [9] would allow greater automation.

GrabCut usually needs an initial hint (here, a bounding box) from the user about the location of the desired segment. Later, they may add additional "hints" for refinements on later iterations to improve the segmentation. We allow for these. Figure 3 (left) shows a typical result.

Silhouette Extraction After segmentation, the SOR is the only object in the image. From this, we apply the commonly-used Canny edge detector [4] to extract the contours, of which there may be several. This does require some interactive and perhaps unintuitive adjustments to a set of parameters; thus, our method would once again benefit from using a more precise and automated technique here. Even so, using Canny's technique, we finally take the contour with the largest area to be the silhouette. Figure 3 (right) shows this.

We need to extract the *sub-silhouettes*, those parts of the silhouette that arise from the base and the top of the SOR. We assume that the SOR rests on the xz (i.e., $y = 0$) plane and that the camera is positioned above it (i.e., $o_y > 0$) defining the z axis. The base sub-silhouette, then, is the projection of the front part of the base.

If the camera height is less than the height of the SOR, the top sub-silhouette is the projection of the front part ($z > 0$) of the SOR. Otherwise, the top sub-silhouette is is the projection of the back part ($z < 0$) of the sub-silhouette. We compute both possibilities and choose the solution that most closely matches that of the bottom sub-silhouette.

To extract the sub-silhouettes, we look for discontinuities in the curvature. Although we considered a number of methods for digital curvature estimation [25], in the end we chose our own approach. We first smooth the pixel coordinates with a Gaussian kernel, then compute the three-point curvature at each pixel. We then smooth the curvatures with another Gaussian kernel and, starting from the bottom or top extremum, find their left and right maxima. The sub-silhouettes, then, lie between these two maxima.

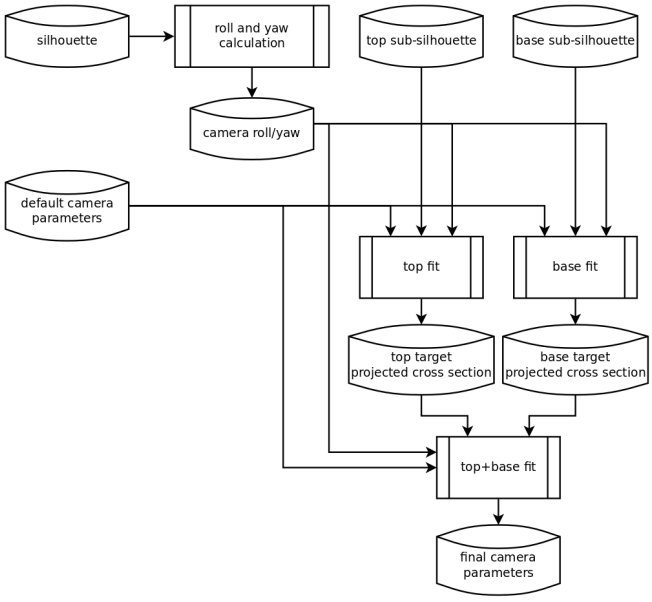

Fig. 4. Camera Calibration

Camera Calibration Our task is to map each pixel inside the silhouette to a 2D texture. This requires finding the transform from any point on the 3D surface of the SOR to the texture and its inverse.

Due to the nature of the SOR, the intrinsically circular base and top cross sections project to the elliptical sub-silhouettes. That projection, which determines both sub-silhouettes, depends on the camera's pose.

In general, a camera pose has six degrees of freedom: its 3D position ($\equiv \tilde{\mathbf{o}}$) relative to the origin and its orientation (roll, pitch, and yaw). Due to the symmetry

of the SOR, we can reduce this to five by requiring that the camera lie within the yz plane (i.e, o_x is 0).

