university of groningen / faculty of science and engineering / Bernoulli institute

Computer-Assisted Virtual Acetabular Fracture Reconstruction

Master's Thesis

August 2018

Student: L.E.N. Baakman (s1869140)

Primary supervisor: RNDr. J. Kosinka, Ph.D.

Secondary supervisor: dr. ir. P.M.A. van Ooijen

Tertiary supervisor: dr. M.J.H. Witjes

Contents

Preface

Before you lies my master's thesis, the basis of which is an application that attempts to assist with the solving of three-dimensional puzzles made up of the fragments of fractured bones. It has been written to fulfill the graduation requirements of the Computing Science master at Groningen University (RUG). I was engaged in researching and writing this thesis from October 2017 until August 2018.

This thesis has been supervised by supervisors both at the RUG and the University Medical Center Groningen (UMCG). The structure of this document is slightly unconventional since it is an hybrid of the standards used in computing science and in medicine. Part I presents the principal component of my research, the interested reader who is short on time can suffice with that part. The chapters in Part II expound on the material in the earlier part, and present work done as part of my thesis project that did not make the cut for the paper. Each of these chapters can be read independently of the others and independently of Part I. I strongly urge anyone disheartened by the results in Part I to read Chapter 4. This chapter seems to indicate that the proposed registration method has some promise, in spite of its disappointing performance on clinical data.

This project would not have been possible without my supervisors at both the RUG and the UMCG. It was always helpful to bat ideas around with the other (PhD) students at the 3D lab. I want to name Anne in particular for making her data available to me and processing it especially for me and for answering my never ending questions. This thesis would have been a lot worse without the proofreading by Jiří, Dirk, and my parents. Especially the first two have reviewed enormous amounts of texts. Without Julie I would be a lot less certain about the correctness of the more medical sections. Any mistakes left are solely mine. Without Jelle, Bastiaan and especially Dirk I would not have come through this project with my sanity fully intact. I owe the former two for providing me with caffeine when I was at my most stressed and for listening to me complain endlessly about Unity, and the latter for that and everything else. Now that I have (nearly) graduated I finally get to keep the promise I have repeatedly made to him for the past six years; after I have handed this in I won't be as stressed.

Laura Baakman

Part I

Paper

Computer-Assisted Virtual Acetabular Fracture Reconstruction

L.E.N. Baakman (s1869140)

August 19, 2018

Abstract

A new method to reduce acetabular fractures has been pioneered at the University Medical Center Groningen (UMCG). It requires manual virtual reduction of the fracture, which is time-consuming and difficult. We use Iterative closest point (ICP) for the computer-assisted reduction of acetabular fractures, together with our newly introduced error metric which punishes intersections between fragments. To allow the incorporation of the intersection term of this error metric, this research also proposes the use of iterative gradient descent (IGD) to find the transformation at each iteration of the ICP algorithm. The performance of the proposed method is compared with that of existing variants of the ICP algorithm on the clinical models of eight patients treated at the UMCG. We have found no improvement in performance due to the inclusion of an intersection term in the error metric. However, the results of the other implemented and tested registration methods suggest that factors other than the used error metric may have been at fault. Therefore, we conclude that further research into this error metric and its minimization is needed.

1 Introduction

In the past decades, incidence of pelvic fractures has increased due to raising rates of high-speed motor vehicle accidents and falls from heights. The rate of mortality of major pelvic fractures is 10% to 20% percent, open fractures have mortality rates as high a 50% [1]. Out of the 224 patients treated for an acetabular fracture at the Radboud University Medical Center in the Netherlands between 2004 and 2014, 15% needed a total hip arthroplasty (THA). Patients who have a THA often have a worse functional outcome than patients with a preserved hip joint [2]. One of the factors that significantly impacts the need for a total hip arthroplasty is the quality of the fracture reduction. From the data from different hospitals, between 6% and 26% of acetabular fracture reductions are not satisfactory [2–6].

Merema et al. [7] are pioneering a new procedure for acetabular fracture surgery at the University Medical Center Groningen (UMCG). By using computed tomography (CT) data and surgical planning software that uses 3D visualization, they create a virtual model of the fractured pelvis. Based on the virtual reduction of this fracture, patient-specific fixation devices and intra-operative drilling

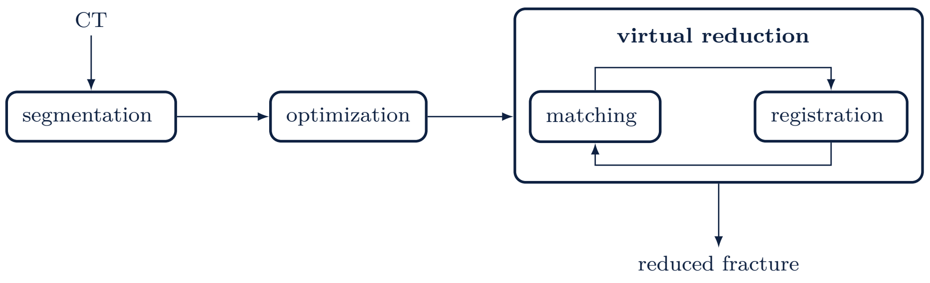


Figure 1: *Proposed pipeline for automated virtual reduction.*

guides are designed. Contrary to the current standard of care, this approach allows the use of personalized fixation devices that can be tailored to both the shape of the pelvis and the type of fracture. The virtual reduction of the fracture is also used to determine screw directions and sizes. Comparable approaches have been used at the UMCG for reconstruction of the mandible [8, 9] and fibula [10], and for secondary maxillofacial reconstruction [11].

The preparatory work for these procedures, i.e. the segmentation of the computed tomography scans and the virtual reduction of the fractures, is time-consuming. We propose to structure the preparatory work according to the stages presented in Figure 1. Segmentation is concerned with the generation of bone models from the CT scans. On these models, further optimization techniques may be applied to obtain adequate representation for subsequent stages. The virtual reduction stage focuses on the repositioning and aligning of bones and fragments. Since most registration approaches only work for two fragments at once, a pairwise approach is used. The matching stage selects two fragments to be registered to each other. After registration, the new alignment of these fragments can be taken into account at the matching phase [12]. Eventually a fully reduced fracture is generated. A more detailed version of this pipeline is presented and discussed in Chapter 1.

The research presented in this paper is focused on the pairwise registration of fracture fragments. The problem of registering two fragments to each other is akin to solving a three-dimensional puzzle [13], with the added difficulty that multiple fracture surfaces may be irregular and that they may not match fully due to comminution [14].

The problem of registering two bone fragments to each other is comparable to the problem of shape matching that has been encountered with range images. These images result from attempts to digitally capture physical objects. Since few objects can be described by a single range image, multiple images have to be combined to arrive at a complete digital description. Formally, the registration problem for range images is the computation of a rigid transformation that brings the points of one range image into alignment with another range image [15]. Several solutions to this problem have been proposed [15–18]. Another problem that is comparable is the reconstruction of broken artifacts. However, compared to range image matching, several new challenges are introduced. Firstly, data is lost due to tiny disappearing fractured pieces or due to deterioration of material surfaces. Furthermore, a single fragment can combine with any number of other fragments, making the exact matching relationship hard to define [19]. Orthopedic reconstruction is even more challenging than the reconstruction of general artifacts. To begin with, computed tomography data are less accurate than the laser scan data that are in most cases used for general artifacts [20–22]. The segmentation of the CT data into three-dimensional meshes amplifies the noise present in an unpredictable manner. Secondly, bones break differently than archaeological artifacts. They tend to splinter, generating pieces that have a single smoothly varying fracture surface that may correspond to one or multiple fragments. Archaeological artifacts typically consist of hard brittle materials that often break in such a way that fracture surfaces may be extracted by their geometric properties [23].

Before introducing our approach to orthopedic reconstruction we review earlier work on this subject in Section 2. The fracture reduction method we propose is the subject of Section 3. Sections 4 to 6 introduce the data used to test the approach, present the result of the experiment, and discuss them, respectively. Section 7 concludes the paper and Section 8 presents some ideas for new research based on this work.

2 Related Work

Jiménez-Delgado et al. [12] observed that most preoperative orthopedic reduction tools focus on the long bones. The shaft of these bones is cylindrical and completely surrounded by cortical tissue. At the ends of these bones the cortical tissue is very thin, hence cancellous tissue can appear in the outer part of the bones. This extra information has favored the development of computer assisted methods for the reduction of fractures of long bones. Very little research has looked at reducing irregular bones such as the acetabulum. Since this extra information is not available from the acetabulum, we do not consider registration methods that use it.

We structured this section based on another classification of preoperative orthopedic fracture reduction applications by Jiménez-Delgado et al. [12], namely one that distinguishes based on the used registration method. Approaches that use a statistical template, discussed in Section 2.1, register the fragments of the broken bone to an anatomical atlas of the healthy bone. The methods discussed in Section 2.2 use a physical template, for example the mirrored contralateral bone or a pre-existing scan of the healthy bone. Finally, in Section 2.3 we discuss geometric approaches to fracture reduction. These methods use properties of the 3D models representing the bone fragments to register bone fragments to each other.

2.1 Statistical Template

Moghari and Abolmaesumi [24] register fragments of a fractured humerus to an anatomical atlas of that bone in a local registration step, after which a global registration step is used to register the fragments to each other, and the template. The statistical shape model (SSM) of the humerus is generated with principal component analysis of a small population of CT scans of whole humeri. Kovler et al. [14] argue that this approach may be inadequate for far-from-average cases. Furthermore, a precomputed atlas might not always be available. A final issue with this approach is that a statistical model is not necessarily representative for the whole population. Wu et al. [25] note that there are large differences between the humeri of Asian and Western patients.

Albrecht and Vetter [26] propose the use of a statistical shape model only if a contralateral or pre-fracture scan is not available. Their iterative method alternates between aligning the main fracture fragments to the template with iterative closest point (ICP), and adapting the SSM to the individual's anatomy. A disadvantage of this method is that the user has to manually input the length of the bone, as the algorithm will fail otherwise. Furthermore, the method fails completely when it is used for the fracture reduction of a deformed bone.

An alternative to the anatomical atlases used by Moghari and Abolmaesumi [24] and Albrecht and Vetter [26] is offered by Oura et al. [27]. They predict the shape of the whole bone from its partial shape using partial least squares regression. The accuracy of the resulting prediction is similar to that of methods based on bilateral symmetry. An advantage of this method is that it is not sensitive to initialization, since it is deterministic. However, it is susceptible to inter-operator errors, as manual identification of landmarks is required.

All reviewed statistical methods have difficulty handling non-standard bones. Although Albrecht and Vetter [26] base the statistical bone on numerous scans and adapt the model to the patient's anatomy, they still had problems with non-standard anatomies, suggesting that statistical models have to be based on an extremely high number of scans, which may not be available [14].

2.2 Physical Template

Okada et al. [28] evaluate different fracture reduction methods on femoral head fractures. One of these approaches registers fracture fragments to the contralateral bone. They found that this approach shows a large rotation error compared to the other methods, even after removing all cases where registration failed completely, which mostly occurred with contralateral matching.

