SHELLCAM: INTERACTIVE GEOMETRY-AWARE VIRTUAL CAMERA CONTROL

Tamy Boubekeur

Telecom ParisTech CNRS LTCI Institut Mines-Telecom

(Author's Draft - Final version to be published in the proceedings of the IEEE International Conference on Image Processing 2014)

ABSTRACT

We introduce *ShellCam*, a geometry-aware virtual camera control model which defines a smooth motion subspace enabling Pan&Zoom navigation on arbitrary 3D objects. The basic idea is to define a scale-dependent offset *shell* around the visible geometry which provides, at any point, a meaningful tangent direction for panning and helps computing the camera-object distance to rule accurately a logarithmic zoom motion. We define the underlying motion space as a visualization hull and evaluate it on-the-fly using a moving least-squares approach. As a result, ShellCam provides smooth object-aware 3D motions, combining rotations and translations, based on a simple 2D user input such as typically produced by mouse motions. We also provide an efficient GPU implementation which makes use of the standard rasterization pipeline to compute this 3D motion efficiently. Our approach is robust to inconsistent geometry such as point clouds or polygon soups, works on shapes with complex topology, does not require any pre-computation and can be used on dynamic data. ShellCam offers a convenient control for 3D inspection tasks and a transparent swap with other control models for more general 3D navigation. Last, our model is straightforward to integrate in any 3D application.

Index Terms— Virtual camera control model, interactive navigation, 3D user interfaces, GPU algorithms.

1. INTRODUCTION

Interactive shape inspection is an important task in computer graphics. Typical 3D tools allow to "walk around" 3D models using a simple camera control model (CCM) – such as the virtual trackball for instance – to map 2D user input (e.g., mouse or touch screen) onto 6 degrees of freedom (DoF) camera motions. Unfortunately, even for expert users, such models imply a tedious manual sequence of rotations and translations to actually reach a desired inspection point of view (PoV).

We propose ShellCam, a "geometry-aware" CCM which uses a view-dependent motion space deduced from the observed geometry. Building upon point-based shape analysis, we introduce the notion of *Visualization Hull* to define a dynamic view- and scale-dependent *shell* subspace for navigating around surfaces or closely inspecting them (see Fig 1). This mimics the popular, novice-user-friendly, *Pan&Zoom* mechanism present in online mapping systems (e.g., GoogleMap). As a result, our CCM allows easy navigation along complex trajectories around a shape and avoids "lost in space" pitfalls. On the contrary to other advanced CCMs, ShellCam does not make any assumption about the scene (geometry, topology, representation), does not require any pre-computed data structure, is naturally GPU-supported and is fully compatible with others CCMs.

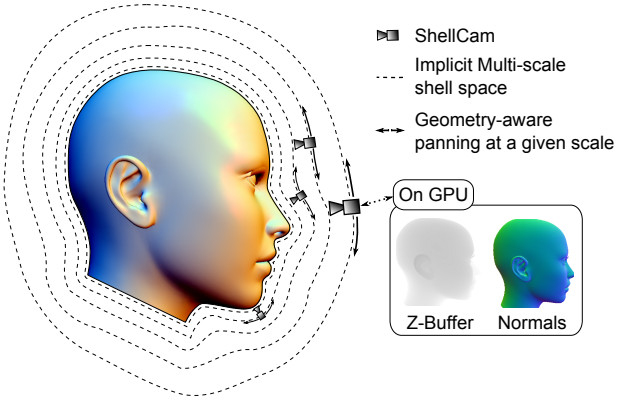

Fig. 1. **Overview.** A navigation shell is defined from the scene's geometry and the camera PoV using a dynamic point-based MLS approximation. Consequently, our ShellCam provides a scale-dependent Pan&Zoom metaphor by mapping the 2D user input onto this local, scale-dependent motion subspace.

2. PREVIOUS WORK

Camera control [1] is a broad area covering topics which can be as various as optimization, eye tracking and video games. We focus on CCMs which are the most similar to ours: CCMs for interactive surface inspection.

Beyond the virtual trackball metaphor, recently improved by Henriksen [2], and which offers a 6-DoF control by mapping a mouse motion combined to a key control onto a subset of the 6 dimensions, a number of CCM interaction metaphors have been proposed and classified [3], such as *orbiting and flying* [4], *eye-ball-in-hand* [5], *through-the-lens control* [6] or *virtual sidewalks* [7].

