Mesh Alignment Using Grid Based PCA

David Kaye, Ioannis Ivrissimtzis

Keywords:
Mesh alignment; Principal Component Analysis; Grid based methods

Abstract:
An algorithm is presented to align one mesh to another by means of a regular 3D grid upon which Principal Components Analysis (PCA) is performed. The use of a 3D lattice external to both inputs increases the robustness of PCA, particularly when dealing with meshes of different and possibly uneven vertex density. The proposed algorithm was tested on meshes that have undergone standard mesh processing operations such as smoothing, simplification and remeshing. In several cases the results indicate an improved robustness compared to performing PCA directly on mesh vertices.

1 INTRODUCTION

It is often useful to compare two meshes, point clouds, or a mixture of the two. For example, in shape recognition one might compare two meshes or two point clouds asking whether they belong to the same category.

The comparison of two meshes usually starts with their alignment. It should be noted that the exact way in which two meshes are best aligned is not clearly defined. In fact, the best alignment can be application-dependent, and mesh alignment should be seen as an ill-posed problem. Nevertheless, a good alignment algorithm is expected to be able to align a mesh with a version of it that has undergone common mesh processing operations such as smoothing, simplification or remeshing.

Alignment is typically done by computing a translation, a scaling and a rotation, which are then applied to one of the meshes to align it with the other. The computation of the translation is usually done by aligning barycentres, while the scaling is done by aligning bounding boxes or bounding spheres (Sfikas et al., 2014). These are both considered less challenging to compute than the rotation, which is the focus of this paper.

The simplest and most widely used method of aligning meshes aligns the principal axes of the mesh vertex sets. Despite its popularity, it is well documented that in demanding applications such as shape recognition the results of this alignment method may not be satisfactory, especially in the case when the mesh undergoes processing that potentially disturbs the distribution of mesh vertices such as simplification and remeshing.

One way around this problem is to voxelise the mesh and then apply an alignment algorithm for volumetric data. However, such a method can be computationally demanding, and the cost of the voxelisation cannot be fully justified if it is used for mesh alignment only. A second approach would be to apply PCA not on the mesh vertex set but on a more uniform point set produced by a mesh sampling method. However, such a method would depend on the quality of the triangulation and for example, a large number of long thin triangles in the mesh could cause problems.

Here, we propose a solution in between the above two approaches, that is, a mesh sampling method which, without being a fully volumetric method, is based on creating a point sample on a regular grid and then performing PCA on that point sample.

1.1 Contribution

This paper presents a novel method for mesh alignment based on performing PCA on a point sample on a regular grid. The algorithm is deterministic and does not depend on any initial orientation or any other prior knowledge.

The algorithm can also process point clouds as inputs, allowing point clouds to be aligned with meshes, or two point clouds to be aligned.

Connectivity information is used to provide robustness in the face of poorly meshed objects, for instance, those with large triangles.

Should large meshes need to be aligned, memory could be saved by running the algorithm in a layered, or even striped fashion, as proposed in (Kaye and Ivrissimtzis, 2011).

1.2 Limitations

The main limitation of the algorithm is that since PCA is applied to a set of points from a relatively coarse grid, the method is less accurate than PCA on mesh vertices when the two meshes are identical or almost identical. In a sense this larger error is a trade-off for the increased robustness and our experiments show that it is in a range that in most applications would be considered tolerable.

Since the method is underpinned by PCA, it is not particularly well suited for inputs that have high levels of rotational symmetry.

Finally, since the algorithm is highly geometric in nature, it would be challenging to adapt to incorporate non-geometric primitives such as colour (Roy et al., 2004).

2 RELATED WORK

Most of the work on mesh alignment focuses on and enhances the *Iterative Closest Point* (ICP) algorithm, which takes a set of points common to both input meshes and iteratively rotates the mesh until the common points are aligned as closely as possible. A variety of modifications and enhancements to the original ICP algorithm have been developed and studied; to make it geometrically stable (Gelfand and Rusinkiewicz, 2003), and to make it work with approximate nearest neighbouring points or with added noise (Maier-Hein et al., 2010). Other ICP based methods require an explicitly defined initial guess, which prevents the method being used in a completely automated manner (Besl and McKay, 1992).

Due to the variants of the ICP algorithms requiring subset inputs, and possibly some manual intervention, they are not directly comparable to the algorithm presented here. In reality, many practitioners use PCA directly on the vertices of the input meshes in order to provide an alignment that is good enough to work with. PCA is a robust, statistical method that is used extensively for non-geometric problems requiring dimensionality reduction. It is also efficient, since it is essentially a quadratic optimisation problem based on variance maximisation. A technique that is similar in spirit, called Independent Component Analysis (ICA) (Hyvärinen et al., 2001), is based on quartic optimisation and has also been used for 3D object recognition (Sahambi and Khorasani, 2003).