The method based on the same idea introduced by Fürnstahl et al. [29] outperforms the one by Okada et al. [28] in all experiments on proximal humerus fractures. However the approach by Okada et al. [28] has a significantly lower runtime and requires fewer user-defined parameters.

There are two main disadvantages to registering to the physical bone. First, when using the healthy contralateral bone as a template for fracture reduction one assumes bilateral symmetry. Although Berg et al. [30] found scaphoid poles to be sufficiently symmetrically aligned to serve as a reference in surgery, others have found that humeri [31–33], radii, and ulnae [33, 34] show sufficient within-patient lateral asymmetry to caution against blindly using the contralateral bone as a template for surgical reconstruction [32]. Second, in most cases the approach using a contralateral bone requires an additional CT scan of the uninjured bone, which increases the patient's radiation exposure [29].

2.3 Geometric Methods

Buschbaum et al. [35] register fragments to each other by detecting fracture lines; the lines separating the strongly curved surfaces lines from the smooth surface of the bone. Based on computed surface curvatures they extract and connect points in areas of high convex curvature. The resulting opposite fracture lines are mapped together based on the surface normal vector, and the first and second principal curvature direction. The errors of the resulting registrations fall within the clinically acceptable range.

Winkelbach et al. [36] only use curvature supplementally. Their registration method uses a random sample matching approach, based on the random sample consensus (RANSAC) algorithm. This repetitive procedure generates a likely hypothesis from the input dataset, and subsequently evaluates the quality of the registration, which is expressed as the number of contact points between the fragments [37]. This method maximizes surface contact and minimizes penetration between bone fragments. It cannot be generalized to all bones, for example it is not suitable for long bones, since fractures of these bones have a large area that is not necessarily part of the fracture area [14]. Furthermore, due to the infinite search loop used by RANSAC it is impossible to decide whether the optimal result has been found [37].

Willis et al. [23] propose an interactive method for global registration of highly comminuted bone fractures. Pairs of coarsely corresponding user-selected fracture-surface patches are aligned to each other using ICP. This algorithm iteratively refines the transformation that aligns one point cloud to another by minimizing an error metric, generally some distance measure between matched pairs of points from the two point clouds. It is guaranteed to always monotonically converge to the nearest local minimum [38]. Consequently, it is sensitive to the initial alignment of the fragments. Zhou et al. [39] improve the method proposed by Willis et al. [23] in several aspects. Firstly, surfaces belonging to the same bone fragment are grouped to prevent oscillations in the pairwise registration. Furthermore, the introduce a new subsampling method that allows idiosyncratic geometric surface variations to more heavily influence the final registration. Contrary to methods discussed earlier, Willis et al. [23] and Zhou et al. [39] use an interactive approach. The advantage of such an approach is that the user is able to influence the reconstruction process and select the best option of multiple potential reductions [40].

Manual annotation is also used by Okada et al. [28]. Instead of manually selecting fracture surfaces, users are asked to find the fracture lines by steering 3D line tracking software that operates on 3D curvature images of the bone fragments. These line fragments are used as input to the ICP algorithm that registers the fragments to each other.

Kovler et al. [14] do not use manual identification of fracture surface pairs, instead they extract them automatically by searching for connected components of high density in the CT scan. Outliers are removed by only keeping surfaces whose maximal principal curvature is higher than some threshold. They compute the coarse alignment of the fragments with a principal component analysis (PCA)-based method. The finer final alignment is computed with ICP. Users have the option to incorporate further clinical considerations by manually changing the fracture reduction.

A new method to find corresponding points between fragments for ICP is introduced by Chowdhury et al. [41]. They use complete bipartite graph matching to find correspondences such that no two pairs share a common point, to avoid distortion of the fracture shape. They try to ensure that the ICP algorithm finds the global minimum of the registration by generating multiple initial orientations for each fragment. Possible initializations are constructed from the fragments' bounding boxes. The best initializations are selected using local and global shape constraints [29]. This alternative initialization of the conventional ICP algorithm results in improved accuracy and convergence compared to the normal implementation on the five tested clinical datasets.

It seems that ICP is the most used geometric registration method [14, 23, 41, 42], although some alternative methods have been introduced [35, 36]. Strikingly, the alternative methods are all fully automatic, whereas only half of the discussed methods using ICP do not require user input. Since the fully automatic approaches using ICP compute an initial alignment before starting the iterative method, we expect that this difference is due to the sensitivity of the ICP algorithm to local minima.

3 Method

Iterative closest point aims to find the linear transformation matrix $\boldsymbol{M}$ that transforms a model fragment $\mathscr{Y}$ to best match a static fragment $\mathscr{X}$. The algorithm starts with an initial guess for the rigid-body transform between the two models, and iteratively refines it by repeatedly generating pairs of corresponding points on the meshes and minimizing an error metric until some termination criterion is met. Many variants of the algorithm have been introduced since its inception by Chen and Medioni [16] and Besl and McKay [43]. Rusinkiewicz and Levoy [44] identify the following stages in the algorithm:

1. *Selection* of some set of points in one or both of the meshes.
2. Matching these points to samples in the other mesh, to generate *correspondences*, pairs of points, with one point contributed from each mesh.
3. *Weighting* the resulting correspondences appropriately.
4. Computing the value of an *error metric* based on the point pairs.
5. *Minimizing* the error metric.

We add a final stage, namely stopping the algorithm when some *termination criterion* is met.

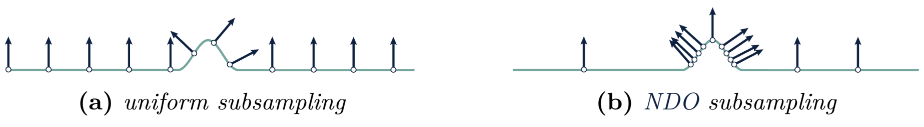

(a) *uniform subsampling* (b) *NDO subsampling*

Figure 2: *The difference between (a) uniform and (b) normal distribution optimization (NDO) sampling illustrated. Observe how the small feature is overwhelmed by the surrounding flat area if uniform subsampling is used. Images adapted from Rusinkiewicz and Levoy [44].*

Due to its sensitivity to local minima in the error, the algorithm may not converge if the two models are placed too far away from each other [42]. Therefore we take the user-manipulated pose of the fragments in the virtual environment as the initial registration.

Up until now we have only discussed the registration of two models to each other. To handle multiple models we use pairwise matching. This entails that to match for example three fragments, different pairs of those three fragments need to be matched to each other, while one of the fragments is not considered.

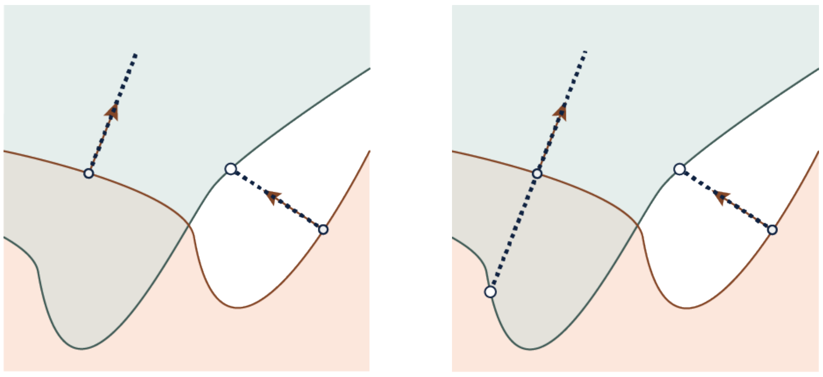
Stages 1 and 2 are discussed in Sections 3.1 and 3.2, respectively. We do not weigh points, since Rusinkiewicz and Levoy [44] found in their comparison of different variants of the ICP algorithm that this hardly influences the convergence rate. The different error metrics and their minimization is the subject of Section 3.3. Finally the termination of the introduced ICP variant is discussed in Section 3.4. The variant of the ICP algorithm presented in this section has been implemented with Unity [45].

3.1 Sampling

We have used two different sampling methods, namely all-points sampling and normal distribution optimization (NDO) subsampling. The first method simply uses all available points. This approach is only used if the models are relatively small, i.e. if they have fewer than 1000 vertices.

NDO subsampling is used for all larger meshes. This method chooses points such that the distribution of normals among the selected points is a large as possible; if one were to bin the normals in angular space after subsampling each bin would have approximately the same number of elements. Consequently, small features, which can be vital for the determination of the correct registration, do not disappear [44]. For example, in case of a mesh representing a small ridge in an otherwise flat surface, uniform subsampling results in a sample containing mostly normals from the flat surface. NDO subsampling with three bins produces a set of samples which consists in equal number of vertices coming from the flat surface, the left side and the right side of the ridge. The effect of features that disappear using uniform subsampling, and that stay visible with NDO subsampling is illustrated in Figure 2.

We have implemented NDO subsampling by binning all vertices according to the direction of their normals and then sampling as uniformly as possible across the bins. To bin the vertex normals into n bins, each bin is associated with a face



(a) *original normal shooting* **(b)** *adapted normal shooting*

Figure 3: *(a) The problem of the original normal shooting algorithm and (b) the used adapted version. The dashed lines indicate the cast rays that are used to find the intersection. The static fragment boundary is shown in orange, the model fragment boundary in green.*

normal of a uniform polyhedron. The exact polyhedron that is used depends on the choice of n. A vertex normal $\mathbf{n}$ is added to the bin associated with the face normal $\mathbf{r}_i \in \{\mathbf{r}_1, \ldots, \mathbf{r}_n\}$ such that

$$\arg\max_{i\in[1,n]} \mathbf{n} \cdot \mathbf{r}_i .$$

3.2 Correspondences

To find correspondences, we use an adaptation of the normal shooting method proposed by Chen and Medioni [16]. The original method finds the intersection with the model fragment of the ray originating at a vertex on the static model in the direction of the vertex normal. A disadvantage of this approach is that it cannot find correspondences in areas where the two models intersect. To solve this, we first cast a ray in the direction of the normal. If that does not result in an intersection we shoot in the flipped direction. This problem, and the proposed solution, are illustrated in Figure 3.

The adapted correspondence finding method has been implemented with the Unity [45] methods to detect collisions between mesh colliders and rays. Due to the application programming interface (API) of that functionality we are required to set a maximum within correspondence distance. We have set this distance in two ways, namely to a fixed distance, and to a distance that is a percentage of the bounding box of all fragments of the fracture. The advantage of the latter approach is that it is independent of the scale of the meshes that represent the fracture fragments. The second step of the correspondence finding is the filtering of correspondences. Due to Unity's API, a filter on the within-correspondence distance is implicitly implemented. This filter rejects any correspondence for which the distance between the point sampled from the static fragment and the point sampled from the model fragment is greater than some threshold. We have not added any other filters.

3.3 Error Metric

ICP aims to find the registration between two models by iteratively minimizing some error metric. This metric generally depends on some aggregation of the errors associated with the different correspondence pairs.