For close inspection tasks, the HoverCam system [8] al-

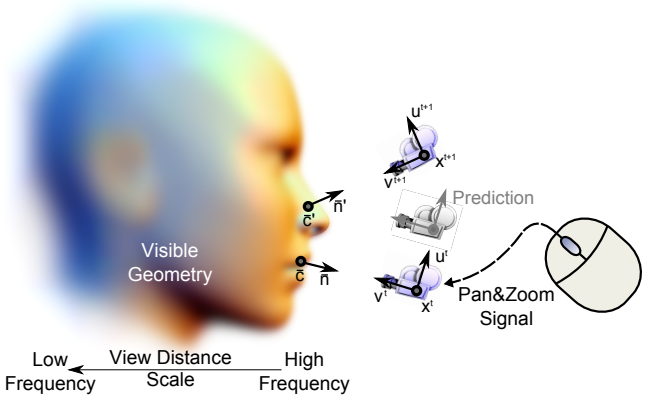

Fig. 2. **ShellCam Motion.** The user input signal is converted to a predicted motion from which the next frame visualization hull is evaluated and used to position the camera.

lows to track surfaces by mapping mouse interaction according to the observed model, switching between *egocentric* (object centered) and exocentric (world centered) control. Basically, HoverCam encapsulates all motion dimensions into a search procedure where the user input signal defines a direction into which "searching" for the next point to observe on the surface. Special rules are applied to prevent unnatural behavior around concavities and sharp corners. The search procedure depends on a static data structure which strongly limits scalability and the ability to handle dynamic scenes. As the camera-object distance does not influence the search procedure, HoverCam is sensitive to high-frequency geometric content and noisy data. Last, the search procedure supposes an invasive implementation, with direct access to the application scene graph.

When examining closely an object, points-of-interest can lead the camera motion, such as with the UniCam [9]. Similarly, in the context of mobile interaction, Navidget [10] offers a point-and-focus interface where the user can instantly define a location and an incident view direction on a 3D surface.

ScrutiCam [11] is a simpler system where the user points and clicks on the surface to enforce camera alignment to the surface normal. Although perfectly adapted to low-end devices, such models lack a scale analysis and may lead to erratic motions in the presence of high-frequency geometry.

More advanced scene analysis methods help designing camera motions. For instance, the ShowMotion system [12] allows designing cinematic sequences from a set of predefined camera motion. *Constraint optimization* [6] can also be used to define indirectly the camera parameters from the image content.

In this paper, our aim is similar to the one developed in HoverCam: providing a CCM which runs interactively and derives its motion from the scene's geometry to simplify user input, while providing a view angle being as "orthogonal" as possible to the overall observed surface.

3. MODEL

Principle: Basically, the idea behind the ShellCam CCM is to move the camera in a "fair" subspace near the object. By fair, we mean smooth, not sensitive to high frequencies, taking into account the scale at which the shape is perceived and offering orthogonal exploration (*pan*) and closer inspection (*zoom*) control primitives. Our key idea is to model all these constraints as a subspace which can be intuitively seen as a low frequency offset of the visible scene's geometry. We call this object a *Visualization Hull* (VH) and use it to define both *Pan&Zoom* interactions (Fig. 1).

The geometry-aware *pan* operator is defined as a tangent motion on the VH and is typically mapped from the mouse (or fingers for touch screens) movement.

The geometry-aware *zoom* operator offers progressive close-up allowing users to zoom with a decaying speed when getting closer to the geometry. Thus, we map the zoom control onto a logarithmic interpolation between the VH (where the camera is located) and the observed geometry itself. This control is typically mapped from the mouse middle button/wheel (or from the touch screen's *pinch* gesture).

These two camera controls are ultimately expressed as rigid transformations of $\mathcal{C}^t = \{\mathbf{x}^t, \mathbf{v}^t, \mathbf{u}^t, \mathbf{r}^t\}$ with $\mathbf{x}^t$ the camera's center, $\mathbf{v}^t$ the view direction and $\mathbf{u}^t/\mathbf{r}^t$ the up/right vectors at frame t.

