Software tool for cranial orthosis design

Alena Vasatova Milan Jaros Tomas Karasek Petr Strakos

VSB - Technical University of Ostrava
National Supercomputing Center IT4Innovations

alena.vasatova@vsb.cz

November 1, 2017

Introduction

- Treatment of skull deformities of children by cranial orthosis has been increasingly used since it was first documented in 1979.



Introduction

- Treatment of skull deformities of children by cranial orthosis has been increasingly used since it was first documented in 1979.
- Increasing number of the cranial deformities due to the recommended sleeping supine position (to reduce sudden death syndrome) and keeping infant too long in one position.



Introduction

- Treatment of skull deformities of children by cranial orthosis has been increasingly used since it was first documented in 1979.
- Increasing number of the cranial deformities due to the recommended sleeping supine position (to reduce sudden death syndrome) and keeping infant too long in one position.
- Need to be designed individually.



Introduction

- Currently fully manual task. Tool should make this process semi-automatic with only small intervention from user and as such speed up whole process.

Introduction

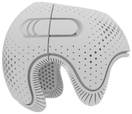
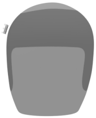
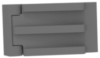
- Currently fully manual task. Tool should make this process semi-automatic with only small intervention from user and as such speed up whole process.
- In future it will be part of whole process from 3D scanning patient to 3D printing and it will allow product the orthosis anywhere, without expensive devices on place through web services.

Introduction

- Currently fully manual task. Tool should make this process semi-automatic with only small intervention from user and as such speed up whole process.
- In future it will be part of whole process from 3D scanning patient to 3D printing and it will allow product the orthosis anywhere, without expensive devices on place through web services.
- Contractual research HS7831610 conducted in collaboration with ING corporation spol. s.r.o..

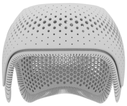
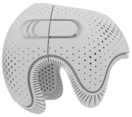
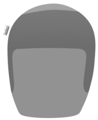
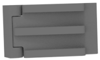
Cranial orthosis model

- The model itself is composed of two parts: a helmet and a locking mechanism. Each part is represented by high-resolution mesh.

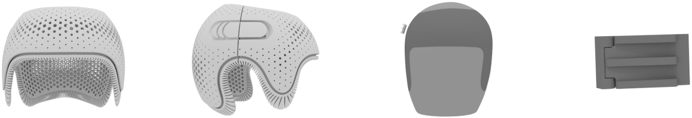
Cranial orthosis model

- The model itself is composed of two parts: a helmet and a locking mechanism. Each part is represented by high-resolution mesh.
- The locking mechanism can only translate and rotate to preserve its functionality.

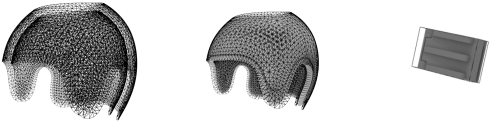





Cranial orthosis model

- The model itself is composed of two parts: a helmet and a locking mechanism. Each part is represented by high-resolution mesh.
- The locking mechanism can only translate and rotate to preserve its functionality.

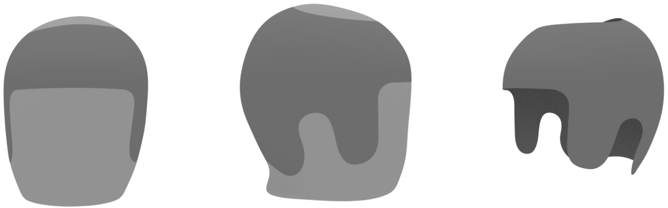


- Beside that, we also need auxiliary meshes, so-called cages.



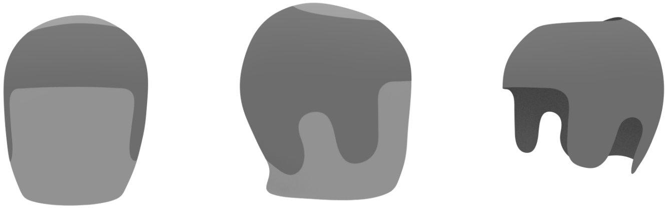
Cranial orthosis model

- The model is modified to individual patient based on 3D scan of the head. The scan of head is cropped afterwards by outlines specified by the medical technician.