A (homogeneous) point $\tilde{\mathbf{p}}$ in the world frame of the SOR (typically, on its surface) maps to a point $\tilde{\mathbf{p}}'$ on the image as

$$\begin{bmatrix} p'_x p'_w \\ p'_y p'_w \\ p'_z p'_w \\ p'_w \end{bmatrix} = \mathbf{M}_{\text{vp}} \mathbf{M}_{\text{can}} \mathbf{M}_{\text{persp}} \mathbf{M}_{\text{cam}} \tilde{\mathbf{p}} \tag{1}$$

where the matrix product is a typical "modelview" transform [20]. $\mathbf{M}_{\text{vp}}$, the viewport transform, depends on the (known) image horizontal and vertical resolutions. $\mathbf{M}_{\text{can}}$, the canonical view transform, depends on the camera's (unknown) field of view and (known) image aspect ratio. $\mathbf{M}_{\text{cam}}$, the camera transform, depends on the (unknown) camera pose. $\mathbf{M}_{\text{persp}}$, the perspective transform, also depends on the camera pose.

As the base of the SOR is at the origin, a uniform scale of all coordinates produces no change in the image, so length units are arbitrary. We therefore choose our length unit to be the SOR's height. The top of the SOR is therefore at $y = 1$.

We can determine the camera's roll and yaw from the silhouette alone. In the case of roll, we use principal component analysis [12] to find the silhouette's major and minor axes. The departure of one of them (usually the major) from the vertical gives us the roll. Assuming that the center of the image corresponds to the optical axis, the yaw may then be computed trigonometrically.

We therefore have the following six unknown parameters: two components of the camera's position, o_y and o_z, the camera's vertical field of view, θ_y, the camera's pitch, β_{cam}, and the radii of the base and top cross sections, r_{base} and r_{top}.

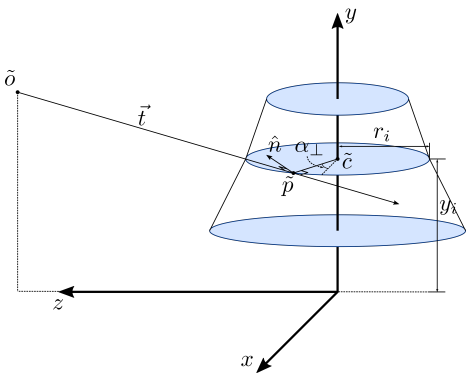

Fig. 5. Viewing Ray-Silhouette Intersection

We initially tried to fit all parameters at once but this approach proved not to be robust. Instead, we determine their values using three multivariate "fits", as shown in Figure 4. Each of them uses Nelder-Mead optimization [15] to yield all or a subset of the parameters. The first fit applies to the base sub-silhouette, yielding values for all parameters except r_{top}. The second fit applies to the top sub-silhouette, yielding values for all parameters except r_{base}. The third fit applies to whole ellipses based on values from the first two fits and robustly yields all parameters.

Geometry Extraction We next extract the SOR's geometry. Due to its axial symmetry, this is simply the radius as function of height, i.e. $r = r(y)$. To derive this from the silhouette, we first treat it as a set of horizontal extrema $\{w_j\}$. For a given row of pixels j, w_j is the distance between the left and right extrema of the silhouette. We also start with a given N $\{y_i, r_i\}$ "cross section" samples of $r(y)$, $(0 \leq i < N)$. The y_i's are uniformly spaced. The (unknown) r_i's are initially zero.

It is important to keep in mind that not every row of the silhouette contributes to $r(y)$: The top or base may obstruct some part of the geometry. In an extreme case, imagine a conical SOR viewed from a great height ($o_y \gg 1$). In that case, the silhouette would only show the elliptical base of the SOR and no radii could be extracted.

Looking at Figure 5, valid w_j's that contribute to $r(y)$ arise in situations where a viewing ray is tangential to the normal $\hat{\mathbf{n}}$ at a "horizon point" $\tilde{\mathbf{p}}$ on the surface of the SOR. This corresponds to an angle $\alpha_\perp$, a particular value of the azimuthal angle α measured clockwise from $\hat{\mathbf{x}}$ in the xz plane. All pixels with $\alpha_\perp \leq \alpha \leq \pi - \alpha_\perp$ are visible.