Despite its popularity, PCA has been reported to perform poorly when aligning meshes for 3D model recognition and this has been cited as motivation for developing rotationally invariant mesh descriptors (Kazhdan et al., 2003). Nevertheless, several important shape descriptors, such as (Cantoni et al., 2013), shape histograms (Ankerst et al., 1999) and descriptors based on higher order moments (Elad et al., 2002), are not rotationally invariant and thus require alignment. Extensions to PCA to overcome its shortcomings include PCA performed on the normals of a surface (Papadakis et al., 2007), and a continuous version of PCA applied to whole mesh triangles rather than just their vertices (Vranic et al., 2001). The latter is independent from distribution of vertices within the mesh and thus, overcomes some of the limitations of PCA the same way our method does. However, it requires a triangle mesh as input and has no obvious extensions to point clouds.

Finally, we note that PCA on mesh vertices is equivalent to least square fitting of a plane, and thus, the method of choice for the local tangent plane estimation required in applications ranging from surface reconstruction (Hoppe et al., 1992) to feature detection (Pauly et al., 2003).

In our implementation, we used the Point Cloud Library (Rusu and Cousins, 2011) for PCA and MeshLab (Cignoni et al., 2008) for the various geometric operations we performed as part of our testing; smoothing, simplification, adding noise, and remeshing.

3 ALIGNMENT ALGORITHM

We begin with two meshes (A and B), assuming that mesh B has been obtained from mesh A after a rotation by an unknown angle around an unknown axis and that some mesh processing operation may have been applied to B. Each mesh is centred on the origin as a preprocessing step. The translation can be stored and the reverse operation applied at the end of the procedure. The basic alignment algorithm first creates a regular grid around each mesh, then computes the subset of the grid nodes that are near to the mesh, and finally applies PCA to this subset of nodes.

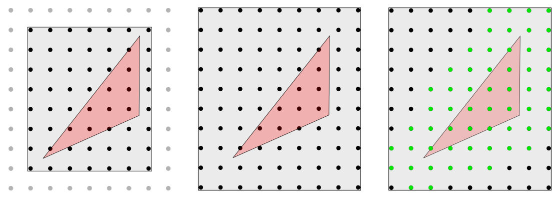


Figure 1: *Left:* the black nodes are the smallest subgrid that completely contains the red face. *Centre:* the smallest subgrid is extended to decrease discontinuities. *Right:* the nodes highlighted in green are the imprint of the red face on the lattice.

3.1 The Basic Algorithm

For each mesh M, we first create a regular 3D lattice, L_M, around the mesh M. The dimension of the grid is given by the user and trades-off the speed of the algorithm against the accuracy of the alignment. We then cycle through all the faces in M, and perform the following for each face $f \in M$:

- Calculate the smallest rectangular subgrid, P_f in L_M that completely contains f in all dimensions.
- To avoid discontinuities, P_f is expanded by one node in each direction along each axis, so a $2 \times 2 \times 3$ subgrid becomes $4 \times 4 \times 5$.
- For each lattice node, $n \in P_f$, determine the shortest distance from n to f.
- If the distance from n to f is less than 2 times the edge of a grid cell, export the node to list I_M (the 'imprint' of the mesh M on the lattice).

For each mesh imprint (from Figure 1; the collection of green nodes from every face in the mesh), we perform PCA on the nodes' coordinates and sort the output eigenvectors in decreasing order of eigenvalue magnitude. Note that a more sophisticated implementation of the algorithm would apply a weighted PCA, with the weight of each node derived from its distances to the mesh triangles that pushed it in I_M. However, we have found experimentally that would not have a significant effect on the results and thus, we opted for the much simpler unweighted PCA.

Between the two eigenbases (one for each mesh) formed by the principal components, pairs were formed based on them having the largest, middle, or smallest eigenvalue magnitude (blue, green, and red correspondingly in Figure 2). Since PCA does not provide oriented principal components, we have to ensure that the two

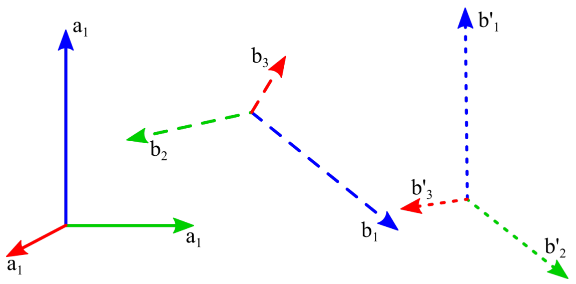


Figure 2: *Left, solid:* eigenbasis of mesh A. *Centre, dashed:* eigenbasis of mesh B. *Right, dotted:* eigenbasis of mesh B'.

eigenbases are consistently oriented. Were an inconsistent orientation was detected, the sign of the eigenvector with the smallest eigenvalue was flipped in one of the meshes.