Cranial orthosis model

- The model is modified to individual patient based on 3D scan of the head. The scan of head is cropped afterwards by outlines specified by the medical technician.
- The goal of the transformation is a non-rigid deformation of the orthosis body to fit the cropped scan and a rigid transformation of the locking mechanism to its specified position.



Morphing algorithm

1. Modification and enhancement of Blender for rapid testing of proposed methodology - MeshDeform modifier + Shrinkwrap modifier.

Morphing algorithm

1. Modification and enhancement of Blender for rapid testing of proposed methodology - MeshDeform modifier + Shrinkwrap modifier.
2. MeshDeform modifier was parallelized using MPI technology to improve its speed and to allow handling of large data sets.

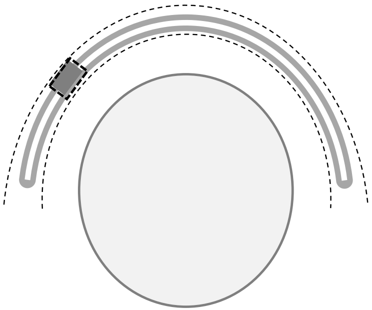
Morphing algorithm

1. Modification and enhancement of Blender for rapid testing of proposed methodology - MeshDeform modifier + Shrinkwrap modifier.
2. MeshDeform modifier was parallelized using MPI technology to improve its speed and to allow handling of large data sets.
3. MeshDeform modifier replaced by transformation using radial basis function (RBF), which is computationally less expensive.

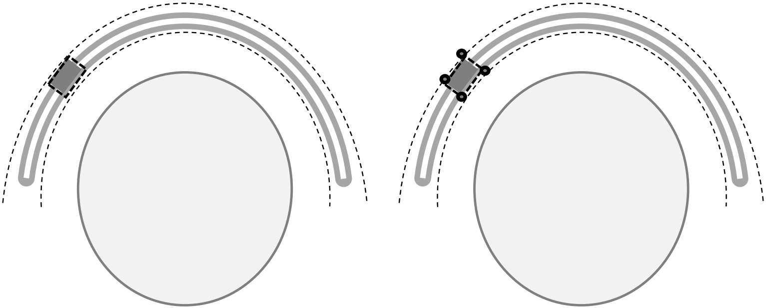
Morphing algorithm

1. Modification and enhancement of Blender for rapid testing of proposed methodology - MeshDeform modifier + Shrinkwrap modifier.
2. MeshDeform modifier was parallelized using MPI technology to improve its speed and to allow handling of large data sets.
3. MeshDeform modifier replaced by transformation using radial basis function (RBF), which is computationally less expensive.
4. Transformation by RBF also allows easy incorporation of the rigid parts, which MeshDeform modifier cannot implicitly do.