In the orthographic view ($o_z \to \infty$), $\alpha_\perp = 0$. Or if the SOR were a cylinder, $\alpha_\perp$ would be a constant for any camera position. But for a general SOR, $\alpha_\perp$ depends on $\hat{\mathbf{n}}$, which depends on $\widehat{\mathbf{N}} = (N_x, N_y, 0)$, the "unrotated" normal of the SOR in the xy plane (for $x \geq 0$, the right side of the image). In [13], we derive

$$\alpha_\perp = \arcsin\left[\frac{r_i N_x + N_y(y_i - o_y)}{o_z N_x}\right] \tag{2}$$

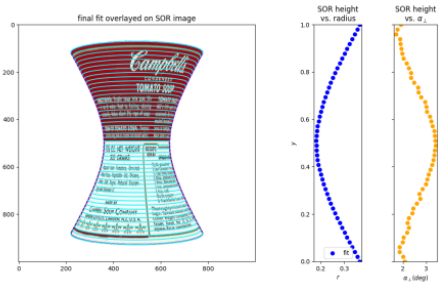

Fig. 6. Geometry Extraction

There are two complications: First, $\widehat{\mathbf{N}}$ depends on the derivative of $r(y)$, the quantity we're trying to solve for. To approximate $\widehat{\mathbf{N}}$ at a cross-section i, we compute the normal to parametrically-fit parabolas based on previous estimates of (y_{i-1}, r_{i-1}), (y_i, r_i), and (y_{i+1}, r_{i+1}), so solving for r_i requires r_{i-1} and r_{i+1}. This requires the solution to be iterative. The second complication arises if the absolute value of the arcsin argument in Equation (2) exceeds 1. Then the camera cannot see that horizon point, so that value of r_i is invalid, at least on that iteration.

To solve for $\{r_i\}$, we developed the *scaled difference method*. Starting with an initial $\{r_i\}$, we project these to a set of silhouette widths $\{w_i'\}$ and use the departures of these from the known $\{w_i\}$s to produce a new $\{r_i\}$. When $\{w_i'\}$ falls within a given (RMSD) tolerance of the $\{w_i\}$, we have our $\{r_i\}$. It was necessary to incorporate an empirically-determined scaling factor which slows the rate of convergence in order to reduce the chance for the $\{r_i\}$'s to produce impossible $\alpha_\perp$ values. Figure 6 shows typical $\{r_i\}$ and $\{\alpha_{\perp,i}\}$ results.

Texture Extraction The inputs to texture extraction are the camera calibration and the SOR geometry. The output is a texture of any desired resolution. Figure 7 shows a typical result.

Fig. 7. Texture Extraction

Rows correspond to varying α values, where $\pi \geq \alpha \geq 0$. Since we only have data for $\alpha_\perp \leq \alpha \leq \pi - \alpha_\perp$, we use the transparency channel to indicate valid pixels.

Figure 7 shows a typical result using a synthetic image of known geometry. Height corresponds to the vertical path length along the side of the SOR. We expect a certain amount of distortion at the edges and where the SOR is narrow. These correspond to regions in the original image where the details may be difficult to see.

Texture Stitching We have shown that we can capture the geometry of an SOR from a single image, but that image can show at most half of the SOR's texture. It may be desirable to combine, or "stitch", texture data from multiple images into a single texture that can be used, for instance, to re-create a textured 3D model viewable from any direction, to 3D print a replica of the SOR, or to display the whole of the SOR's surface in 2D media (an archaeological research publication, say).

So far, we have assumed that each image has its own coordinate system: The SOR's base is at the origin, its axis of symmetry is the y axis, and the camera lies above the z axis. To stitch images together, we would further need to know the relative azimuthal (α) angular differences of the images, but these are not provided. Each texture being a parameterization of path length vs. α, there should ideally be an exact match of the region of the SOR visible in both textures when they are overlaid with the proper vertical[1] and horizontal offsets.

Goshtasyby's text [10] on image registration contains detailed descriptions of algorithms for this, some of which have implementations available in computer vision libraries. However, the type of registration we need to do is simplified by the fact that there should be no rotation, scaling, or any complex type of homology necessary. Alignment for stitching two textures can therefore be done simply by finding horizontal and vertical displacements that minimize differences (here, RMSD) between the overlapping regions. Figure 8 shows this.

[1] Due to variations in camera pitch (β_{cam}), textures need to be aligned vertically as well as horizontally, if only slightly. Figure 8 exaggerates this for illustration.