We consider three different error metrics. For one metric, the point-to-point error, two minimization methods have been implemented. In Section 3.3.1 we discuss the point-to-point error and its two minimization methods. The point-to-plane error is presented in Section 3.3.2, and finally the newly proposed intersection error is the subject of Section 3.3.3.

3.3.1 Point-to-Point Error

The point-to-point error of the correspondence $\mathcal{C} = \langle \mathbf{x}, \mathbf{y} \rangle$ with $\mathbf{x} \in \mathscr{X}$ and $\mathbf{y} \in \mathscr{Y}$ and the transformation matrix $\boldsymbol{M}$ is defined as

$$e_{\text{po}}\left(\mathcal{C}\right) = \left|\, \boldsymbol{M}\mathbf{x} - \mathbf{y} \,\right|^2. \tag{1}$$

Several closed-form solutions that minimize the sum of the point-to-point error have been proposed [46–49]. A comparison of these methods by Eggert et al. [50] showed that the differences among them with respect to numerical accuracy and stability are small. We have chosen to use the method introduced by Horn [47] using unit quaternions. Given the set of correspondences $\mathscr{C} := \{\, \mathcal{C}_i = \langle \mathbf{x}_i, \mathbf{y}_j \rangle \mid i = 0, 1, \dots, N \wedge \mathbf{x}_i \in \mathscr{X} \wedge \mathbf{y}_j \in \mathscr{Y} \,\}$, this method finds the translation vector $\mathbf{t}$ and the rotation matrix $\boldsymbol{R}$ such that

$$E_H = \sum_{i=1}^{N} \left|\, \mathbf{y}_i - \boldsymbol{R}\left(\mathbf{x}_i - \mathbf{x}_c\right) - \mathbf{t} \,\right|^2 \tag{2}$$

is minimized, where $\mathbf{x}_c$ is the centroid of $\mathscr{X}$. Horn [47] showed that $\mathbf{t}$ is the difference between the centroids of $\mathscr{X}$ and $\mathscr{Y}$. The rotation is found by constructing the cross-covariance matrix between zero-centered pairs of points. The final rotation is defined as the eigenvector corresponding to the eigenvalue of a matrix that is built from the cross-covariance matrix [15]. We refer to the minimization of the point-to-point-error with the method proposed by Horn [47] as point-to-point closed form (CF$_{\text{po}}$).

As an alternative to the closed-form solution, we have investigated an iterative approach to minimizing Equation (2). We have chosen to use iterative gradient descent (IGD) [51]; better results may be achieved with more sophisticated iterative methods. Wheeler and Ikeuchi [52] introduce the prerequisites for using gradient methods to minimize Equation (2). They define the following least squares function:

$$E_W = \frac{1}{4N} \sum_{i=1}^{N} \left(\boldsymbol{R}\left(\mathbf{q}\right)\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\right)^2 \tag{3}$$

where $\boldsymbol{R}\left(\mathbf{q}\right)$ represents the 3×3 orthonormal rotation matrix defined by the unit quaternion $\mathbf{q}$. The factor $4N$ is introduced for aesthetic reasons. The partial derivatives of this function w.r.t. the quaternion $\mathbf{q}$ and the translation vector $\mathbf{t}$ are

$$\frac{\partial E_W}{\partial \mathbf{q}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{R}(\mathbf{q})\,\mathbf{x}_i \times (\mathbf{t} - \mathbf{y}_i)\,, \tag{4}$$

$$\frac{\partial E_W}{\partial \mathbf{t}} = \frac{1}{2N}\sum_{i=1}^{N} \boldsymbol{R}(\mathbf{q})\,\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i. \tag{5}$$

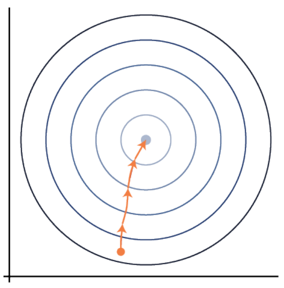
Using these equations the update rule for the translation vector used in IGD becomes

$$\mathbf{t}' = \frac{\lambda}{\zeta}\frac{\partial E_W}{\partial \mathbf{t}}, \tag{6}$$

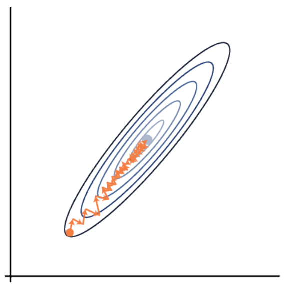
where λ denotes the learning rate, and ζ is a scaling factor.

The scaling factor corrects for the dissimilar scaling behavior of rotation and translation in rigid body motion. Setting the scaling parameter to one would result in an error landscape with a canyon, which makes gradient-based search methods inefficient [52]. Figure 4 shows how a lack of normalization causes such a canyon which results in the IGD algorithm requiring more steps to reach convergence. To compute ζ, the ranges of the points in $\mathscr{X}$ are computed in each dimension. ζ is set to the size of the largest range.

The learning rate indicates the size of the steps taken in the direction of the gradient, i.e. the length of the arrows in Figure 4. Setting this parameter to a large value can result in fast convergence. However, it might also cause the algorithm to continuously overshoot the correct value, resulting in an oscillation around the correct solution. Choosing a too small learning rate generally avoids these oscillations, but it can make convergence slow. We have set λ to the empirically determined value 0.001.

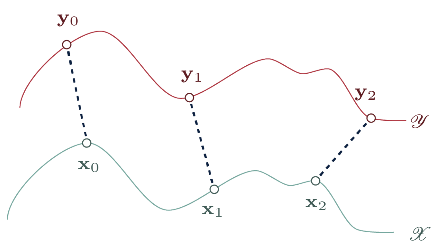


(a) *normalized*

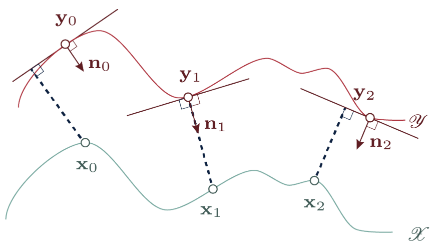


(b) *unnormalized*

Figure 4: *Illustration of the difference between performing iterative gradient descent, (a) with and (b) without the scaling factor ζ. Shown is a 2D error landscape, the contours of which are indicated by the blue circles. The steps of the 'path' the IGD algorithm takes are shown by orange arrows. This path leads to a light blue dot, which represents the local minimum nearest to the initialization position, indicated by an orange dot.*



(a) *point-to-point*



(b) *point-to-plane*

Figure 5: *Illustration of (a) the point-to-point and (b) the point-to-plane error. The dashed lines connect correspondences between points sampled from the model fragment $\mathscr{X}$ and the static fragment $\mathscr{Y}$. The point-to-plane image also shows the vertex normals of the static points. Images adapted from Low [53].*

The update rule for the rotation, expressed as a quaternion, is

$$\mathbf{q}' = -\frac{\lambda}{\zeta^2}\frac{\partial E_W}{\partial \mathbf{q}}. \tag{7}$$

The scalar value of the resulting quaternion $\mathbf{q}'$ should be one, since the gradients are implicitly evaluated at the identity quaternion. To handle floating point errors we set the scalar value of the quaternion to 1 after every iteration.

IGD terminates if the current error, defined in Equation (3), is lower than the convergence error, or if a fixed number of iterations has been reached. We have set this last number to a relatively low value, namely 200, since any possible errors will be improved upon in further iterations of the ICP algorithm. The resulting error metric and minimization combination is referred to as point-to-point iterative gradient descent (IGD_{po}).

3.3.2 Point-to-Plane Error

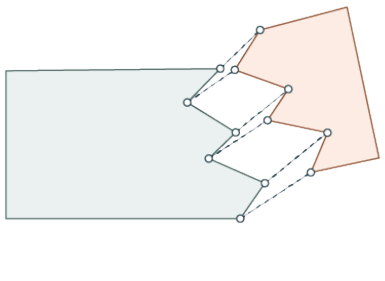
The ICP version introduced by Chen and Medioni [16] uses the point-to-plane error. This metric computes the squared distance from each point on the model shape to the plane that is perpendicular to the static point's normal, which contains the static point. For the correspondence $\mathcal{C}$ the point-to-plane is

$$e_{\text{pl}}(\mathcal{C}) = \left((\boldsymbol{R}\mathbf{x} + \mathbf{t} - \mathbf{y}) \cdot \mathbf{n}\right)^2, \tag{8}$$

where $\mathbf{n}$ is the normal of the point $\mathbf{y}$ sampled from $\mathscr{Y}$. Figure 5 compares this error metric to the point-to-point error. If no vertex normals are included with the fragments we compute them according to Newell's method [54].

No closed-form solutions are available for this error metric [44]. The least-squares equations derived from Equation (8),

$$E_L = \sum_{i=1}^{N} e_{\text{pl}}(\mathcal{C}_i) \tag{9}$$



(a) *non-intersecting fragments* (b) *intersecting fragments*

Figure 6: *The sum of square distances between the shown correspondences in (a) the case where the fragments do not intersect and (b) the case where they do is the same. As a result both the point-to-point and the point-to-plane error assign the same error to these different cases.*

may be solved using a generic non-linear method, for example Levenberg-Marquardt [44]. However these methods are computationally expensive. If the rotation that minimizes E_L is small, Equation (9) can be solved by linearizing the rotation matrix. A 3D rigid-body transformation matrix is defined as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \boldsymbol{R}_z(\gamma)\, \boldsymbol{R}_y(\beta)\, \boldsymbol{R}_x(\alpha)\,, \tag{10}$$

where $\boldsymbol{R}_z(\gamma)\,,\, \boldsymbol{R}_y(\beta)\,,\, \boldsymbol{R}_x(\alpha)$ denote the rotation of α, β, γ radians around respectively the x, y and z-axes. When we assume that the rotation θ is small we can use the approximation $\sin(\theta) \approx 0$ and $\cos(\theta) \approx 1$. This allows $\boldsymbol{M}$ to be approximated as the linear matrix

$$\hat{\boldsymbol{M}} = \begin{bmatrix} 1 & -\gamma & \beta & t_x \\ \gamma & 1 & -\alpha & t_y \\ -\beta & \alpha & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

Using $\hat{\boldsymbol{M}}$, linear least squares can be used to find the translation vector $\mathbf{t}$ and the rotation around the x, y, and z-axes, α, β and γ, respectively. The final transformation matrix should be computed by plugging these values into Equation (10), since using Equation (11) may result in an invalid transformation matrix [53]. This linear approximation is equivalent to treating the transformation of the points sampled from $\mathscr{X}$ as a displacement by a vector $[\,\mathbf{r} \times \mathbf{x} + \mathbf{t}\,]$, where $\mathbf{r} = [\,r_x, r_y, r_z\,]$ is a vector of rotations around the x, y, and z-axes [55]. We refer to the minimization of the point-to-plane error with this minimization metric as point-to-plane closed form ($\mathrm{CF}_{\mathrm{pl}}$).