Motion: Let $\mathcal{P} = \{\{\mathbf{p}_0, \mathbf{n}_0\}, ..., \{\mathbf{p}_m, \mathbf{n}_m\}\}$ be a sampling of the scene's geometry with $\mathbf{p}_i \in \mathbb{R}^3$ the position and $\mathbf{n}_i \in \mathbb{R}^3$ the normal of the $i-th$ sample. We consider $\mathcal{P}$ dense enough so that it provides a good approximation of the scene. The user's control signal is defined as $\mathbf{k}^t = \{\mathbf{d}^t, s^t\}$ with $\mathbf{d}^t \in \mathbb{R}^2$ the 2D pan signal and $s^t \in \mathbb{R}$ the zoom signal.

We define the ShellCam motion as:

$$\mathcal{C}^{t+1} = \mathcal{V}^{\mathcal{P}}_{\mathbf{k}^t}(\mathcal{C}^t)$$

This motion must account for the current distance and scale – toning down the influence of small scale features w.r.t. the current PoV – and provide smooth movements. We propose to define the supporting smooth subspace VH of $\mathcal{V}$ by means of a view-dependent shell approximation (see Fig. 2).

First, starting from $\mathcal{C}^t$, we compute a weighted centroid $\overline{\mathbf{c}}$ and normal $\overline{\mathbf{n}}$ from $\mathcal{P}$:

$$\overline{\mathbf{c}} = \frac{\sum_i \omega^{\mathbf{v}^t}_{\mathbf{x}^t}(\mathbf{p}_i)\Pi^{\mathbf{n}_i}_{\mathbf{p}_i}(\mathbf{x}^t)}{\sum_i \omega^{\mathbf{v}^t}_{\mathbf{x}^t}(\mathbf{p}_i)} \text{ and } \overline{\mathbf{n}} = \frac{\sum_i \omega^{\mathbf{v}^t}_{\mathbf{x}^t}(\mathbf{p}_i)\mathbf{n}_i}{||\sum_i \omega^{\mathbf{v}^t}_{\mathbf{x}^t}(\mathbf{p}_i)\mathbf{n}_i||}$$

with $\Pi^{\mathbf{n}}_{\mathbf{p}}(\mathbf{x})$ the orthogonal projection of $\mathbf{x}$ onto the tangent plane at $\mathbf{p}$. Using this projection makes the combination *Hermitian* which has proven to better preserve convexity than linear combinations [13]. We compute weights in a view-dependent fashion by centering an anisotropic weighting kernel ω at $\mathbf{x}^t$. This kernel is defined as the product of two kernels: one accounting for distance in image space (i.e. samples near the screen border have less impact) and one accounting

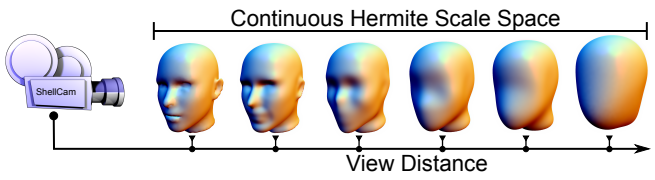

Fig. 3. **Visualization Hull.** A different shape scale is considered according to view distance. Here, we illustrate what would be the visualization hull if completely reconstructed from many views scatterred among 6 distances. This implicitly defines an hermite scale-space.

for 3D distances to the camera (i.e., far away samples have less importance than nearby ones). Gaussian kernels can be used safely in both cases. Second, we compute $\alpha = ||\mathbf{x}^t - \overline{\mathbf{c}}||$ which captures the camera distance to the visible geometry at the current scale. Third, we predict a tangent motion from $\mathcal{C}^t$ by computing

$$\mathbf{x}' = \mathbf{x}^t + \mathbf{d}_0.\mathbf{r}^t + \mathbf{d}_1\mathbf{u}^t$$

with $\overline{\mathbf{c}}'$ and $\overline{\mathbf{n}}'$ defined accordingly.

Finally, we compute the updated camera position

$$\mathbf{x}^{t+1} = \overline{\mathbf{c}}' + f_{s^t}(\alpha) * \overline{\mathbf{n}}'$$

with f a logarithmic zoom function, $\mathbf{v}^{t+1} = -\overline{\mathbf{n}}'$, $\mathbf{u}^{t+1}$ an orthogonal vector to $\mathbf{v}^{t+1}$ which minimizes torsion with $\mathbf{u}^t$ and $\mathbf{r}^{t+1} = \mathbf{v}^{t+1} \otimes \mathbf{u}^{t+1}$. Users can optionally define a "global" up direction anytime and the ShellCam will conform minimal $\mathbf{u}$-torsion to it while maintaining $\mathcal{C}$ ortho-normal. This is particularly useful for objects having a natural up-right orientation [14].