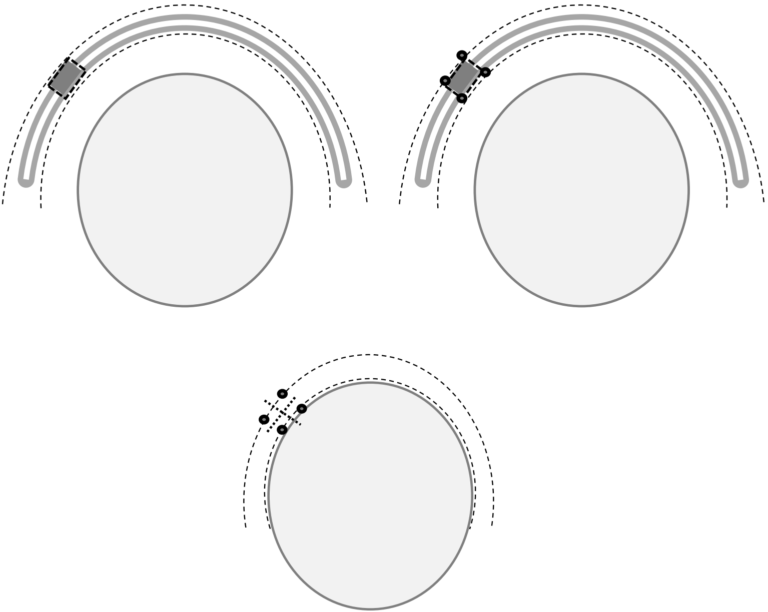
Morphing algorithm



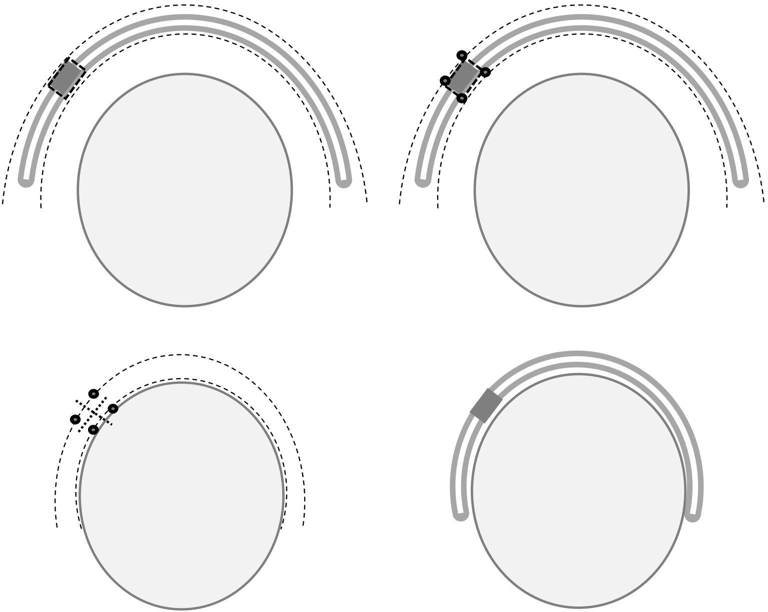
Morphing algorithm



Morphing algorithm



Morphing algorithm



Radial basis function

- Function $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ that
  - exactly interpolates the displacement $u_j \in \mathbb{R}^3$ of given control points $x_j \in \mathbb{R}^3$, $j = 1, ..., m$,
  - smoothly interpolates this displacement into the mesh.

Radial basis function

- Function $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ that
  - exactly interpolates the displacement $u_j \in \mathbb{R}^3$ of given control points $x_j \in \mathbb{R}^3$, $j = 1, \ldots, m$,
  - smoothly interpolates this displacement into the mesh.
- The displacement function is then represented as

$$\varphi_l(x) = \sum_{i=1}^{n_p} \beta_{l,i} p_i(x) + \sum_{j=1}^{m} \theta_{l,j} \rho_{x_j}(x),$$

where n_p is dimension of used polynomials, we choose linear, thus $n_p = 4$.

Radial basis function

- Function $\varphi : \mathbb{R}^3 \to \mathbb{R}^3$ that
  - exactly interpolates the displacement $u_j \in \mathbb{R}^3$ of given control points $x_j \in \mathbb{R}^3$, $j = 1, ..., m$,
  - smoothly interpolates this displacement into the mesh.
- The displacement function is then represented as

$$\varphi_l(x) = \sum_{i=1}^{n_p} \beta_{l,i} p_i(x) + \sum_{j=1}^{m} \theta_{l,j} \rho_{x_j}(x),$$

where n_p is dimension of used polynomials, we choose linear, thus $n_p = 4$.