3.3.3 Intersection

As illustrated in Figure 6, both the point-to-point and the point-to-plane error metrics are insensitive to intersections between fragments. In the context of fracture registration, the reduction shown in Figure 6(a) without intersections is superior to the one with intersections in Figure 6(b). However, this is not reflected in the point-to-point or point-to-plane error. Therefore, we introduce the intersection error

$$e_{\text{int}}\left(\mathcal{C}\right) = \omega_D \left(\boldsymbol{M}\mathbf{x} - \mathbf{y}\right)^2 + \omega_I \xi \left|\, \boldsymbol{M}\mathbf{x} - \mathbf{y} \,\right|^2, \tag{12}$$

where ω_D and ω_I denote the weights of the distance and intersection term, respectively. We have set both terms to 0.5, giving them equal weight. The factor ξ in the intersection term is 0 if $\mathbf{x}$ does not fall within the static model, and 1 otherwise.

To obtain the factor ξ we need to determine if $\mathbf{x}$ lies within the static fragment, $\mathscr{Y}$. Since the fragments may be concave, we cannot use the sign of the dot product to determine containment, and have to use more complex methods. Consequently we cannot use a closed-form solution to determine the translation, we use an iterative method instead. If the axis-aligned bounding boxes (AABBs) of the two fragments do not overlap, we set all $\{\, \xi \mid i = 0, 1, \ldots, N \,\}$ to zero. An overview of more sophisticated methods of object collision is given in Chapter 3. If there is some overlap between the AABBs of the fragments, we use a two-stage approach to determine the value of ξ_i for every correspondence. In the first, computationally cheap, stage we check if $\mathbf{x}$ falls within the AABB of $\mathscr{Y}$. Only if this is the case do we count the intersections with $\mathscr{Y}$ on the line from $\mathbf{x}$ to a random point outside of $\mathscr{Y}$'s bounding box. If the number of intersections is odd, $\mathbf{x}$ is contained within the static fragment, and ξ is set to 1. If the number of intersections is even, or if $\mathbf{x}$ does not lie within the bounding box, ξ is set to zero.

To minimize the error in Equation (12) iteratively we introduce the least squares function

$$E_I = \sum_{i=1}^{N} \left(\omega_D + \omega_I \xi_i\right) \left(\boldsymbol{R}\left(\mathbf{q}\right) \mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\right) \tag{13}$$

which, following the method discussed by Wheeler and Ikeuchi [52], has the following partial derivatives w.r.t. $\mathbf{q}$ and $\mathbf{t}$

$$\frac{\partial E_I}{\partial \mathbf{q}} = \frac{1}{N} \sum_{i=1}^{N} \left(\omega_D + \omega_I \xi_i\right) \left(\boldsymbol{R}\left(\mathbf{q}\right) \mathbf{x}_i \times \left(\mathbf{t} - \mathbf{y}_i\right)\right), \tag{14}$$

$$\frac{\partial E_I}{\partial \mathbf{t}} = \frac{1}{2N} \sum_{i=1}^{N} \left(\omega_D + \omega_I \xi_i\right) \left(\boldsymbol{R}\left(\mathbf{q}\right) \mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\right). \tag{15}$$

Given these partial derivatives we use the iterative approach discussed in Section 3.3.1 to find the transformation at each iterative closest point iteration. We name this new error metric and its minimization intersection iterative gradient descent (IGD$_\text{i}$).

3.4 Termination

We terminate the ICP algorithm if at least one of three conditions is true:

(i) We have executed the maximum acceptable number of iterations.

(ii) The error of the current iteration is lower than some threshold.

(iii) The error has stabilized according to some criterion.

Finally, the algorithm also terminates if fewer then six correspondences are found. The rationale behind this is that at least six observations are needed to determine six unknowns.

Condition (i) is true if the number of ICP iterations is greater than 500. This threshold ensures a reasonable computation time for each pairwise registration.

The termination condition (ii) halts the algorithm if the current registration error is lower than some threshold. Although the closed-form methods minimize the sum of errors, the convergence error is always computed as the mean of the correspondence errors. This avoids undue influence of the number of found correspondences on the convergence. The value of the error threshold is set to ψ times the initial registration error. The first advantage of this approach is that it ensures that the error threshold always makes sense in the context of the data. Secondly, contrary to a fixed value, it allows the user to rerun the algorithm if the previous run terminated because of Condition (ii) without a satisfactory result. In a clinical, rather than experimental, context, one might set a maximum value for this threshold, to ensure that it is never larger than the clinically acceptable reduction distance.

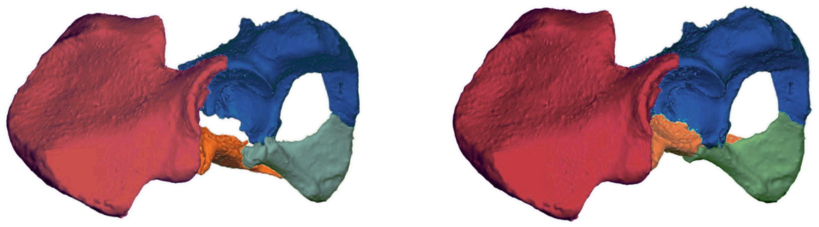
Finally, Condition (iii) terminates the algorithm if the error has stabilized, i.e. if continuing the computations is not likely to improve the final registration. To determine stabilization we store the past 50 iteration errors in a set $\mathscr{S}$. We consider the registration to be stable if

$$\mathscr{S}_\sigma \frac{1}{\mathscr{S}_\mu} < 5 \times 10^{-8}\,, \tag{16}$$

where $\mathscr{S}_\mu$ and $\mathscr{S}_\sigma$ denote the mean and biased standard deviation of $\mathscr{S}$, respectively. The standard deviation is scaled with the mean to ensure scale invariance.

4 Experiment

Other than the actual registration, one of the challenges of research in this field is the quantification of results, since a ground truth is generally not available for clinical data. Some are satisfied with a visual inspection of the results [23, 39]. Others go through the trouble of physically breaking fake bones and scanning the resulting fractures [13, 35]. An often used approach is to make a CT scan of healthy bones, break them virtually and manually transform one of the fragments [13, 14, 56]. Finally Okada et al. [28] use clinical fracture data and



(a) *preoperative model* **(b)** *reduced model*

Figure 7: *Visualization of an acetabular fracture (a) before and (b) after virtual reduction. Images adapted from Meesters [57].*

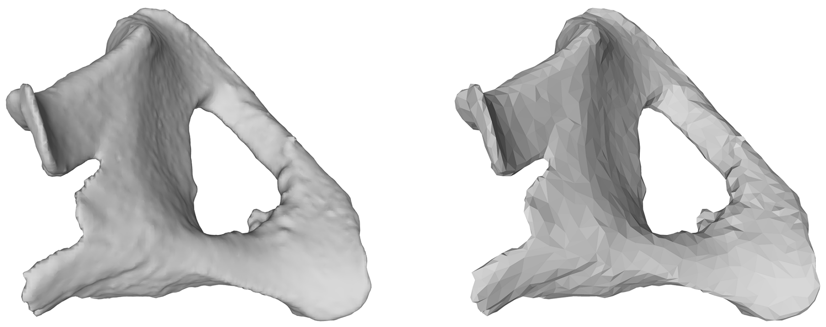
compare the results of their automatic classification with the manual reduction of the same fracture by a medical professional.

We use a mix of these approaches. The generation of the dataset is discussed in Section 4.1. Section 4.2 introduces the method used to evaluate the quality of a reduction. We have performed a comparable experiment with simulated data, which has significantly different outcomes. This experiment is discussed in Chapter 4.

4.1 Data

We use the clinical data of eight patients treated for an acetabular fracture at the UMCG. For each patient, the 3D models of the fracture fragments generated from the preoperative CT scan and the manual virtual reduction by Meesters et al. [58] are available. We use the virtual reduction by Meesters et al. [58] as our ground truth. Figure 7 a 3D model of an acetabular fracture and its virtual reduction.

Preoperative CT scans are made with 512×512 slices, with a thickness in the range 0.6 mm to 2 mm. Table 1 presents the used slice thickness per patient. Based on these CT scans, three-dimensional models are generated with Mimics



(a) *original mesh* **(b)** *downsampled mesh*

Figure 8: *(a) The original mesh of an acetabular fracture fragment, with 90332 faces, and (b) the same mesh downsampled to 7500 faces.*

		fragment										
	ρ	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
$\mathcal{F}_A$	1.00	45.40	49.03	49.45								
$\mathcal{F}_B$	1.00	13.93	42.28	45.85	46.88							
$\mathcal{F}_C$	0.60	29.80	47.66	48.61	49.64							
$\mathcal{F}_D$	1.25	34.28	44.92	46.80	47.67	48.73	49.51					
$\mathcal{F}_E$	2.00	0.00	39.24	44.08	45.45	48.52	49.36					
$\mathcal{F}_G$	2.00	33.60	46.70	47.18	47.54	48.55	49.53					
$\mathcal{F}_H$	1.00	32.87	40.32	47.10	49.10	49.20	49.29					
$\mathcal{F}_I$	1.00	0.00	10.50	34.27	41.01	41.35	44.10	47.31	47.35	48.00	49.08	49.08

Table 1: *The slice thickness, ρ, in* mm *and the percentual reduction in vertex count due to subsampling for the different fragments in $\mathcal{F}_A$, $\mathcal{F}_B$, $\mathcal{F}_C$, $\mathcal{F}_D$, $\mathcal{F}_E$, $\mathcal{F}_G$, $\mathcal{F}_H$, $\mathcal{F}_I$. The shading of the cells indicates the first (●), second (●), third (●), and fourth (●) quantile.*

Medical [59]. First, bone tissue is segmented using a threshold based technique. Loose voxels with the same Hounsfield unit (HU) as bone are removed with region growing. Fracture fragments are split manually with the aid of the split tool provided by Mimics Medical [57]. For the purposes of this experiment the meshes were processed further, using the wrap tool implemented in Materialise 3-matic [60] with the gap closing distance set to 1.0 and the smallest detail to 0.5, otherwise the default parameters were used.

Since these meshes have too many vertices to be processed by Unity, they have to be downsampled before they can be imported in the our application. Using the Quadric Edge Collapse Decimation implemented in MeshLab [61] we downsample each object until it has 7500 faces. Next, MeshLab is used to remove all connected components composed of fewer than 5000 triangles. Figure 8 shows a fracture fragment before and after downsampling. Table 1 presents the percentual downsampling rate per fragment per dataset. This table also includes the slice thickness of the CT scans the fragments are based on.

We add noise to the transformations of the fragments of the manually reduced fractures to simulate a user placing them in an approximately correct position, before letting the application find the optimal registration. Fragments are rotated around the x, y, and z-axis with an angle in the range $[-5°, 5°]$ sampled from a uniform distribution. For the translation we compute the oriented bounding box (OBB) of the fragment. Let w be the size of the OBB in some dimension, the translation in that dimension is then sampled from a uniform distribution with the range $[-0.001w, +0.001w]$.

The order in which fragment pairs are registered is determined based on their vertex count. We first register the fragment with the fewest vertices to the one with the most, then to fragment with the second most, and so on, until it has been registered to all other fragments. Next, the next smallest fragment is registered to all other fragments that have a higher vertex count in the same way. This is repeated until the largest fragment has been registered to the second largest fragment. The process is then executed once more to reach the final registration. It should be noted that better results may be achieved with one of the more sophisticated approaches to multi-fragment registration. For example the one used by Fürnstahl et al. [29].