For the actual mesh alignment, we start with the first pair of principal components, a_1, b_1, (those with the largest eigenvalues; blue). The rotation aligning a_1 with b_1 (blue) is computed and applied to mesh B to produce mesh B'. Lattice imprinting and PCA is then performed on B'. The rotation around a_1 (or, equivalently at this point, b_1) aligning a_2 with b'_2 (green) is then computed and applied to B' to produce a mesh B'' which is in alignment with A.

Note that it would have been possible to work out both rotations (or even, a single rotation) from the initial PCA, however this is likely to be less accurate, as the imprint could be different after the first rotation. Re-running the analysis ensures that PCA is run on a dataset that is closer to the original.

3.2 Iterative Algorithm

The basic algorithm can be repeated on mesh A and the mesh B'' (which has been aligned with A). The procedure usually leads to a closer alignment, but the decrease is not monotonic, and in some cases it could even lead to poorer alignment.

We believe that the reason for the non-monotonic decrease is the discrete nature of the grid relative to the mesh itself, as relatively small rotations will result in more significant changes to the distribution of grid nodes marked for processing.

3.3 Eigenvector Orientation

In order for the method to work, the principal components of meshes A and B must be consistently oriented. However, PCA does not define a

Table 1: Mean errors (in degrees) when recovering angles from a set of known rotations.

Mesh	Mesh Imprint	Vertex PCA
Bunny	0.45407	0.00648
Armadillo	0.18480	0.01150
Fandisk	1.11274	0.00075
Blade	0.93554	0.00650
Statuette	5.62708	0.01968

consistent orientation. In order to align the two principal components a_i and b_i consistently we evaluated a sum of distances function on the two extreme mesh vertex projections on the principal axis. For each of these points, the distance to every data point was summed. Then the principal axis was oriented in the direction of the point with the largest associated sum.

3.4 Input Types

The algorithm can, with minimal alteration, be used to align a mesh with a point cloud, or even to align two point clouds. Each point in the cloud is simply interpreted as a face with zero area. Here, the benefit of using the lattice instead of performing PCA directly on the point cloud is the increased robustness of the calculations due to the external reference. Which allows, for example, the alignment of a point cloud and a simplified version thereof.

When working with a point cloud, the same procedure would be followed, but treating each point as a face with no area, so the initial containing subgrid (P_f above) would always be a single cube, which would become a $3 \times 3 \times 3$ subgrid after expansion.

4 RESULTS

In the first experiment, each mesh had its principal components computed by mesh imprinting and was rotated by a known angle around the largest principle component. The proposed algorithm was then used to recover the angle by which the mesh had been rotated. This was then repeated using vertex PCA and the results were compared. The results are summarised in Table 1.

Since mesh alignment is an ill-posed problem, in a second experiment we evaluated the visual relevance of the reported errors, by rotating the test meshes by an unknown angle around an unknown axis. The algorithm was used to bring them back into alignment. The results are shown in Figures 3 - 6.

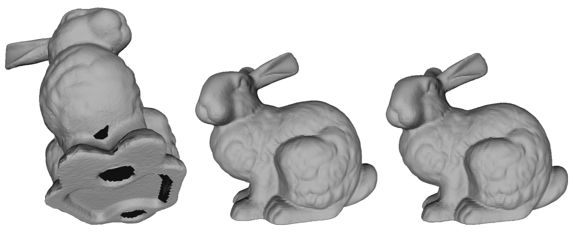


Figure 3: Standard Bunny results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

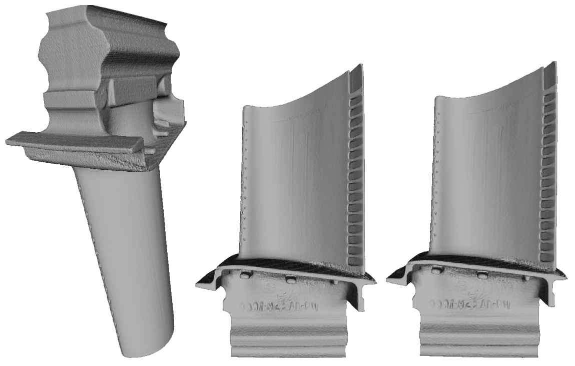


Figure 4: Standard Blade results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

4.1 Robustness against mesh processing operations

In the first experiment, three iterations of Laplacian smoothing were applied to each model and the process of random rotation repeated. The results are shown in Figures 7 - 10.