Fig. 8. Texture Stitching

4 Implementation

We implemented SORGATE as an application program *sorgate* in the `Python` programming language, which supports a number of packages that were extremely useful. We used `NumPy` for array computations, `SciPy` for optimization, and `OpenCV` for image I/O, segmentation, filtering, edge detection, and contour extraction.

We built its user interface with `tkinter` and `matplotlib`. We rendered textures using `OpenGL`, `PIL`, and `GLFW`. Space does not allow us to describe it in detail, but Figure 9 shows its general operation. We also used it to create many of the other images in this paper.

5 Evaluation

Synthetic Images The images shown in Figures 6 and 7 were created with a custom-built raytracer at a resolution of 1000×1000 pixels. This allowed us to compare the parameters and $\{r_i\}$ values *sorgate* extracted with the raytracer inputs and guaranteed that the object itself was a true SOR.

For these images we used a uniform background, so segmentation and silhouette extraction were trivially correct. Extracted and true camera parameters differed by no more than a fraction of a degree (for the angles) few percent (for the positions) in the worst cases. Figures 6 to 9 show typical results for geometry, texture extraction, and texture stitching. Figure 10 shows the quick convergence of the $\{r_i\}$ fit results.

Real Images The goal of SORGATE is to extract data from real images, so those are a critical part of its evaluation. For segmentation and silhouette extraction, the *GrabCut* and Canny algorithms performed well on real images, as shown in Figure 3. Extracted camera roll and yaw parameters were, again, within fractions of a degree.

Fig. 9. Overview of the *sorgate* Application. This illustrates the data flow shown in Figure 2.

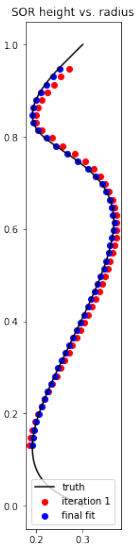

Fig. 10. $\{r_i\}$ Fit Results

More complicated images, such as those with SORs whose colors were similar to the background, or those with lighting that reduced the prominence of silhouette, required a lot more time spent making precise touch-ups and fine-tuning the Canny parameters to produce meaningful results; in these scenarios, the early stages of our method are quite cumbersome.

Figure 1 shows the result: We use stitched textures extracted from several real images used to create a 3D model of the original object. Although largely successful, certain images provided more of a challenge for the camera calibration parameters. SORs with rounded bases complicated sub-silhouette extraction, so it was necessary to adjust the base and top radii manually. Problems also arose with SORs where the viewing ray intersects both the stem and base, such as a goblet viewed from a high camera position. For this reason, we must, as we've done in Figure 10, limit the range of $\{r_i\}$ values.

Problems arose from stitching as well, as shown in Figure 11 (the same vase shown in Figure 1). Directional lighting and dealing with a glossy surface with large uniform patches confused the simple RMSD metric and we had to make several ad-hoc assumptions about the horizontal and vertical search ranges to make this work.

Before After

Fig. 11. Stitching Two Textures Taken from Real Images

6 Conclusions and Future Work

We have shown that SORGATE is capable of extracting geometry and texture from a single SOR image and can combine multiple images of the same SOR into a single texture.

It worked well with synthetic images, but real images required more work. Segmentation, silhouette, and sub-silhouette extraction needed more user input, but the resulting camera calibration and geometry and texture extractions for individual images were satisfactory. Stitching multiple images together required more *ad hoc* adjustments to account for lighting and the SOR's material properties.

Future development of SORGATE should address these problems. We should be able to take better advantage of the SOR's symmetry and the smoothness of the silhouette (except at the sub-silhouette limits). Also, once we have an initial model of the SOR's geometry, we should be able to incorporate a lighting model to remove both diffuse and specular effects and retrieve the underlying material properties for use in rendering the SOR with arbitrary lighting.

The incorporation of the projective geometry approach (as in [6, 16]) would also be likely to increase robustness and move us towards our goal of being able of extracting geometry and texture with a minimum of user intervention.