We used an adaptive error threshold with $\psi \in \{0.2, 0.6\}$ as one of the termination conditions.

4.2 Quantification

We take the same approach as Kovler et al. [14] and Chowdhury et al. [41] and use the Hausdorff distance to compare the quality of registrations. Between two meshes $\mathscr{A}$ and $\mathscr{B}$, the Hausdorff distance is [62]

$$D_H\left(\mathscr{A}, \mathscr{B}\right) = \max\left(D_h\left(\mathscr{A}, \mathscr{B}\right), D_h\left(\mathscr{B}, \mathscr{A}\right)\right), \tag{17}$$

where

$$D_h\left(\mathscr{A}, \mathscr{B}\right) = \max_{\mathbf{a}\in\mathscr{A}}\left(\min_{\mathbf{b}\in\mathscr{B}}|\mathbf{a}-\mathbf{b}|\right). \tag{18}$$

The directed Hausdorff distance, defined in Equation (18), is computed with the implementation provided by MeshLab [63].

5 Results

The experiment described in Section 4 results in a dataset with two factors, namely the four used registration methods and ψ, the scaling factor used to compute the adaptive error threshold. The first factor has four levels, one for each of the used registration methods point-to-point closed form ($\mathrm{CF}_{\mathrm{po}}$), point-to-plane closed form ($\mathrm{CF}_{\mathrm{pl}}$), point-to-point iterative gradient descent ($\mathrm{IGD}_{\mathrm{po}}$), and intersection iterative gradient descent ($\mathrm{IGD}_{\mathrm{i}}$). The second factor is binary, its two levels are 0.2 and 0.6.

Let $\mathcal{F} = \{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ be a registration of a fracture with n fragments. To compare two different registrations, $\mathcal{F}_x$ and $\mathcal{F}_y$, of the same fracture we use

$$D_M\left(\mathcal{F}_x, \mathcal{F}_y\right) = \frac{1}{n}\sum_{i=1}^{n} D_H\left(\mathsf{f}_{(x,i)}, \mathsf{f}_{(y,i)}\right), \tag{19}$$

where $\mathsf{f}_{(x,i)}$ denotes the ith fragment of registration $\mathcal{F}_x$. Equation (17) defines $D_H\left(\mathsf{f}_a, \mathsf{f}_b\right)$ as the undirected Hausdorff distance between the two fragments f_a and f_b. For the registration methods defined in Section 3, D_M is computed as the mean undirected Hausdorff distance between the initial alignment of the fragments and the alignment determined by the registration algorithm. The initial alignment is the registration of the fragments after the application of noise to the manual reduction by the expert, as discussed in Section 4.1.

A t-test on the factor ψ shows that the levels 0.2 ($\mu = 0.112$, $\sigma = 0.0839$) and 0.6 ($\mu = 0.104$, $\sigma = 0.0874$) do not differ significantly ($t_{78.0} = 0.358$, $p = 0.721$).

		registration method			
	initial	CF$_{\text{po}}$	CF$_{\text{pl}}$	IGD$_{\text{po}}$	IGD$_{\text{i}}$
$\mathcal{F}_A$	3.000e^{-6}	8.966e^{-2}	6.375e^{-3}	9.759e^{-2}	4.967e^{-2}
$\mathcal{F}_B$	7.500e^{-7}	1.483e^{-1}	1.343e^{-1}	2.289e^{-1}	1.544e^{-1}
$\mathcal{F}_C$	5.250e^{-6}	2.917e^{-7}	2.430e^{-2}	7.916e^{-2}	2.712e^{-1}
$\mathcal{F}_D$	8.333e^{-7}	1.594e^{-2}	5.268e^{-2}	2.026e^{-1}	2.465e^{-1}
$\mathcal{F}_E$	2.000e^{-6}	6.667e^{-7}	7.948e^{-2}	1.445e^{-1}	1.458e^{-1}
$\mathcal{F}_G$	5.833e^{-6}	5.088e^{-2}	3.692e^{-3}	1.870e^{-1}	1.171e^{-1}
$\mathcal{F}_H$	6.333e^{-6}	6.998e^{-2}	3.658e^{-2}	1.283e^{-1}	1.292e^{-1}
$\mathcal{F}_I$	1.273e^{-6}	4.705e^{-2}	3.542e^{-2}	2.273e^{-1}	2.731e^{-1}

Table 2: *The mean undirected Hausdorff distance, D_M, between the ground truth and the initial registration and between the ground truth and the automatic reduction arrived at by the four different registration methods.*

Since the factor ψ does not influence the registration significantly we do not consider it henceforth.

Table 2 presents D_M, the undirected mean Hausdorff distance between the ground truth and the initial registration and the reductions computed by the four registration methods. What stands out most in this table is that all registration methods worsen the initial registration. The closed-form registration methods seem to perform better than the IGD methods on some of the datasets. Most notably, CF$_{\text{po}}$ generates a registration for $\mathcal{F}_C$ and $\mathcal{F}_E$ that is better than the initial registration w.r.t. to D_M.

We perform paired t-tests to see if the differences between the computed registrations and the initial registration are significant at the 0.001 significance level. The results of these tests are presented in Table 3. Firstly, this table confirms our finding from Table 2, that the registration methods deteriorate the initial registration ($\mu = 3.02 \times 10^{-6}$, $\sigma = 2.09 \times 10^{-6}$) w.r.t. D_M. Furthermore, the closed-form methods, CF$_{\text{pl}}$ ($\mu = 0.0414$, $\sigma = 0.0407$), and CF$_{\text{po}}$ ($\mu = 0.0469$, $\sigma = 0.0474$), significantly worsen the initial registration but far less so than the iterative gradient descent methods, IGD$_{\text{po}}$ ($\mu = 0.160$, $\sigma = 0.0514$), and IGD$_{\text{i}}$ ($\mu = 0.176$, $\sigma = 0.0721$).

We consider the influence of the percentual downsampling by comparing the values of $D_H\left(\mathsf{f}_{(I,i)}, \mathsf{f}_{(F,i)}\right)$ for all matched fragments between the initial, $\mathcal{F}_I$, and computed registration, $\mathcal{F}_F$. Since we have already established that the registration method influences D_M, we use four one-way ANOVAs on the factor percentual downsampling, with the four levels indicated in Table 1. We find no significant influence of this factor for CF$_{\text{po}}$ ($F_{3.0,8.0} = 2.15$, $p = 0.172$), CF$_{\text{pl}}$ ($F_{3.0,12.0} = 1.78$, $p = 0.205$), IGD$_{\text{po}}$ ($F_{3.0,30.0} = 1.70$, $p = 0.189$), or IGD$_{\text{i}}$ ($F_{3.0,31.0} = 2.87$, $p = 0.052$). Therefore we conclude that the different registration methods respond in similar ways to the effects of downsampling.

	registration method			
	CF$_{\mathrm{po}}$	CF$_{\mathrm{pl}}$	IGD$_{\mathrm{po}}$	IGD$_{\mathrm{i}}$
initial	p<0.050 (−2.797)	p<0.050 (−2.877)	p<0.001 (−8.794)	p<0.001 (−6.893)
CF$_{\mathrm{po}}$		p=0.809 (0.246)	p<0.001 (−4.570)	p<0.001 (−4.225)
CF$_{\mathrm{pl}}$			p<0.001 (−5.106)	p<0.001 (4.587)
IGD$_{\mathrm{po}}$	p<0.001 (−4.570)			p=0.618 (0.509)

Table 3: *The t-statistic and the p-value of the paired t-tests between the initial alignment and the results of the different registration methods. The degrees of freedom of each test is 34. Cells where the results of the test is significant at the 0.001 level are shaded.*

We do not have sufficient data to make any observations about the influence of the number of fragments or the slice thickness on the quality of generated registrations.

6 Discussion

In Section 5 we found no significant influence of the factor ψ that controls the adaptive error threshold on the registration error. This could be explained by the low number of executions of the registration algorithm that terminated due to termination condition (ii), i.e. the error being lower than the threshold. If we observe the reasons that the different executions of the pairwise registration algorithm terminated, we find that for $\psi = 0.6$ the registration methods CF$_{\mathrm{po}}$ and CF$_{\mathrm{pl}}$ never terminate because the current error is lower than the threshold. Pairwise registration with IGD$_{\mathrm{po}}$ terminates due to the error threshold for 7.9 %, 2.6 %, and 2.5 % of fragment pairs from dataset $\mathcal{F}_H$, $\mathcal{F}_G$, and $\mathcal{F}_I$, respectively. For the other datasets registration with the method IGD$_{\mathrm{po}}$ never stopped due to termination condition (ii). The registration method IGD$_{\mathrm{i}}$ terminates only on dataset $\mathcal{F}_A$ because the error is lower than the threshold, and only in 12.5 % percent of the cases. Decreasing ψ to 0.2 results in even fewer executions of the pairwise registration algorithms terminating due to termination condition (ii), since the threshold is lower than for $\psi = 0.6$. One would expect the performance of IGD$_{\mathrm{po}}$ and IGD$_{\mathrm{i}}$ to be reasonable, if they terminate because the error is below the threshold. However, the directed Hausdorff distances between fragment pairs that terminated due to termination condition (ii) are comparable to those of the other pairs. We ascribe the lack of difference to the error being computed with the correspondences of the current iteration. Using the undirected Hausdorff distance between the two fragments as the error to determine termination might be a better approach, since that metric does not depend on correspondences that differ between iterations.

Table 2 shows that in general all registration methods increase the mean undirected Hausdorff distance for all datasets. One possible cause for this is lack of precision, which is lost in several places. Firstly, Hu et al. [64] observed that for their registration algorithm to succeed slices should not be thicker than

1.25 mm. Dataset $\mathcal{F}_E$ and $\mathcal{F}_G$ were extracted from CT scans with $\rho = 2$ mm. Secondly, due to the constraints imposed by Unity we had to downsample our meshes quite aggressively. The lost information could negatively impact the performance of the registration algorithm. And finally, Unity forces the use of single precision floats to represent floating point numbers, which may have led to missing data. This problem can be solved by scaling the data with some factor > 1 before applying the registration algorithm.

In Table 2 we also find that the performance of registration method CF_{po} is not consistent between datasets. This method reduces the mean undirected Hausdorff distance w.r.t. the initial error for datasets $\mathcal{F}_C$ and $\mathcal{F}_E$, contrary to its performance on the other datasets. One possible explanation for the difference is that $\mathcal{F}_C$ and $\mathcal{F}_E$ have fewer precision issues, for example because their fragments have fewer vertices than those of the other datasets before downsampling. However their percentual vertex reduction is comparable to that of the other datasets. Furthermore, if that was the reason for the difference we would expect dataset $\mathcal{F}_B$ to have a better performance. Another factor that influences the loss of precision is the slice thickness of the CT scans the meshes were generated from. Although $\mathcal{F}_C$ has the best performance and the lowest slice thickness, the fact that $\mathcal{F}_E$ performs comparably, but was generated from slices with $\rho = 2$ mm suggests that the thickness of the slices is unlikely to be the cause of performance difference. It should be noted that we do not have sufficient data to draw definitive conclusions on the influence of the downsampling rate or the slice thickness. Finally, dataset $\mathcal{F}_C$ and $\mathcal{F}_E$ behaved comparably to the other datasets with regard to which condition triggered the termination of a pairwise registration.