In the second experiment, the test meshes were simplified using clustering decimation with a cell size of 1% of the diagonal of the bounding box. The decimation results are shown in Table

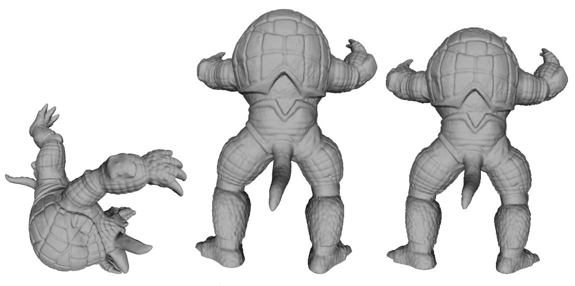


Figure 5: Standard Armadillo results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

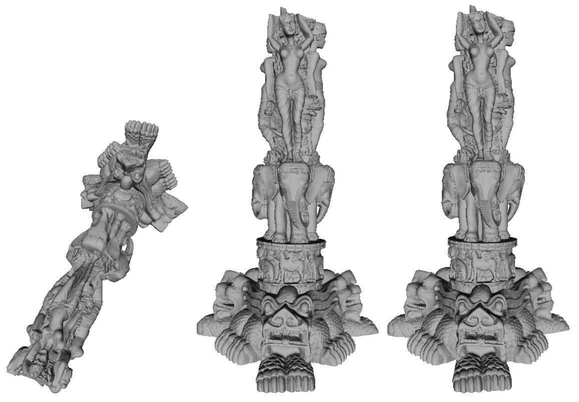


Figure 6: Standard Statuette results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

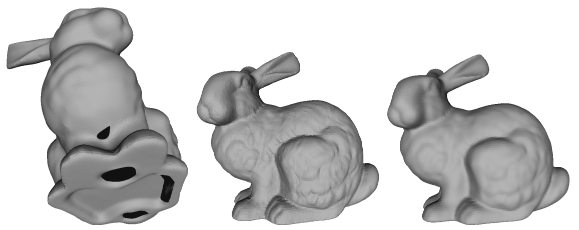


Figure 7: Smoothed and rotated Bunny results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

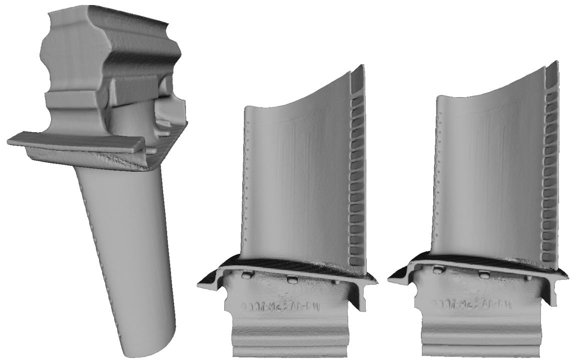


Figure 8: Smoothed and rotated Blade results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

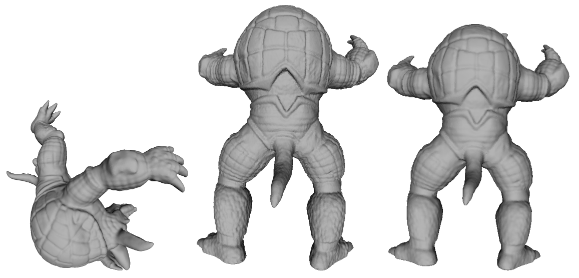


Figure 9: Smoothed and rotated Armadillo results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

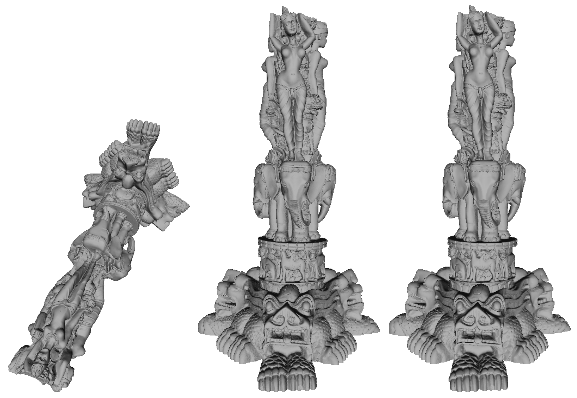


Figure 10: Smoothed and rotated Statuette results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

Table 2: Number of faces in the original and decimated meshes.

Mesh	Original faces	Decimated faces
Bunny	35,947	9,588
Armadillo	172,974	7,540
Blade	1,765,388	16,088
Statuette	10,000,000	18,330

2 and the alignment in Figures 11 - 14.

In the third experiment, the proposed method and vertex PCA were run on the Bunny, Armadillo, Fandisk, Blade and Statuette models and principal components were computed. The algorithms were then run against remeshed, simplified, noisy, and smoothed variants of those same meshes. For each method, the angular deviation between corresponding principal components of the original and the processed mesh were computed. The results are summarised in Tables 3 - 6.

For the noisy meshes, vertices were randomly displaced by 1% of the bounding diagonal. For the remeshed models, surfaces were reconstructed using the Poisson method (Kazhdan et al., 2006) with 10 octree subdivisions.

