MINA: Convex Mixed-Integer Programming for Non-Rigid Shape Alignment —Supplementary Material—

Florian Bernard Zeeshan Khan Suri Christian Theobalt

MPI Informatics Saarland Informatics Campus

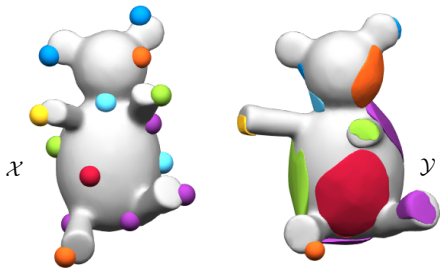


Figure 1. Illustration of control points of $\mathcal{X}$ (left) that are matched to convex polyhedra of $\mathcal{Y}$ (right). Colours indicate correspondences between control points and convex polyhedra.

A. Obtaining Convex Polyhedra on $\mathcal{Y}$

Given the j-th control point of $\mathcal{Y}_\mathcal{J}$, we obtain its associated convex polyhedron using a neighhourhood propagation strategy. To this end, we define a *planarity criterion* using the maximum of the *mean absolute deviation* (MAD) of the surface normals at the points in Z_j. For a given matrix $N \in \mathbb{R}^{n\times 3}$ and its column mean $\overline{N} \in \mathbb{R}^{1\times 3}$, the MAD is defined as $\mathrm{mad}(N) := \frac{1}{n}\sum_i |N_i - \overline{N}|$. As such, starting with $t{=}0$, we consider the vertices of the t-ring of the j-th vertex as Z_j, where we increase t as long as $\max(\mathrm{mad}(N_j^t)) \leq \eta$. Here, N_j^t denotes the matrix of the normals of the t-ring of the j-th vertex and η specifies a user-defined threshold. Once we have determined the largest t such that the t-ring neighborhood is sufficiently planar (below the threshold η), we discard all points in the rows of Z_j that are interior vertices of the convex polygon defined by Z_j (as they are redundant). In Fig. 1 (right) we show so-obtained convex polyhedra.

B. Piece-wise Linear Approximation of $\mathrm{SO}(3)$ Constraints

The constraint $R{\in}\,\mathrm{SO}(3)$ can be expressed as the orthogonality constraint $R^TR{=}\mathbf{I}_3$ in combination with $R_1{\times}R_2{=}R_3$. Hence, the constraint $R{\in}\,\mathrm{SO}(3)$ comprises exactly 6 quadratic equality constraints, which form a non-convex set. In order to define a piece-wise linear approximation we use *specially-ordered set of type 2* (sos2) variables. An sos2 variable is a non-negative vector where at most two consecutive element can be non-zero. With that, such a variable allows to encode a non-convex quadratic function in terms of a piece-wise linear one, so that in the end all quadratic constraints become linear, and the sos2 constraints are imposed based on (few) binary variables.

For illustrative purposes, we will now provide a simple example for a piece-wise linear approximation of a quadratic function. Let us consider the function $h(x) = x^2$ on the interval $[-1, 1]$. First, we split the domain into b bins, so that we evaluate x^2 at these b discrete positions, and then compute all values that fall in-between the sampled points as linear approximation between its two neighbour sample points. Let $b = 4$, and let $\phi = [-1, -0.5, 0, 0.5, 1]^T$ be a vector that contains the discretised domain, so that $\phi^2 = [1, 0.25, 0, 0.25, 1]^T$ defines the elementwise square of ϕ. Moreover, let $\lambda \in \mathbb{R}^{b+1}$ be a non-negative sos2 variable that sums to one (as mentioned, sos2 means that only two consecutive elements can be non-zero). Then, we can approximate

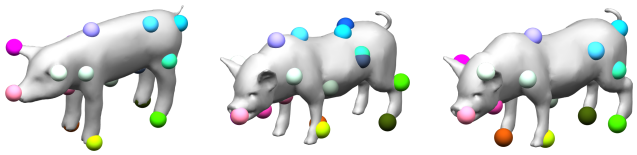
$$h(x) \approx \lambda^T\phi^2 \quad \text{for} \quad x = \lambda^T\phi\,. \tag{1}$$