References

1. Overbeck Vase, ca 1920 | Antiques Roadshow | PBS (2017), `https://image.pbs.org/video-assets/pbs/antiques-roadshow/256002/images/mezzanine_222.jpg.crop.379x212.jpg`
2. Boyer, E., Berger, M.O.: 3D Surface Reconstruction Using Occluding Contours. International Journal of Computer Vision 22(3), 219–233 (1997)
3. Boykov, Y., Jolly, M.P.: Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. Proceedings Eighth IEEE International Conference on Computer Vision(ICCV 2001) 1, 105–112 (2001)
4. Canny, J.: A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8(6), 679–698 (1986)

5. Chellali, R., Fremont, V., Maaoui, C.: A New Approach to 3-D Modeling of Objects With Axial Symmetry. IEEE Transactions on Industrial Electronics 50(4), 1–7 (2003)
6. Colombo, C., Del Bimbo, A., Pernici, F.: Uncalibrated 3D metric reconstruction and flattened texture acquisition from a single view of a surface of revolution. Proceedings - 1st International Symposium on 3D Data Processing Visualization and Transmission, 3DPVT 2002 27(1), 277–284 (2002)
7. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and Rendering Architecture from Photographs: A hybrid geometry-and image-based approach. Tech. rep. (1996)
8. Fitzgibbon, A., Pilu, M., Fisher, R.B.: Direct Least Square Fitting of Ellipses. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(5), 476–480 (1999)
9. Ghosh, S., Das, N., Das, I., Maulik, U.: Understanding deep learning techniques for image segmentation. ACM Computing Surveys 52 (8 2019)
10. Goshtasby, A.A.: Image Registration: Principles, Tools and Methods. Springer Publishing Company, Incorporated (2012)
11. Jain, A.: Automatic 3D Reconstruction for Symmetric Shapes (2016)
12. Jolliffe, I.: Principal Component Analysis. Springer Verlag (1986)
13. Ledesma, A.: SORGATE: Surface of Revolution Geometry and Texture Extraction. Master's thesis (5 2021)
14. Liu, C., Hu, W.: Ellipse fitting for imaged cross sections of a surface of revolution. Pattern Recognition 48(4), 1440–1454 (2015), http://dx.doi.org/10.1016/j.patcog.2014.09.028
15. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal 7(4), 308–313 (1965)
16. Pernici, F.: Two Results in Computer Vision using Projective Geometry. Ph.D. thesis, University of Florence (2005), https://www.micc.unifi.it/pernici/index_files/PhdThesis.pdf
17. Ponce, J., Chelberg, D., Mann, W.B.: Invariant Properties of Straight Homogeneous Generalized Cylinders and Their Contours. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(9), 951–966 (1989)
18. Puech, W., Chassery, J.M., Pitas, I.: Cylindrical surface localization in monocular vision. Pattern Recognition Letters 18(8), 711–722 (1997)
19. Rother, C., Kolmogorov, V., Blake, A.: "GrabCut". ACM Transactions on Graphics 23(3), 309–314 (2004)
20. Shirley, P., Marschner, S.: Viewing. In: Fundamentals of Computer Graphics, chap. 7, pp. 141–159. A K Peters/CRC Press, Boca Raton, 3rd edn. (2009)
21. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer (2011)
22. Tsai, R.Y.: A Versatile Camera Calibration Techniaue for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses ROGER. IEEE Journal on Robotics and Automation 3(4), 323–344 (1987)
23. Ulupinar, F., Nevatia, R.: Shape from Contour: Straight Homogeneous Generalized Cylinders and Constant Cross Section Generalized Cylinders. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(2), 120–135 (1995)
24. Wong, K.Y.K., Mendonça, P.R., Cipolla, R.: Reconstruction of surfaces of revolution from single uncalibrated views. Image and Vision Computing 22(10 SPEC. ISS.), 829–836 (2004)
25. Worring, M., Smeulders, A.W.: Digital Curvature Estimation. Computer Vision and Image Understanding 58(3), 366–382 (1993)
26. Zhang, M., Zheng, Y., Liu, Y.: Using silhouette for pose estimation of object with surface of revolution. Proceedings - International Conference on Image Processing, ICIP pp. 333–336 (2009)