The proposed algorithm performed well on the

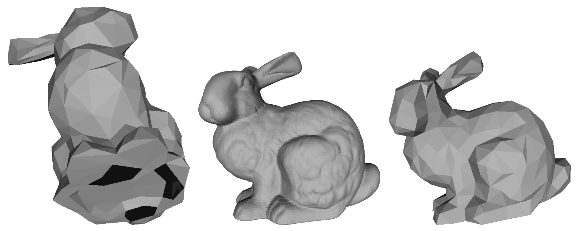


Figure 11: Simplified and rotated Bunny results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

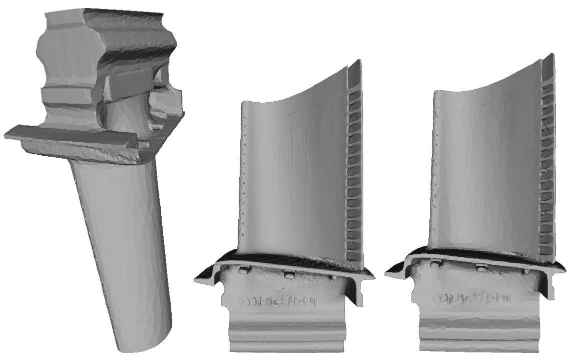


Figure 12: Simplified and rotated Blade results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

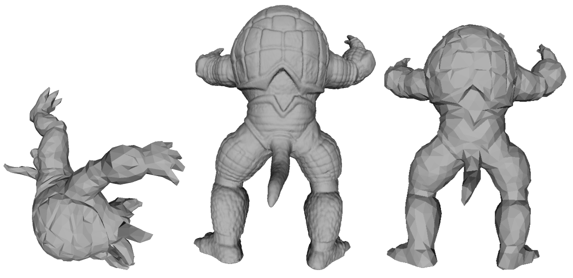


Figure 13: Simplified and rotated Armadillo results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

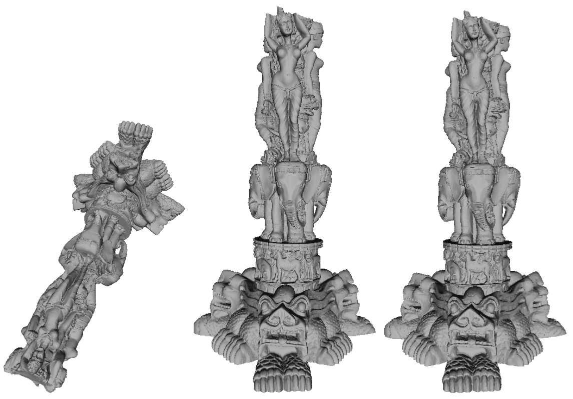


Figure 14: Simplified and rotated Statuette results. Left: initial rotation, middle: original, unrotated mesh, right: four iterations.

Table 3: Average deviation of principal components (in radians) when models were remeshed.

Remeshed	Mesh Imprint	Vertex PCA
Bunny	0.020450	0.044197
Fandisk	0.008237	0.008643
Armadillo	0.003517	0.014637
Blade	0.001020	0.018607
Statuette	0.063070	0.264650

Table 4: Average deviation of principal components (in radians) when models were simplified.

Simplified	Mesh Imprint	Vertex PCA
Bunny	0.008457	0.031510
Fandisk	0.024567	0.116733
Armadillo	0.002427	0.007957
Blade	0.005060	0.008093
Statuette	0.017803	0.300917

Table 5: Average deviation of principal components (in radians) when mesh vertices had noise added.

Noisy	Mesh Imprint	Vertex PCA
Bunny	0.001363	0.000333
Fandisk	0.000047	0.000240
Armadillo	0.000563	0.000757
Blade	0.000753	0.000357
Statuette	0.004997	0.000267

remeshed and simplified variants, but the noisy and smoothed variants were better served by vertex PCA. This is in line with our expectations as remeshing and simplification are likely to have a larger impact on vertex distribution than uniformly applied noise and smoothing.

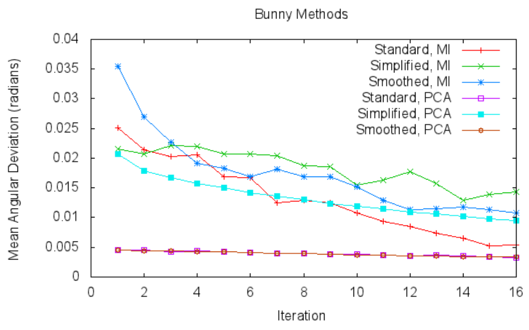
4.2 Iterative Algorithm

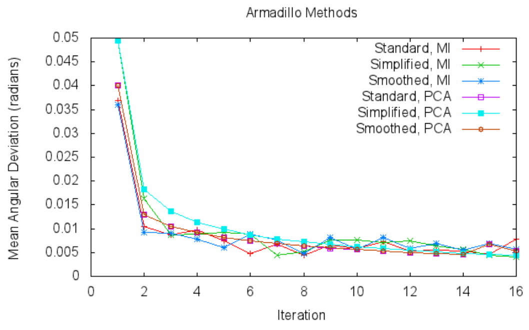
The performance of the iterative algorithm is shown in Figures 15 and 16. We notice that generally, the iterative algorithm has improved accuracy in each successive iteration and that most of the improvement materialises in the first three or four iterations.