For example, for $x = 0.75$, we obtain the sos2 variable $\lambda = [0, 0, 0, 0.5, 0.5]^T$ (since $x = 0.75 = \lambda^T\phi$). With that, we obtain $h(0.75) = 0.5625 \approx 0.625 = \lambda^T\phi^2$. The important property is that (1) allows to approximate the quadratic function $h(\cdot)$ based on a representation that is *linear* in the variables x and λ. In addition to [4] and [9], we refer the interested reader to [2, Ch. 9.1.11][1], where sos2 constraints as well as the idea of using a logarithmic Gray encoding are explained.

C. Search Space Reduction

In addition to using a logarithmic encoding of the $\mathrm{SO}(3)$ discretisation variables, we also impose further constraints

[1] also available online at https://docs.mosek.com/modeling-cookbook/mio.html#continuous-piecewise-linear-functions



$\mathcal{X}$ $\mathcal{Y}$ (ADG) $\mathcal{Y}$ (ours)

Figure 2. Shape $\mathcal{X}$ (left) is matched to $\mathcal{Y}$, where a search space reduction using ADG [6] leads to wrong matchings (middle), whereas ours produces correct correspondences (right).

upon the matching matrix P, so that the size of the overall search space can be reduced. A similar idea has also been pursued in [6], where a scalar criterion based on the average geodesic distance (ADG) was used. In contrast, rather than using a single scalar value for each vertex, we propose to leverage a more powerful approach that considers more descriptive statistics of geodesic distances, see Fig. 2. To this end, for each control point we compute n_{prctile} evenly spaced percentiles from 0 to 100% of the geodesic distance from this control point to all other points. Let $\gamma^{\mathcal{X}} \in \mathbb{R}^{u \times n_{\text{prctile}}}$ and $\gamma^{\mathcal{Y}} \in \mathbb{R}^{v \times n_{\text{prctile}}}$ denote the so-obtained percentile matrices, where the columns are the ordered percentiles from 0 to 100%. As such, the matrices $\gamma^{\mathcal{X}}$ and $\gamma^{\mathcal{Y}}$ can be seen as features of the respective shapes extracted at the control points. Whenever two control points $i \in [u], j \in [v]$ correspond to each other, the features $\gamma_i^{\mathcal{X}}$ and $\gamma_j^{\mathcal{Y}}$ should be similar, so that $d_{ij} := \|\gamma_i^{\mathcal{X}} - \gamma_j^{\mathcal{Y}}\|$ is small. Based on this observation, we use the feature distances $[d_{ij}]_{i,j}$ and sequentially solve n_{LAP} linear assignment problems (LAP) [7] to match features. The idea of solving a sequence of LAPs is to not only find the single best matching P_1 of features, but rather finding multiple solutions $P_1, \dots, P_{n_{\text{LAP}}}$, so that the nonzero elements in $P_{\text{all}} = \sum_{\ell=0}^{n_{\text{LAP}}} P_\ell$ define the allowed matchings in P. Here, the matrix P_ℓ is obtained by performing a feature matching using $[d_{ij}]_{i,j}$ when forbidding all previous matchings $P_1, \dots, P_{\ell-1}$. As such, when optimising MINA, we constrain all elements of P to be zero for those elements where P_{all} is zero. Using this procedure is advantageous over simple thresholding of $[d_{ij}]_{i,j}$, since on the one hand feasibility is guaranteed, and on the other hand the number of allowed matchings is equal for all control points.