4. ANALYSIS

Our motion evaluation procedure defines an underlying subspace VH which captures low frequencies of the observed geometry from a given PoV. Intuitively, this subspace is a view-dependent *offset shell*. More precisely, our camera centered weighted combination of $\mathcal{P}$ is inspired by the *moving least-squares* approximation [15] in its simplified form [16] and automatically adjusts the frequency content of camera movements w.r.t. (visible) samples. We experimented with different weighting kernels and observed highest regularity using a Hermitian combination (see Fig. 3). Moreover, smoothly decaying kernels ensure that global/low frequency trajectories are produced for distant PoVs while local/high frequency modulations are progressively inserted when getting closer to the geometry. In practice, a single horizontal user pan signal may either result in a simple rotation when observing the object remotely, or in a more complex trajectory when zooming closer to it. Note that the VH is **never** entirely reconstructed but only implicitly evaluated at a single location: the camera center.

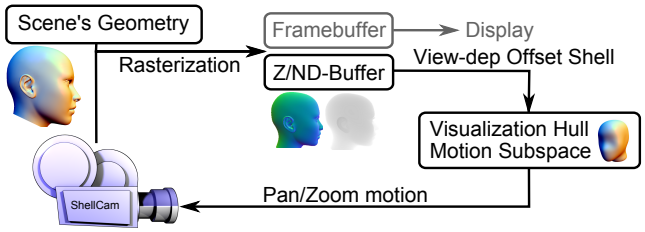

Fig. 4. **Screen Space GPU Motion Evaluation.** Each rendered frame provides a view-dependent scene's sampling for the next camera motion thanks to its Z- (or ND-)Buffer.

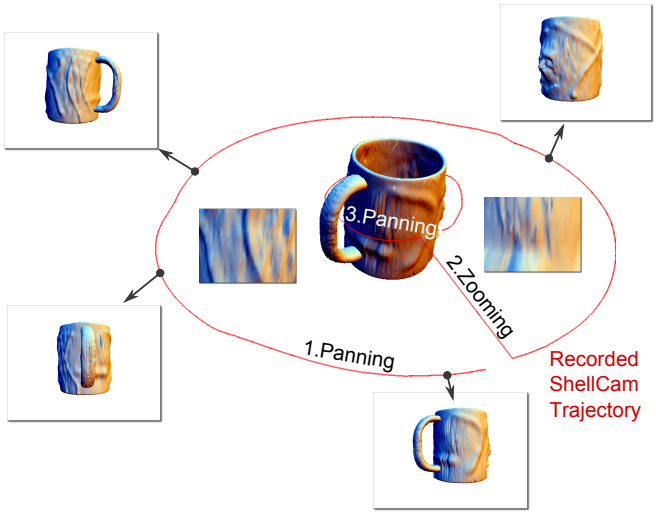

Fig. 5. **View-Dependent Motion**. Here, we recorded the camera trajectory (in red) under ShellCam motion for a pan/zoom/pan sequence. Note how the motion mimics an orbit when performed from a large enough distance.

5. IMPLEMENTATION

While a pure CPU implementation can cope with a dynamic sampling $\mathcal{P}$ at 60Hz or more for scenes made of a few hundreds thousands triangles, we reach much higher performances by exploiting the GPU rendering pipeline: indeed, the rasterization's Z-Buffer already provides a camera-centered sampling of the visible geometry (see Fig 1). Therefore we compute $\mathcal{P}$ from it and either its derivatives in screen space or an extra deferred shading buffer for normals (ND-buffer). In practice, a parallel GPU kernel is executed to perform the motion evaluation (see Fig. 4). On a modern workstation, this GPU evaluation has a negligible computational footprint compared to (even basic) rendering. Nevertheless, we can easily adapt it to a limited horsepower (e.g., mobile device) by simply subsampling $\mathcal{P}$. As we seek for smooth, low frequency motions, this can be done safely up to a strong subsampling ratio. In our GPU implementation, this boils down to a uniform ND-buffer sub-sampling process.