Radial basis function

- The coefficients $\beta_l \in \mathbb{R}^{n_p}$ and $\theta_l \in \mathbb{R}^m$ are defined by

$$\begin{pmatrix} K & B^\top \\ B & 0 \end{pmatrix} \begin{pmatrix} \theta_l \\ \beta_l \end{pmatrix} = \begin{pmatrix} u_l \\ 0 \end{pmatrix},$$

where

$$K := (\rho(\|x_j - x_k\|_{\mathbb{R}^3}))_{j,k=1,\ldots,m} \in \mathbb{R}^{m \times m},$$

$$B := (p_i(x_k))_{\substack{i=1,\ldots,n_p \\ k=1,\ldots m}} \in \mathbb{R}^{n_p \times m}$$

and

$$u_l := (u_{1,l}, \ldots, u_{j,l}) \in \mathbb{R}^m.$$

Radial basis function

- The coefficients $\beta_l \in \mathbb{R}^{n_p}$ and $\theta_l \in \mathbb{R}^m$ are defined by

$$\begin{pmatrix} K & B^\top \\ B & 0 \end{pmatrix} \begin{pmatrix} \theta_l \\ \beta_l \end{pmatrix} = \begin{pmatrix} u_l \\ 0 \end{pmatrix},$$

  where

$$K := (\rho(\|x_j - x_k\|_{\mathbb{R}^3}))_{j,k=1,\dots,m} \in \mathbb{R}^{m\times m},$$

$$B := (p_i(x_k))_{\substack{i=1,\dots,n_p \\ k=1,\dots m}} \in \mathbb{R}^{n_p\times m}$$

  and

$$u_l := (u_{1,l}, \dots, u_{j,l}) \in \mathbb{R}^m.$$

- We use triharmonic or thin plate spline (TPS)

$$\rho(r) = r^3, \quad \rho(r) := \frac{\Gamma(3/2-q)}{2^{2q}\pi^{3/2}(q-1)!} r^{2q-3}.$$

Radial basis function with rigid transformation

- n rigid objects.

Radial basis function with rigid transformation

- n rigid objects.
- Predefined linear transformations $L_q \in \mathbb{R}^{3 \times n_p}$, $q = 1, .., n$.

Radial basis function with rigid transformation

- n rigid objects.
- Predefined linear transformations $L_q \in \mathbb{R}^{3\times n_p}$, $q = 1, .., n$.
- Distance between given point and closest point of the mesh using octree-based spatial algorithm, which can search the mesh to quickly locate the point on the mesh face:
  - $\mathcal{D}_0(x)$ represents the distance from a point x to the closest object,
  - $\mathcal{D}_q(x)$, $q = 1, ..., n$ to q-th object.

Radial basis function with rigid transformation

- n rigid objects.
- Predefined linear transformations $L_q \in \mathbb{R}^{3\times n_p}$, $q = 1, .., n$.
- Distance between given point and closest point of the mesh using octree-based spatial algorithm, which can search the mesh to quickly locate the point on the mesh face:
  - $\mathcal{D}_0(x)$ represents the distance from a point x to the closest object,
  - $\mathcal{D}_q(x),\ q = 1, ..., n$ to q-th object.
- These functions ensure that the non-linear part of the transformation and the linear transformations $L_q,\ q \neq r$ tends to zero as we move towards the r-th rigid object.

Radial basis function with rigid transformation

- The coefficients β_l for polynomial corrections are replaced by weighted sum of the individual object linear transformations

$$\mathcal{L}(x) = \sum_{q=1}^{n} w_q(x) L_q, \;\; w_q(x) = \frac{v_q(x)}{\sum_{r=1}^{n} v_r(x)}, \;\; v_q(x) = \frac{1}{\mathcal{D}_q(x)^{\mu}}$$

Radial basis function with rigid transformation

- The coefficients β_l for polynomial corrections are replaced by weighted sum of the individual object linear transformations

$$\mathcal{L}(x) = \sum_{q=1}^{n} w_q(x) L_q, \;\; w_q(x) = \frac{v_q(x)}{\sum_{r=1}^{n} v_r(x)}, \;\; v_q(x) = \frac{1}{\mathcal{D}_q(x)^{\mu}}$$