A striking observation from Table 3, is the significant difference in performance between the closed-form and the IGD registration algorithms. One possible explanation for this difference is that ICP needs more iterations to converge if the transformation matrix is approximated with IGD instead of computed exactly with a closed-form solution. The only difference between the registration methods CF_{po} and IGD_{po} is the method they use to determine the transformation. Therefore, the significant difference in undirected Hausdorff distance between fractures registered with these methods indicates that the use of IGD instead of a closed-form solution worsens performance. Another factor that supports this explanation is the fact that the pairwise registration methods IGD_{po} and IGD_{i} terminate for 34.1 % and 77.4 % of the fragment pairs because the maximum number of iterations has been reached. Whereas, CF_{po} and CF_{pl} only terminate due to this reason for 4.0 % and 9.4 % of the fragment pairs. To improve the performance of the IGD methods increasing the number of iterations for either IGD or ICP is likely to work. The correct number of iterations for the latter can be found by comparing the undirected Hausdorff distances per iteration between CF_{po} and IGD_{po}. To find the correct threshold value for IGD, one could compare the transformation matrices computed at each iteration by CF_{po} and IGD_{po}. These comparisons should result in reasonable thresholds for both IGD and ICP, that can be applied to IGD_{i} as well.

7 Conclusion

We have introduced two ideas that to the best of our knowledge are new. Namely, the use of the an adaptive error threshold to determine the termination of the ICP algorithm and a new error metric that not only considers the distance between correspondences but that also takes intersections between the two fragments into account.

Unfortunately we cannot draw any definitive conclusion about the usefulness of the adaptive error term, since computational limits meant we had to set the maximum number of allowed iterations for the ICP algorithm quite low, since a higher threshold resulted in unreasonably long run times. Consequently, the pairwise registration algorithm generally terminated due to this threshold before the adaptive error termination condition could have an effect.

The second newly introduced idea is the use of an error metric for ICP that takes intersections between fragments into account. Based on our results we can only conclude that the use of IGD to find the transformation matrix for a single iteration of the ICP algorithm worsens the quality of the final registration. However, this might be due to the maximum number of iterations set for IGD. We have also identified several possible causes other than the method used to find the transformation matrix for the difference in performance in Section 6. The most important of which is that Unity, although very suitable for the implementation of the user interface (UI), is not the best framework to use for high performance computing. Therefore, we conclude that our experiment should be repeated with a new implementation that addresses the identified precision issues and that allows exploration of the correct parameters for the termination of IGD and ICP.

8 Future Work

This section introduces some ideas for future research, based on the presented work. The fracture fragments extracted from CT scans according to the procedure outlined in Section 4.1 are large triangle meshes that have up to 510 700 vertices. One way to handle these large meshes during the registration would be to start the registration with aggressively downsampled fragments, and to use meshes of an increasingly higher resolution as the quality of the registration increases. This would speed up the initial coarse registration which is, in our opinion, unlikely to suffer from the use of low resolution fracture fragments.

Moreover, we propose to take the fracture area into account during the downsampling of the fragments. Since the fracture surfaces of the fragments are the parts that should be registered to each other, as much information as possible should be left intact in those areas. As the non-fracture surface of the fragment is of far less interest to the registration algorithm one could compensate for the higher number of vertices sampled from the fracture area by sampling fewer points from the non-fracture surface. The obtained fracture surface can be included in the ICP algorithm in several ways. One way to do this would be to only sample points to be used in the correspondence finding stage from the identified fracture area. Alternatively, or additionally, one could include a term in the error function that punishes correspondences with points that are not part of a fracture surface. Several approaches to obtaining the fracture surface have been proposed, ranging from interactive [28, 41] to automatic, based on for example the maximum principal curvature and the Hounsfield intensities [14, 23, 29, 56].

Next to using the fracture surface, incorporating the Hounsfield intensities might be useful. A term that punishes differences in Hounsfield intensities could be added to the error metric. Alternatively, similarity of Hounsfield intensities between the points of a correspondence can be used to weigh the correspondence. It should be noted that for bones with a thin cortical layer the inclusion of these values might not improve performance.

We have identified the dependence of termination condition (ii) on the quality of the algorithm used to find correspondences as a possible weak point of our variation of the ICP algorithm. This dependence can be avoided by terminating the registration based on the current value of an error metric that only uses the current registration of the two fragments. One example of such an error metric is the undirected Hausdorff distance. A disadvantage of this approach is that it is computationally expensive.

Finally, our dataset contained two factors for which we did not have sufficient data to determine their influence, namely the slice thickness of the CT scans and the downsampling rate. We recommend that future experiments investigate their influence on the quality of the computed registration.

References

[1] E. Llopis, V. Higuera, P. Aparisi, J. Mellado, and F. Aparisi. "Acute Osseous Injury to the Pelvis and Acetabulum." In: *Musculoskeletal Imaging.* 2015. Chap. 20, pp. 254–274. DOI: 10.1016/B978-1-4557-0813-0.00020-1.

[2] B. Frietman, J. Biert, and M. Edwards. "Patient-reported outcome measures after surgery for an acetabular fracture." In: *Bone Joint J* 100.5 (2018), pp. 640–645. DOI: 10.1302/0301-620X.100B5.BJJ-2017-0871.R3.

[3] P. Giannoudis, M. Grotz, C. Papakostidis, and H. Dinopoulos. "Operative treatment of displaced fractures of the acetabulum: a meta-analysis." In: *The Journal of bone and joint surgery. British volume* 87.1 (2005), pp. 2–9. DOI: 10.1302/0301-620X.87B1.15605.

[4] M. Tannast, S. Najibi, and J. M. Matta. "Two to twenty-year survivorship of the hip in 810 patients with operatively treated acetabular fractures." In: *JBJS* 94.17 (2012), pp. 1559–1567. DOI: 10.2106/JBJS.K.00444.

[5] T. Borg and N. P. Hailer. "Outcome 5 years after surgical treatment of acetabular fractures: a prospective clinical and radiographic follow-up of 101 patients." In: *Archives of orthopaedic and trauma surgery* 135.2 (2015), pp. 227–233.

[6] J. Clarke-Jenssen, O. Røise, S. Ø. Storeggen, and J. E. Madsen. “Long-term survival and risk factors for failure of the native hip joint after operatively treated displaced acetabular fractures.” In: *The bone & joint journal* 99.6 (2017), pp. 834–840. DOI: 10.1302/0301-620X.99B6.BJJ-2016-1013.R1.

[7] B. J. Merema, J. Kraeima, K. Ten Duis, K. Wendt, R. Warta, E. Vos, R. Schepers, M. Witjes, and F. IJpma. “The design, production and clinical application of 3D patient-specific implants with drilling guides for acetabular surgery.” In: *Injury* 48.11 (2017), pp. 2540–2547.

[8] T. Woo, J. Kraeima, Y. O. Kim, Y. S. Kim, T. S. Roh, D. H. Lew, and I. S. Yun. “Mandible reconstruction with 3D virtual planning.” In: *J Int Soc Simul Surg* 2.2 (2015), pp. 90–93.

[9] R. H. Schepers, G. M. Raghoebar, A. Vissink, L. U. Lahoda, W. J. Van der Meer, J. L. Roodenburg, H. Reintsema, and M. J. Witjes. “Fully 3-dimensional digitally planned reconstruction of a mandible with a free vascularized fibula and immediate placement of an implant-supported prosthetic construction.” In: *Head & neck* 35.4 (2013), E109–E114. DOI: 10.1002/hed.21922.

[10] R. H. Schepers, G. M. Raghoebar, A. Vissink, M. W. Stenekes, J. Kraeima, J. L. Roodenburg, H. Reintsema, and M. J. Witjes. “Accuracy of fibula reconstruction using patient-specific CAD/CAM reconstruction plates and dental implants: a new modality for functional reconstruction of mandibular defects.” In: *Journal of Cranio-Maxillofacial Surgery* 43.5 (2015), pp. 649–657. DOI: 10.1016/j.jcms.2015.03.015.

[11] R. H. Schepers, J. Kraeima, A. Vissink, L. U. Lahoda, J. L. Roodenburg, H. Reintsema, G. M. Raghoebar, and M. J. Witjes. “Accuracy of secondary maxillofacial reconstruction with prefabricated fibula grafts using 3D planning and guided reconstruction.” In: *Journal of Cranio-Maxillofacial Surgery* 44.4 (2016), pp. 392–399. DOI: 10.1016/j.jcms.2015.12.008.

[12] J. J. Jiménez-Delgado, F. Paulano-Godino, R. PulidoRam-Ramírez, and J. R. Jiménez-Pérez. “Computer assisted preoperative planning of bone fracture reduction: Simulation techniques and new trends.” In: *Medical Image Analysis* 30 (2016), pp. 30–45. ISSN: 1361-8415. DOI: 10.1016/j.media.2015.12.005.

[13] T. P. Thomas, D. D. Anderson, A. R. Willis, P. Liu, M. C. Frank, J. L. Marsh, and T. D. Brown. “A Computational/Experimental Platform for Investigating Three-Dimensional Puzzle Solving of Comminuted Articular Fractures.” In: *Computer Methods in Biomechanics and Biomedical Engineering* 14.3 (2011), pp. 263–270. DOI: 10.1080/10255841003762042.

[14] I. Kovler, L. Joskowicz, Y. A. Weil, A. Khoury, A. Kronman, R. Mosheiff, M. Liebergall, and J. Salavarrieta. “Haptic Computer-Assisted Patient-Specific Preoperative Planning for Orthopedic Fractures Surgery.” In: *International Journal of Computer Assisted Radiology and Surgery* 10.10 (Oct. 2015), pp. 1535–1546. ISSN: 1861-6429. DOI: 10.1007/s11548-015-1162-9.

[15] G. Turk and M. Levoy. “Zippered polygon meshes from range images.” In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, pp. 311–318. DOI: 10.1145/192161.192241.

[16] Y. Chen and G. Medioni. “Object modeling by registration of multiple range images.” In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. Vol. 3. Apr. 1991, pp. 2724–2729. DOI: 10.1109/ROBOT.1991.132043.

[17] L. Ikemoto, N. Gelfand, and M. Levoy. “A hierarchical method for aligning warped meshes.” In: *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings*. Oct. 2003, pp. 434–441. DOI: 10.1109/IM.2003.1240279.

[18] K. Pulli. “Multiview registration for large data sets.” In: *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*. Ottawa, Canada, Oct. 1999, pp. 160–168. ISBN: 0-7695-0062-5. DOI: 10.1109/IM.1999.805346.