Of particular note are the simplified Fandisk results in Figure 15. Whilst Vertex PCA appears to instantly converge to a highly accurate result, in reality this was a very poor alignment - the simplified Fandisk had many large, thin triangles. This significantly altered the distribution of vertices in the mesh and makes the comparison of the two sets of principal components invalid. The Mesh Imprint results on the simplified mesh are a true representation of the alignment, as are the Vertex PCA results for the standard and smoothed Fandisk.

Table 6: Average deviation of principal components (in radians) when meshes were simplified.

Smoothed	Mesh Imprint	Vertex PCA
Bunny	0.012873	0.000170
Fandisk	0.006650	0.000080
Armadillo	0.003757	0.000800
Blade	0.000270	0.000107
Statuette	0.001483	0.000000





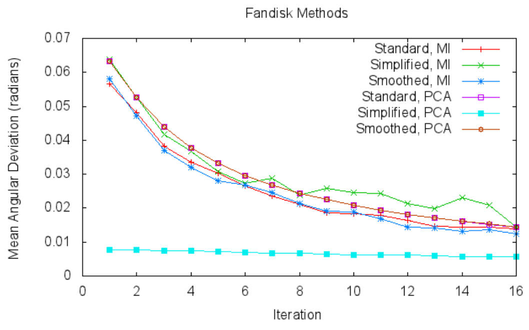


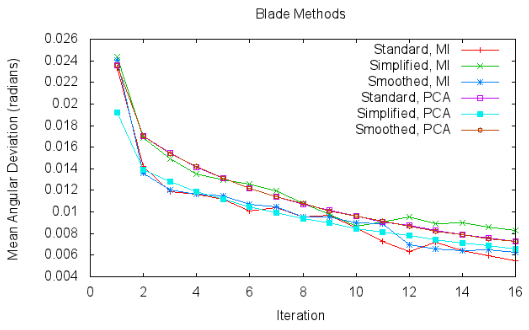
Figure 15: Mean angular deviation plotted against number of iterations for the Bunny, Armadillo and Fandisk.

4.3 CAD Meshes

The proposed method is particularly well suited for CAD meshes that have undergone mesh processing operations.

The Room 215 model shown in Figure 17 is a hand-made replica of an office created using CAD software, it has 171,711 faces and significant differences in vertex density. For instance, large areas of walls are represented by huge triangles, but tiny triangles are used to pick out the detail and high-curvature of the radiator grills and chairs. The simplified form has 16,080 faces.

When the principal components of each were computed and compared, Mesh Imprinting proved very effective, with a maximum deviation of 0.0086 radians across all principal components,



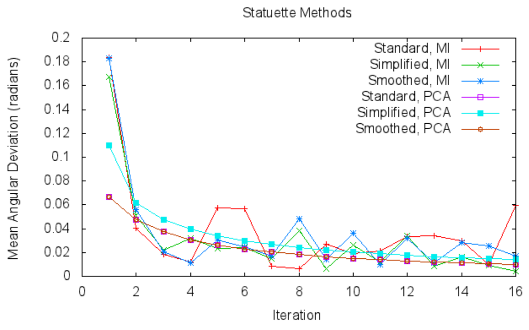


Figure 16: Mean angular deviation plotted against number of iterations for the Blade and Statuette.

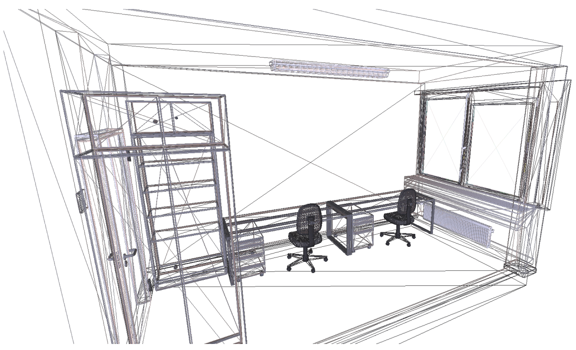


Figure 17: Wireframe view of the Room 215 model. Areas of high curvature have more triangles and appear as solid colours.

compared to a minimum deviation of 0.11 radians for vertex PCA.

The same test was run against a simple house model and a remeshed form thereof shown in Figure 18. The original mesh had 1,396 faces, the remeshed model had 98,818. The maximum deviation between the standard house and the remeshed form thereof was 0.075 radians. Vertex PCA however, had a minimum deviation of 0.12.