- The kernels are also weighted

$$\tilde{\rho}_{x_j}(x) = |\mathcal{D}_0(x)| \, |\mathcal{D}_0(x_j)| \, \rho_{x_j}(x).$$

Radial basis function with rigid transformation

- The coefficients β_l for polynomial corrections are replaced by weighted sum of the individual object linear transformations

$$\mathcal{L}(x) = \sum_{q=1}^{n} w_q(x) L_q, \;\; w_q(x) = \frac{v_q(x)}{\sum_{r=1}^{n} v_r(x)}, \;\; v_q(x) = \frac{1}{\mathcal{D}_q(x)^{\mu}}$$

- The kernels are also weighted

$$\tilde{\rho}_{x_j}(x) = |\mathcal{D}_0(x)| \, |\mathcal{D}_0(x_j)| \, \rho_{x_j}(x).$$

- Thus we get transformation function and rewritten equation system

$$\varphi_l(x) = \sum_{i=1}^{n_p} \mathcal{L}(x) p_i(x) + \sum_{j=1}^{m} \theta_{l,j} \tilde{\rho}_{x_j}(x),$$

$$K\theta_l + T = u_l, \;\; l = 1,2,3, \;\; T = \begin{pmatrix} p(x_1)^T \mathcal{L}(x_1)^T \\ p(x_2)^T \mathcal{L}(x_2)^T \\ \vdots \\ p(x_m)^T \mathcal{L}(x_m)^T \end{pmatrix}.$$

Radial basis function with rigid transformation

- The coefficients β_l for polynomial corrections are replaced by weighted sum of the individual object linear transformations

$$\mathcal{L}(x) = \sum_{q=1}^{n} w_q(x) L_q, \;\; w_q(x) = \frac{v_q(x)}{\sum_{r=1}^{n} v_r(x)}, \;\; v_q(x) = \frac{1}{\mathcal{D}_q(x)^{\mu}}$$

- The kernels are also weighted

$$\tilde{\rho}_{x_j}(x) = |\mathcal{D}_0(x)| \, |\mathcal{D}_0(x_j)| \, \rho_{x_j}(x).$$

- Thus we get transformation function and rewritten equation system

$$\varphi_l(x) = \sum_{i=1}^{n_p} \mathcal{L}(x) p_i(x) + \sum_{j=1}^{m} \theta_{l,j} \tilde{\rho}_{x_j}(x),$$

$$K\theta_l + T = u_l, \;\; l = 1, 2, 3, \;\; T = \begin{pmatrix} p(x_1)^T \mathcal{L}(x_1)^T \\ p(x_2)^T \mathcal{L}(x_2)^T \\ \vdots \\ p(x_m)^T \mathcal{L}(x_m)^T \end{pmatrix}.$$

Rigid transformation

- Linear transformation matrix L_q consists of 12 coefficients, and we need 4 points in space to determine them.

Rigid transformation

- Linear transformation matrix L_q consists of 12 coefficients, and we need 4 points in space to determine them.
- The easy way is to take these points from principal axes of the rigid cage, Principal axes are obtained through the principal component analysis (PCA).

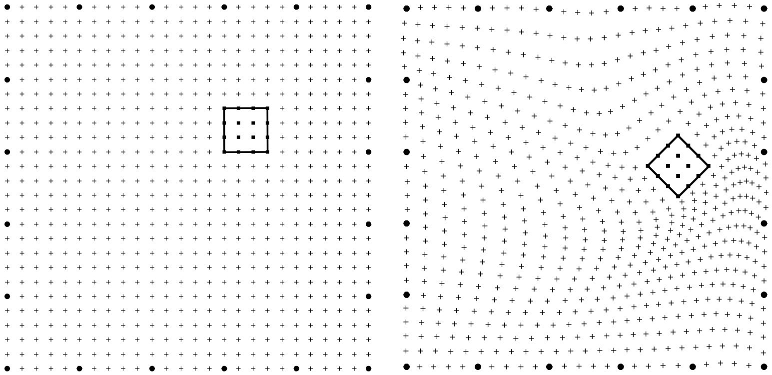
Rigid transformation

- Linear transformation matrix L_q consists of 12 coefficients, and we need 4 points in space to determine them.
- The easy way is to take these points from principal axes of the rigid cage, Principal axes are obtained through the principal component analysis (PCA).
- PCA is statistical method used to estimate the necessary information from the measured data.