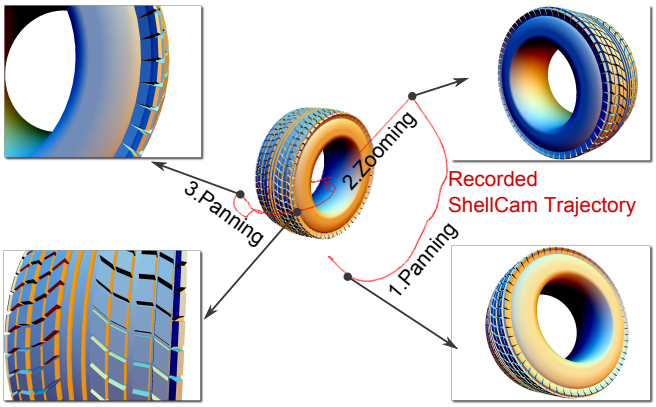

Fig. 6. **Multiscale**. Higher frequencies are introduced progressively in the trajectory (in red) when getting closer.

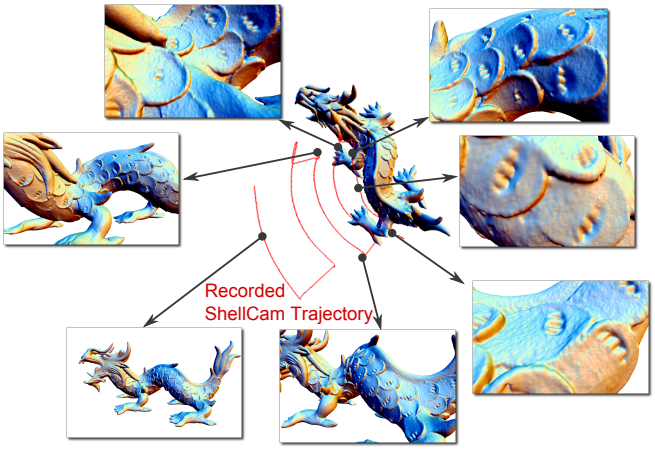

Fig. 7. **Scalability**. Our GPU implementation allows to use ShellCam on large models such as this 7.2M tri. Dragon.

6. RESULTS

The primary focus of our work is object inspection. For this task, our approach exhibits a number of useful properties. First, the observed shape may have an arbitrary topology, the motion will naturally adapt to it (see Fig. 5). Second, geometric features can appear at different scales: our motion procedure avoids jaggy trajectories when curvature varies quickly in image space (see Fig. 6). Third, as our approach is free from any search process or pre-computation and builds upon the hardware supported standard graphics pipeline, it scales easily to large, potentially dynamic scenes (see Fig. 7). Last, even a sparse polygon soup (e.g. a tree) is enough to evaluate the visualization hull and preserve a smooth motion (see Fig. 8). We illustrate the ShellCam behavior compared to models which are blind to geometry (see Fig. 9). While such CCMs provides the same motion whatever the scene and PoV, our ShellCam provides geometry-aware panning motions conforming to visible shapes. Note that ShellCam

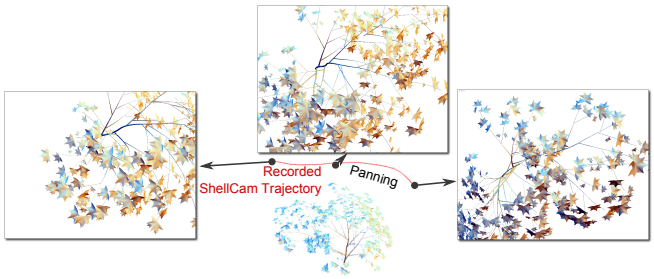

Fig. 8. **Robustness**. Even on a polygon soup as chaotic as this tree mesh, ShellCam panning remains a smooth motion.

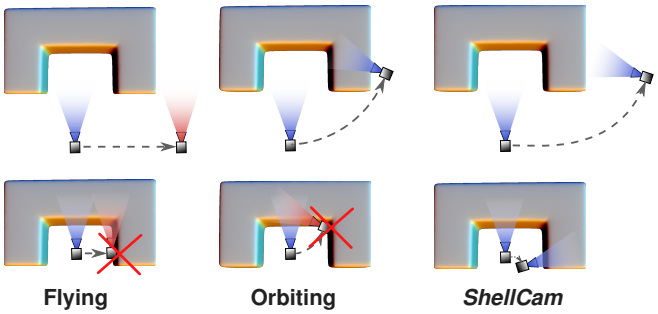

Fig. 9. **Motion comparison**. Moving according to a 1D "pan" user signal and mapping it to 2 classical motions (left and middle) and ShellCam motion (right).

can be (des)activated at any time during inspection, allowing a transparent swap to and from other CCMs. Consequently, we believe that ShellCam is more a complementary model to other CCMs than an alternative. Indeed, this opens an interesting direction for future work on "meta" CCMs, blending and combining efficiently different models [9] according to higher level analysis of both scenes and users.