The model/dressed models in Figure 19 are a pair of models that both depict a human figure. This figure is nude in the regular model, but has long hair and is wearing bulky/baggy clothes in

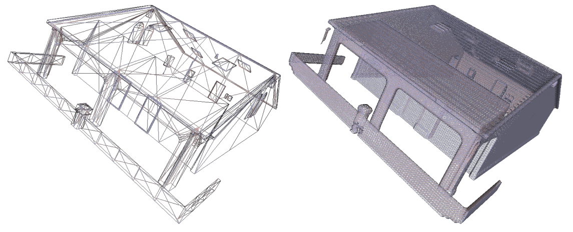


Figure 18: Wireframe view of the original House model and its (much higher vertex density) remeshed form.

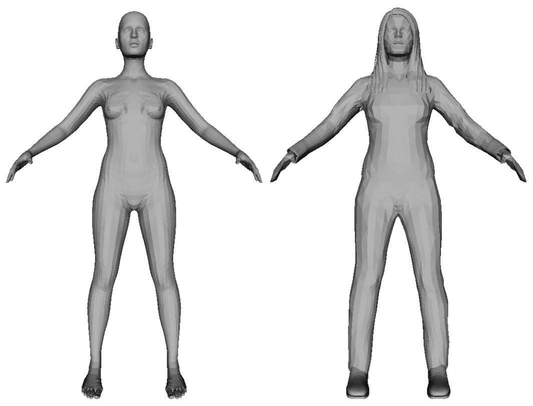


Figure 19: Model/dressed model.

the second.

Both were analysed and had their principal components compared. The computation was reasonably stable for both models, showing only small deviations between the two meshes, but with smaller deviations being produced by the mesh imprint. The maximum deviation between principal components computed by imprinting was 0.0022 radians, and the minimum computed by vertex PCA was 0.012.

This is not too surprising, as the two figures have the same pose and the changes from the regular model to the dressed model are relatively rotationally symmetric between the two minor principal components. The largest principal component is along the height of the model, and the proportions do not change sufficiently to make much difference to this.

Table 7: Mean angular deviations (in radians) between the simplified and standard Room215, standard and remeshed House, and standard and dressed Model.

Mesh	Mesh Imprint	Vertex PCA
Room 215	0.00647	0.11853
House	0.06064	0.20983
Model	0.00331	0.01230

Table 8: Initial scale of the meshes.

Mesh	Volume
Bunny	$78 \times 77 \times 60$
Armadillo	$127 \times 151 \times 115$
Fandisk	$121 \times 131 \times 67$
Blade	$352 \times 598 \times 274$
Statuette	$235 \times 396 \times 203$
Sphere	$105 \times 108 \times 105$
Vase	$55 \times 101 \times 55$

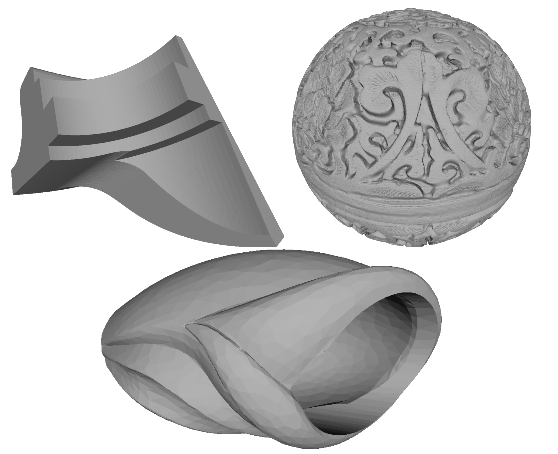


Figure 20: Top left: Fandisk, top right: Sphere, bottom: Vase.

4.4 Effect of Resolution

We experimented on the meshes at 10%, 50%, 100%, and 150% of their original scale. The initial size (100%) of each mesh is shown in table 8. In addition to the meshes used earlier, the Sphere and Vase models were also used, and can be seen in Figure 20.

Predictably, lower-resolution analyses usually produced alignments that were not so accurate as higher-resolution analyses. However for the Bunny and Armadillo models the 150% resolution alignments were actually slightly less accurate than the 100% resolution analyses.

The Vase, Sphere and Statuette all highlighted a limitation of the eigenvector orientation method; they produced inaccurate results as one principal component was incorrectly aligned. This occurs when the input meshes have high levels of rotational symmetry. When run at an appropriate resolution the algorithm is able to orient correctly the principal components, leading to a successful alignment. However, there appears to be no universally optimal resolution for the lattice in this regard.

5 DISCUSSION

The implementation of each algorithm was not optimised due to the wide variety of different techniques and circumstances under which each is possible and appropriate. For instance, an implementation could be multithreaded, or run on the GPU, or increase its memory footprint in order to save on disk reads/writes. Our implementations took the simple approach of reading the full file from the hard drive, processing the data entirely in memory, and writing the output back to the hard drive in a single execution threaded.