Rigid transformation

- Linear transformation matrix L_q consists of 12 coefficients, and we need 4 points in space to determine them.
- The easy way is to take these points from principal axes of the rigid cage, Principal axes are obtained through the principal component analysis (PCA).
- PCA is statistical method used to estimate the necessary information from the measured data.
- We are able to determine the axes through the use of eigenvalues and eigenvectors of the covariance matrix consisting of a small group of neighboring points.

Rigid transformation



Implementation

- VTK library to work with 3D geometry and mesh models.

Implementation

- VTK library to work with 3D geometry and mesh models.
- Intel MKL library to solve large systems of linear equations.

Implementation

- VTK library to work with 3D geometry and mesh models.
- Intel MKL library to solve large systems of linear equations.
- OpenMP technology to parallelize transformation.

Results

- We have performed measurements focusing on algorithm speed and its possible speed-up by utilizing OpenMP framework on multiple cores.

Results

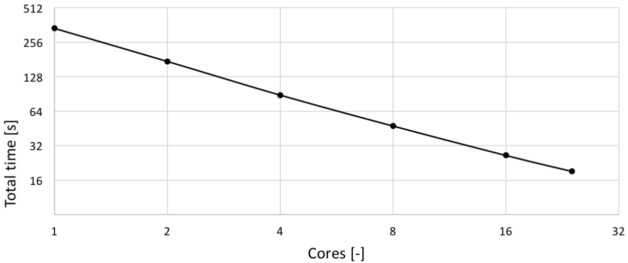
- We have performed measurements focusing on algorithm speed and its possible speed-up by utilizing OpenMP framework on multiple cores.
- We have also measured computational demands of the algorithm based on the model size.

Results

- We have performed measurements focusing on algorithm speed and its possible speed-up by utilizing OpenMP framework on multiple cores.
- We have also measured computational demands of the algorithm based on the model size.
- For all the tests, configuration of the RBF and the solver was as follows:
  - Thin plate spline (TPS) as a kernel function $\rho(r)$.
  - Bunch-Kaufman factorization of a symmetric matrix using packed storage has been used as a solver.

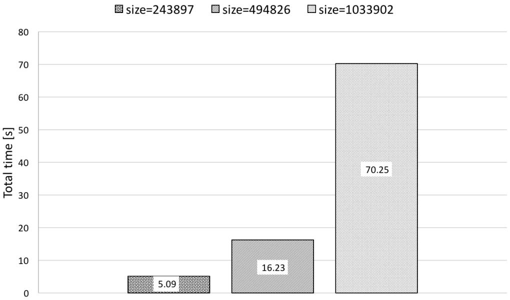
Algorithm speed and possible speed-up by OpenMP on multiple CPU cores

CPU cores [-]	1	2	4	8	16	24
Shrink [s]	0.80	0.75	0.73	0.72	0.73	0.72
Prepare [s]	85.36	46.44	23.32	12.30	7.96	5.63
Solve [s]	3.94	2.65	2.18	1.96	1.88	1.88
Transform [s]	248.07	124.19	62.29	32.56	15.75	10.87
Total [s]	338.16	174.03	88.51	47.54	26.33	19.09



Computation times for different sizes of the model

Model size [vertices] (1d)	243897	494826	1033902
Shrink [s] (2)	0.24	0.72	2.61
Prepare [s] (3)-(4)	1.65	4.83	18.54
Solve [s] (5)	0.44	1.89	10.19
Transform [s] (6)	2.76	8.79	38.91
Total [s] (2)-(6)	5.09	16.23	70.25



Result model