7. CONCLUSION

ShellCam offers an intuitive virtual camera control to users, with complex trajectories easily defined through the ubiquitous *Pan&Zoom* metaphor. By computing a geometry-aware motion on-the-fly, ShellCam works on any dynamic scene, including non-consistent geometry, and its GPU implementation allows using it on large models as well. Last, the user can swap between ShellCam and other CCMs (e.g., trackball) anytime. Although ShellCam cannot replace full 6-DoF CCMs in general, it represents an efficient alternative for model inspection tasks.

Acknowledgements. This work has been partially funded by the European Commission under contracts FP7-323567 HARVEST4D and FP7-287723 REVERIE, and by the ANR iSpace&Time project.

8. REFERENCES

[1] Marc Christie, Patrick Olivier, and Jean-Marie Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008.

[2] Knud Henriksen, Jon Sporring, and Kasper Hornbaek, "Virtual trackballs revisited," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 2, pp. 206–216, 2004.

[3] D. Bowman, D. Johnson, and L. Hodges, "Testbed environment of virtual environment interaction.," in *Proc. ACM symposium on Virtual Reality Software and Technology*, 1999, pp. 26–33.

[4] Desney S. Tan, George G. Robertson, and Mary Czerwinski, "Exploring 3d navigation: combining speed-coupled flying with orbiting," in *Proc. ACM SIGCHI*, 2001, pp. 418–425.

[5] Colin Ware and Steven Osborne, "Exploration and virtual camera control in virtual three dimensional environments," in *Proc. ACM Symposium on Interactive 3D (I3D)*, 1990, pp. 175–183.

[6] Michael Gleicher and Andrew Witkin, "Through-the-lens camera control," in *Proc. ACM SIGGRAPH*, 1992, pp. 331–340.

[7] Andrew J. Hanson and Eric A. Wernert, "Constrained 3d navigation with 2d controllers," in *Proc. IEEE Visualization*, 1997, pp. 175–ff.

[8] Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach, "Hovercam: interactive 3d navigation for proximal object inspection," in *Proc. ACM Symposium on Interactive 3D (I3D)*, 2005, pp. 73–80.

[9] Robert Zeleznik and Andrew Forsberg, "Unicam, 2d gestural camera controls for 3d environments," in *Proc. ACM Symposium on Interactive 3D (I3D)*, 1999, pp. 169–173.

[10] Martin Hachet, Fabrice Dècle, Sebastian Knödel, and Pascal Guitton, "Navidget for easy 3d camera positioning from 2d inputs," in *Proc. IEEE Symposium on 3D User Interfaces (3DUI)*, 2008, pp. 83–89.

[11] Fabrice Dècle, Martin Hachet, and Pascal Guitton, "Scruticam : Camera manipulation technique for 3d objects inspection," in *Proc. IEEE Symposium on 3D User Interfaces (3DUI)*, 2009, pp. 19–22.

[12] Nicolas Burtnyk, Azam Khan, George Fitzmaurice, and Gordon Kurtenbach., "Showmotion: Camera motion based 3d design review.," in *Proc. ACM Symposium on Interactive 3D (I3D)*, 2006, pp. 167–174.

[13] Marc Alexa and Anders Adamson, "Interpolatory point set surfaces—convexity and hermite data," *ACM Transactions on Graphics*, vol. 28, no. 2, pp. 1–10, 2009.

[14] Hongbo Fu, Daniel Cohen-or, Gideon Dror, and Alla Sheffer, "Upright orientation of man-made objects," *ACM Transactions on Graphics*, pp. 1–7, 2008.

[15] David Levin, "The approximation power of moving least-squares," *Math. Comput.*, vol. 67, pp. 1517–1531, 1998.

[16] M. Alexa and A. Adamson, "On normals and projection operators for surfaces defined by point sets," in *Proc. Eurographics Symposium on Point-based Graphics*, 2004, pp. 149–156.