[19] T.-g. Son, J. Lee, J. Lim, and K. Lee. “Reassembly of Fractured Objects using Surface Signature.” In: *The Visual Computer* (2017), pp. 1–11. DOI: 10.1007/s00371-017-1419-0.

[20] F. Cohen, Z. Liu, and T. Ezgi. “Virtual reconstruction of archeological vessels using expert priors and intrinsic differential geometry information.” In: *Computers & Graphics* 37.1-2 (2013), pp. 41–53. DOI: 10.1016/j.cag.2012.11.001.

[21] Q.-X. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. “Reassembling fractured objects by geometric matching.” In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 569–578. DOI: 10.1145/1179352.1141925.

[22] N. Mellado, P. Reuter, and C. Schlick. “Semi-automatic geometry-driven reassembly of fractured archeological objects.” In: *VAST 2010: The 11th international symposium on virtual reality, archaeology and cultural heritage*. 2010, p. 00.

[23] A. Willis, D. Anderson, T. Thomas, T. Brown, and J. L. Marsh. “3D Reconstruction of Highly Fragmented Bone Fractures.” In: vol. 6512. Mar. 2007. DOI: 10.1117/12.708683.

[24] M. H. Moghari and P. Abolmaesumi. “Global registration of multiple bone fragments using statistical atlas models: Feasibility experiments.” In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Aug. 2008, pp. 5374–5377. DOI: 10.1109/IEMBS.2008.4650429.

[25] K. Wu, K. Wong, S. Ng, S. Quek, B. Zhou, D. Murphy, Z. Daruwalla, and H. Ren. “Statistical atlas-based morphological variation analysis of the asian humerus: Towards consistent allometric implant positioning.” In: *International Journal of Computer Assisted Radiology and Surgery* 10.3 (Mar. 2015), pp. 317–327. ISSN: 1861-6429. DOI: 10.1007/s11548-014-1084-y.

[26] T. Albrecht and T. Vetter. “Automatic Fracture Reduction.” In: *Mesh Processing in Medical Image Analysis 2012: MICCAI 2012 International Workshop, MeshMed 2012, Nice, France, October 1, 2012. Proceedings.* Ed. by J. A. Levine, R. R. Paulsen, and Y. Zhang. Nice, France: Springer Berlin Heidelberg, Oct. 2012, pp. 22–29. ISBN: 978-3-642-33463-4. DOI: 10.1007/978-3-642-33463-4_3.

[27] K. Oura, Y. Otake, A. Shigi, F. Yokota, T. Murase, and Y. Sato. “Prediction of forearm bone shape based on partial least squares regression from partial shape.” In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 13.3 (2017), e1807. DOI: 10.1002/rcs.1807.

[28] T. Okada, Y. Iwasaki, T. Koyama, N. Sugano, Y.-W. Chen, K. Yonenobu, and Y. Sato. “Computer-Assisted Preoperative Planning for Reduction of Proximal Femoral Fracture Using 3-D-CT Data.” In: *IEEE Transactions on Biomedical Engineering* 56.3 (Mar. 2009), pp. 749–759. ISSN: 0018-9294. DOI: 10.1109/TBME.2008.2005970.

[29] P. Fürnstahl, G. Székely, C. Gerber, J. Hodler, J. G. Snedeker, and M. Harders. “Computer assisted reconstruction of complex proximal humerus fractures for preoperative planning.” In: *Medical Image Analysis* 16.3 (2012). Computer Assisted Interventions, pp. 704–720. ISSN: 1361-8415. DOI: 10.1016/j.media.2010.07.012.

[30] P. W. ten Berg, J. G. Dobbe, S. D. Strackee, and G. J. Streekstra. “Three-Dimensional Assessment of Bilateral Symmetry of the Scaphoid: An Anatomic Study.” In: *BioMed research international* 2015.2015 (2015). DOI: 10.1155/2015/547250.

[31] J. A. DeLude, R. T. Bicknell, G. A. MacKenzie, L. M. Ferreira, C. E. Dunning, G. J. King, J. A. Johnson, and D. S. Drosdowech. “An anthropometric study of the bilateral anatomy of the humerus.” In: *Journal of Shoulder and Elbow Surgery* 16.4 (2007), pp. 477–483. ISSN: 1058-2746. DOI: https://doi.org/10.1016/j.jse.2006.09.016.

[32] L. Vlachopoulos, G. Székely, C. Gerber, and P. Fürnstahl. “A scale-space curvature matching algorithm for the reconstruction of complex proximal humeral fractures.” In: *Medical Image Analysis* 43 (2018), pp. 142–156. ISSN: 1361-8415. DOI: 10.1016/j.media.2017.10.006.

[33] B. M. Auerbach and C. B. Ruff. “Limb bone bilateral asymmetry: variability and commonality among modern humans.” In: *Journal of Human Evolution* 50.2 (2006), pp. 203–218. ISSN: 0047-2484. DOI: 10.1016/j.jhevol.2005.09.004.

[34] J. Vroemen, J. Dobbe, R. Jonges, S. Strackee, and G. Streekstra. “Three-Dimensional Assessment of Bilateral Symmetry of the Radius and Ulna for Planning Corrective Surgeries.” In: *The Journal of Hand Surgery* 37.5 (2012), pp. 982–988. ISSN: 0363-5023. DOI: 10.1016/j.jhsa.2011.12.035.

[35] J. Buschbaum, R. Fremd, T. Pohlemann, and A. Kristen. "Computer-assisted fracture reduction: a new approach for repositioning femoral fractures and planning reduction paths." In: *International journal of computer assisted radiology and surgery* 10.2 (2015), pp. 149–159. DOI: 10.1007/s11548-014-1011-2.

[36] S. Winkelbach, M. Rilk, C. Schönfelder, and F. M. Wahl. "Fast Random Sample Matching of 3D Fragments." In: *Joint Pattern Recognition Symposium*. Springer. 2004, pp. 129–136. DOI: 10.1007/978-3-540-28649-3_16.

[37] Q. Li, M. Zhou, and G. Geng. "Fracture Surfaces Matching for Reassembling Broken Solids." In: *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE* 9.16 (2012), pp. 4847–4855.

[38] H. Pottmann, S. Leopoldseder, and M. Hofer. "Registration without ICP." In: *Computer Vision and Image Understanding* 95.1 (2004), pp. 54–71. DOI: 10.1016/j.cviu.2004.04.002.

[39] B. Zhou, A. Willis, Y. Sui, D. Anderson, T. Thomas, and T. Brown. "Improving inter-fragmentary alignment for virtual 3D reconstruction of highly fragmented bone fractures." In: *Medical Imaging 2009: Image Processing*. Vol. 7259. International Society for Optics and Photonics. 2009, pp. 7259-1–7259-9. DOI: 10.1117/12.810967.

[40] T. P. Thomas. "Virtual pre-operative reconstruction planning for comminuted articular fractures." PhD thesis. University of Iowa, 2010.

[41] A. S. Chowdhury, S. M. Bhandarkar, R. W. Robinson, and C. Y. Jack. "Virtual Craniofacial Reconstruction Using Computer Vision, Graph Theory and Geometric Constraints." In: *Pattern Recognition Letters* 30.10 (2009), pp. 931–938. DOI: 10.1016/j.patrec.2009.03.010.

[42] B. Zhou, A. Willis, Y. Sui, D. Anderson, T. Brown, and T. Thomas. "Virtual 3D bone fracture reconstruction via inter-fragmentary surface alignment." In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. Sept. 2009, pp. 1809–1816. DOI: 10.1109/ICCVW.2009.5457502.

[43] P. J. Besl and N. D. McKay. "A Method for Registration of 3-D Shapes." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[44] S. Rusinkiewicz and M. Levoy. "Efficient variants of the ICP algorithm." In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

[45] Unity Technologies. *Unity*. Version 2017.1.1.f1 personal. July 5, 2018. URL: https://unity3d.com/.

[46] K. Arun, T. Huang, and S. Blostein. "Least-Squares Fitting of Two 3-D Point Sets." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9.5 (Sept. 1987), pp. 698–700. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1987.4767965.

[47] B. K. Horn. "Closed-form solution of absolute orientation using unit quaternions." In: *JOSA A* 4.4 (1987), pp. 629–642. DOI: 10.1364/JOSAA.4.000629.

[48] B. K. Horn, H. M. Hilden, and S. Negahdaripour. “Closed-form solution of absolute orientation using orthonormal matrices.” In: *JOSA A* 5.7 (1988), pp. 1127–1135. DOI: 10.1364/JOSAA.5.001127.

[49] M. W. Walker, L. Shao, and R. A. Volz. “Estimating 3-D location parameters using dual number quaternions.” In: *CVGIP: image understanding* 54.3 (1991), pp. 358–367. DOI: 10.1016/1049-9660(91)90036-O.

[50] D. W. Eggert, A. Lorusso, and R. B. Fisher. “Estimating 3-D rigid body transformations: a comparison of four major algorithms.” In: *Machine vision and applications* 9.5-6 (1997), pp. 272–290. DOI: 10.1007/s001380050048.

[51] A. Cauchy. *Méthode générale pour la résolution des systemes déquations simultanées*. 1847, pp. 536–538.

[52] M. D. Wheeler and K. Ikeuchi. “Iterative Estimation of Rotation and Translation using the Quaternion.” Dec. 1995.

[53] K.-L. Low. *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. Tech. rep. University of North Carolina at Chapel Hill, Feb. 2004.

[54] F. Tampieri. “Newell’s Method for Computing the Plane Equation of a Polygon.” In: *Graphics Gems III (IBM Version)*. 1992. DOI: 10.1016/B978-0-08-050755-2.50052-X.

[55] N. Gelfand, S. Rusinkiewicz, L. Ikemoto, and M. Levoy. “Geometrically stable sampling for the ICP algorithm.” In: *null*. IEEE. 2003, p. 260. DOI: 10.1109/IM.2003.1240258.

[56] A. Kronman and L. Joskowicz. “Automatic Bone Fracture Reduction by Fracture Contact Surface Identification and Registration.” In: *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*. IEEE. 2013, pp. 246–249. DOI: 10.1109/ISBI.2013.6556458.

[57] A. Meesters. “Three-Dimensional Computed Tomography Measurements of Acetabular Fractures.” MA thesis. University of Twente, 2018.

[58] A. Meesters, J. Kraeima, H. Banierink, C. Slump, K. Ten Duis, W. M.J.H., and I. F.F.A. “Three-Dimensional Computed Tomography Measurements of Acetabular Fractures.” 2018.

[59] Materialise NV. *Mimics Medical*. Version 19.0. URL: https://www.materialise.com/en/medical/software/mimics.

[60] Materialise NV. *Materialise 3-matic*. Version 11.0. URL: https://www.materialise.com/en/software/3-matic.

[61] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. “MeshLab: an Open-Source Mesh Processing Tool.” In: *Eurographics Italian Chapter Conference*. Ed. by V. Scarano, R. D. Chiara, and U. Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[62] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. “Comparing images using the Hausdorff distance.” In: *IEEE Transactions on pattern analysis and machine intelligence* 15.9 (1993), pp. 850–863. DOI: 10.1109/34.232073.