D. Further Implementation Details

We have implemented MINA in the optimisation modelling toolbox Yalmip [5], which uses the conic mixed-integer branch and bound solver MOSEK [1] as backend (with default parameters). In all experiments we used $\lambda_{\text{c}}=4$, $\lambda_{\text{r}}=1$ and $\lambda_{\text{s}}=0.5$, where we account for different problem sizes by multiplying each $\lambda_\bullet$ with $\frac{1}{\sqrt{\#}}$, where $\#$ denotes the total number of elements that the norm is applied to. We set the weights ω_e for the smoothness term to $\omega_e = \frac{d_e}{\sum_{e \in \mathcal{E}} d_e}$, where for $e = (p, q)$ by d_e we denote the length of the common edge of triangles p, q. With that, we achieve that the deformation of two adjacent triangles p, q is more flexible when their common edge is small. We set the planarity threshold to $\eta = \frac{1}{2}$. For keeping the number of variables small, for each convex polyhedron Z_j we only keep the respective control point as well as four additional points obtained via farthest point sampling (FPS) using geodesic distances as metric. Note that this results in convex polyhedra that are either a single point (if none of the t-rings of the j-th control point satisfies the planarity criterion), or Z_j is a 5×3 matrix. Since the non-rigid deformation induced by a sparse set of matched control points is relatively coarse, rather than modelling τ with the original mesh resolution we use downsampled meshes with about 300 faces, similarly as in [8]. We set $n_{\text{LAP}}=5$, $n_{\text{prctile}} = \min(n_{\mathcal{X}}, n_{\mathcal{Y}})$, $M=0.2$ and use $b=4$ bins for the $\text{SO}(3)$ discretisation.

Next, we provide additional details on shape to point cloud matching and the relation between partial shape matching and outlier rejection.

Shape to point cloud matching. The main difference when $\mathcal{Y}$ is represented as a point cloud rather than a mesh is that we need to use a different approach for computing geodesic distances and normals (required for sampling control points, for the definition of the convex polyhedra as described in Sec. A, and for the search space reduction described in Sec. C). In our case we compute geodesic distances and normals based on a nearest neighbour graph, where we use the 3 nearest neighbours. After this information is obtained, the overall optimisation problem is equivalent to the one when $\mathcal{Y}$ is a mesh, since the only information of $\mathcal{Y}$ that is explicitly used in our optimisation problem formulation are the convex polyhedra.

Relation between partial shape matching and outlier rejection. In our considered *partial shape matching* setting we match *all* control points of the partial shape $\mathcal{X}$ to the full shape $\mathcal{Y}$. This is in contrast to our *outlier rejection* setting, where we allow that some control points of $\mathcal{X}$ are not matched to $\mathcal{Y}$. However, although for partial shape matching we do not use outlier rejection, we mention that principally it could be used for matching a full shape to a partial one.

E. Additional TOSCA Results

In Fig. 3 we report runtime statistics over all 71 shape matching instances from the TOSCA datasets for all considered methods. On this dataset, the median processing time of our method is $\approx$15min, whereas the other methods require less than one minute.

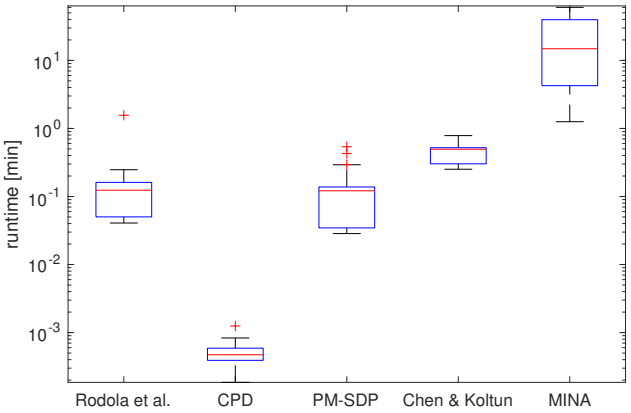


Figure 3. Runtime statistics for the TOSCA dataset. Note that the vertical axis is shown in log-scale.

In Fig. 4 we present further results where also the deformed shape $\tau(\mathcal{X})$ is shown.

References

[1] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. 2

[2] MOSEK ApS. *MOSEK Modeling Cookbook. Release 3.2.1*, 2020. 1

[3] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008. 4

[4] Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research*, May 2017. 1

[5] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *ICRA*, 2004. 2

[6] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):73, 2016. 2

[7] James Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, Mar. 1957. 2

[8] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics (TOG)*, 26(3):80, 2007. 2

[9] Juan Pablo Vielma and George L Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1-2):49–72, 2011. 1