Processing large meshes can require large amounts of memory due to the sheer number of points that must be processed. Our method (by virtue of performing PCA on fewer points) will naturally have a smaller memory footprint. This would depend on the density of the mesh relative to the density of the grid, though the there are likely to be fewer circumstances when two very sparse meshes must be aligned at high resolution than the converse. In such situations however, the sparseness increases the likelihood of small differences in vertex distribution causing problems for Vertex PCA. Since our method is underpinned by PCA, this provides a lower bound for the computational complexity.

In our experiments, the method used to consistently orient the principal components was not always successful. If the method fails then it prompts a large (approximately π radians) rotation to bring the principal component into what it interprets as a correct alignment. This could be solved in a practical scenario by adding metadata to the mesh.

We note that the ICP algorithm is not directly comparable to the proposed algorithm, as it requires that one of the inputs be a subset of the other, that is, that they share points.

Our results show that presented algorithm works well for those inputs that have low rotational symmetry. Since the alignment is performed on a relatively coarse regular grid, the expected loss of accuracy compared to direct vertex PCA is noticeable, but the error is inside a range that would be considered tolerable in most applications.

Mesh Imprinting shows its strengths when original inputs are poorly meshed, for instance, if they have many long, thin triangles, or an uneven distribution thereof. Note that while long thin triangles are very rare in meshes that are acquired through physical optical devices such as laser scanners, they often dominate meshes produced by CAD software. In such cases, simplification and remeshing significantly affect the distribution of the vertices, causing Vertex PCA to produce highly inaccurate alignments.

In the future we plan a systematic analysis of the error of the standard PCA caused by vertex quantisation. Indeed, the small alignment error produced by our method is essentially a vertex coordinate quantisation error, which anyway may be present in the vertex coordinates, if for example the mesh had undergone lossy compression. By showing, as we conjecture, that the alignment error of our method and the vertex PCA error caused by vertex coordinate quantisation are comparable, we will further justify our approach.

6 ACKNOWLEDGEMENTS

The Room215 model is from the Max Planck Institute. The House mesh was created by user “pabong” and downloaded from www.tf3dm.com. The model and dressed model were created using the MakeHuman software from www.makehuman.org. The Vase, Sphere, Statuette, and Blade models were taken from the AIM@SHAPE repository.

REFERENCES

Ankerst, M., Kastenmüller, G., Kriegel, H.-P., and Seidl, T. (1999). 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases*, pages 207–226. Springer.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256.

Cantoni, V., Gaggia, A., and Lombardi, L. (2013). Extended gaussian image. In *Encyclopedia of Systems Biology*, pages 724–725. Springer.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). Meshlab: an open-source mesh processing tool. In *Sixth Eurographics Italian Chapter Conference*, pages 129–136.

Elad, M., Tal, A., and Ar, S. (2002). Content based retrieval of vrml objects: an iterative and interactive approach. In *Proceedings of* *the sixth Eurographics workshop on Multimedia 2001*, pages 107–118. Springer-Verlag.

Gelfand, N. and Rusinkiewicz, S. (2003). Geometrically stable sampling for the icp algorithm. In *Proc. International Conference on 3D Digital Imaging and Modeling*, pages 260–267.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. SIGGRAPH '92, pages 71–78.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. Wiley-Interscience.

Kaye, D. P. and Ivrissimtzis, I. (2011). Memory Efficient Surface Reconstruction Based on Self Organising Maps. In Grimstead, I. and Carr, H., editors, *Theory and Practice of Computer Graphics*, pages 25–32, Warwick, United Kingdom. Eurographics Association.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2003). Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*.

Maier-Hein, L., Santos, T. R. D., Franz, A. M., and Meinzer, H.-P. (2010). Iterative closest point algorithm in the presence of anisotropic noise.

Papadakis, P., Pratikakis, I., Perantonis, S., and Theoharis, T. (2007). Efficient 3d shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recogn.*, 40(9):2437–2452.

Pauly, M., Keiser, R., and Gross, M. (2003). Multi-scale feature extraction on point-sampled surfaces. In *Computer Graphics Forum*, volume 22, pages 281–289.

Roy, M., Foufou, S., and Truchetet, F. (2004). Mesh comparison using attribute deviation metric. *Journal of Image and Graphics*, 4:1–14.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Sahambi, H. S. and Khorasani, K. (2003). A neural-network appearance-based 3-d object recognition using independent component analysis. *IEEE Trans. Neur. Netw.*, 14(1):138–149.

Sfikas, K., Theoharis, T., and Pratikakis, I. (2014). Pose normalization of 3d models via reflective symmetry on panoramic views. *The Visual Computer*, pages 1–14.

Vranic, D., Saupe, D., and Richter, J. (2001). Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. In *Workshop on Multimedia Signal Processing*.