[63] P. Cignoni, C. Rocchini, and R. Scopigno. “Metro: measuring error on simplified surfaces.” In: *Computer Graphics Forum*. Vol. 17. 2. Blackwell Publishers. 1998, pp. 167–174.

[64] Y. Hu, H. Li, G. Qiao, H. Liu, A. Ji, and F. Ye. “Computer-assisted virtual surgical procedure for acetabular fractures based on real CT data.” In: *Injury* 42.10 (2011), pp. 1121–1124. doi: 10.1016/j.injury.2011.01.014.

Part II

Supporting Material

Chapter 1

Virtual Fracture Reduction Pipeline

The treatment of fractures is a complex task. For simple fractures, e.g. the spiral fracture shown in Figure 1.1(a), an X-ray image is usually sufficient to properly plan the surgery, if surgery is needed. Comminuted fractures, illustrated in Figure 1.1(b), usually result in small fragments and possibly deformed bones. These kinds of fractures require scan techniques to obtain 3D models that allow the surgeon to review both the osseous and soft tissue structures [1].

Jiménez-Delgado et al. [1] identify three main steps during the planning of orthopedic reconstruction. (i) First, one has to obtain an anatomic reconstruction of the bones and fracture fragments from the 3D scans; see for example the models shown in Figure 1.2. (ii) The next step is concerned with the fixation and stabilization of the fragments to facilitate early recovery of mobility. (iii) During the last step of surgery preparation, the virtual fracture reduction is analyzed under common bio-mechanical conditions. Next to preoperative planning, reduced fractures have also been used in the construction of surgical guides [2–4], the design of patient-specific fixation devices [3, 5], maxillofacial reconstruction [6–8], and the evaluation of a surgeon's performance on a simulated patient-specific surgery [9].

Currently all three steps are performed manually with the aid of 3D medical image processing software. First, the bone tissue is segmented by thresholding on some value of the density measured in the computed tomography (CT) scan. After this, loose voxels are removed [10]. Fragments are obtained by manually

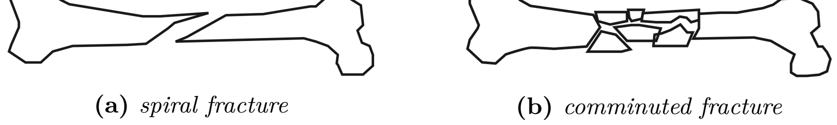

(a) *spiral fracture* **(b)** *comminuted fracture*

Figure 1.1: *(a) A fracture with a single fracture line, and (b) a comminuted fracture. Images adapted from Jiménez-Delgado et al. [1].*

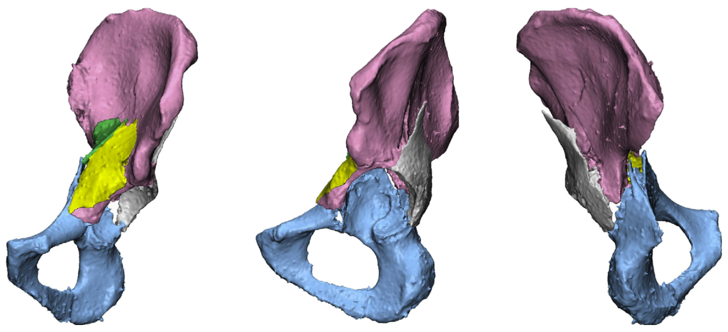


Figure 1.2: *Different views of a 3D reconstruction of a left acetabular fracture before virtual reduction. Image adapted from Merema et al. [3].*

splitting the models, with the aid of tools provided by medical image processing software. The individual fragments are reviewed by a professional who manually adjusts them where necessary and labels them [3]. During step (ii), the fracture is reduced by a professional who manually rotates and translates the fragments. The complexity of acetabular fractures makes their manual reduction particularly difficult [1]. Therefore, the mirrored contralateral bone is often used as a template [3, 10, 11]. We do not consider step (iii), the analysis of the fracture reduction, in this discussion.

Step (ii) is the most time-consuming and challenging aspect of the preoperative planning process [12]. Therefore, we propose to automate steps (i) and (ii); i.e. the processing of CT scans to a reduced fracture, with the pipeline shown in Figure 1.3. The first two stages in Figure 1.3 are concerned with obtaining the 3D models to represent the bone fragments from the medical images, i.e. step (i). These stages are discussed in Sections 1.1 and 1.2, respectively. The virtual reduction stage corresponds to step (ii). This stage is concerned with the translation and rotation of the bone fragment into an anatomically correct configuration and is discussed in Section 1.3.

1.1 Segmentation

Methods to segment CT scans can be classified as being based on either intensities, statistical shape models (SSMs), or atlases. The first method has difficulty handling the joint region, since the cortical bone in that area is thin. This causes gray level inhomogeneities and diffused boundaries [1, 13]. This issue is illustrated for the tibia in Figure 1.4. Generally, global thresholding on intensity values is done with the marching cubes algorithm [14]. However, it is difficult to find a good global threshold to use due to variation in intensity levels between CT scan slices [1]. This issue is often addressed by using an interactive approach, which is time-consuming. For example, Tomazevic et al. [15] let the user apply merging, hole-filling, and separation, tools to generate individually segmented fragments from the result of global thresholding. A less time-consuming approach proposed by Paulano et al. [16] uses 2D region growing. This algorithm is initiated by the user who places seed points inside regions

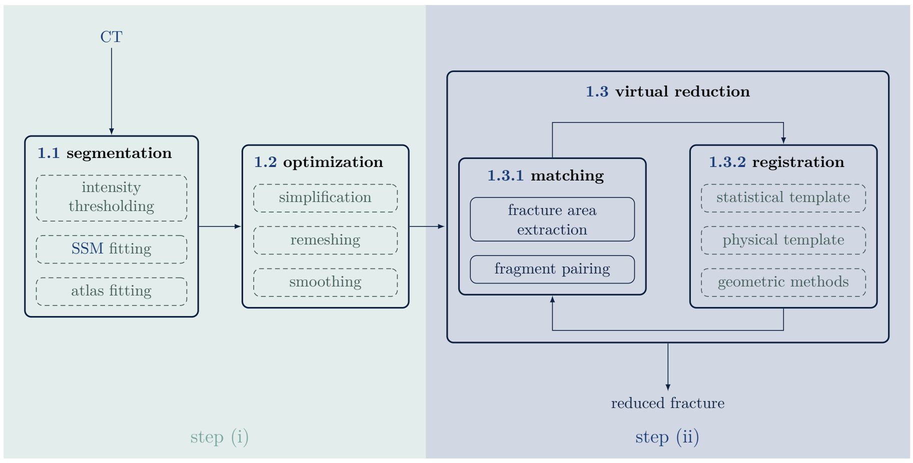


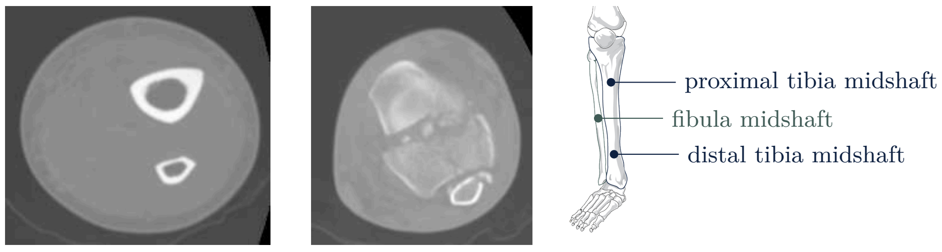
Figure 1.3: *A diagram representing the fracture reduction pipeline. Optional operations are indicated by shapes with a dashed border. The numbers of the sections corresponding to the different stages are included in the figure. The left and right colored blocks indicate stages that are part of step (i) and (ii) of the preoperative planning process, respectively.*

of target bone fragments. After this, a 2D region algorithm is executed for each seed point, and all seeds are propagated through the image stack [1].

Statistical shape model (SSM) based methods fit an SSM to an image to segment it. For these methods to work, the training population used to build the SSM should be representative of the patient whose bone is segmented. Contrary to approaches that use an SSM, atlas based methods employ prior knowledge about shape and intensity distributions [1]. For example, Pettersson et al. [9] use a non-rigid registration algorithm to automatically segment CT scans by iteratively deforming a template to match the patient's image stack. The atlas is generated from scans of a non-fractured bone, where each voxel is labeled as belonging to either the background, the bone surface or the bone interior.

1.2 Optimization

Optimization processes the meshes of bone fragments to obtain desirable properties for the subsequent simulation stage. Which properties are desirable depends on the intended application of the meshes. Based on the work by Jiménez-Delgado et al. [1], we distinguish three groups of non-mutually exclusive optimization techniques. In general, optimization techniques enhance the features of the model to enable the user to interact with it, and to improve its manageability and visualization. Enhancement before user interaction is needed, since models extracted from medical images are generally complex and have a very large number of faces. Simplification aims to reduce the complexity of



(a) *proximal midshaft* **(b)** *distal midshaft* **(c)** *tibia and fibula*

Figure 1.4: *The influence of the thickness of the cortical bone tissue on the CT scan, illustrated with transverse CT scans of the (a) proximal and (b) distal midshaft of (c) the tibia and fibula. The (a) proximal midshaft has a high intensity circular region whereas (b) the more distal scan shows far less contrast. Figures 1.4(a) and 1.4(b) were taken from Willis et al. [17], Figure 1.4(c) was adapted from Mariana Ruiz Villarreal [18].*

the models, to avoid overwhelming the computer used to examine the files with high memory requirements. Oversimplification of models should be avoided to ensure the fragment's original features are retained. Especially the fracture area and the regions around the joints are sensitive to oversimplification.

Remeshing is intended to improve the quality of the 3D mesh representing the fragment. This optimization technique is mostly used to enhance the visualization of the fragment. Meshes computed from medical volume images often show irregularities in the size and shape of geometrical primitives. These irregularities are often caused by the gaps between consecutive slices. By enhancing model properties such as sampling density or the regularity, size, or orientation of the primitives, the quality of the model and its visualization can be improved. It is difficult to apply remeshing near joints and the fracture area due to the heterogeneity of the cancellous tissue in those regions.

Smoothing techniques aim to improve the visualization of the models by extrapolating absent information from between the slices and by removing noise form the areas surrounding the bone. These techniques are particularly useful when the distance between slices is large, or when low resolution images are used. A disadvantage of smoothing is that it can remove small features from the fracture fragments together with the noise [1].

1.3 Virtual Reduction

Jiménez-Delgado et al. [1] define virtual reduction as the process by which fragments are relocated, with the aim of placing them back into their anatomical alignment. The exact problem that this procedure solves depends on the kind of fracture. For a simple fracture, such as the one in Figure 1.1(a), reduction is as simple as aligning the two bone fragments. For comminuted fractures the process is more akin to solving a complex three-dimensional puzzle. If multiple fragments are involved, one first has to match fracture areas together before they can be registered. Since the registration of each pair of fragments may influence the registration of other fragment pairs, the matching and registration processes feed back into each other.

1.3.1 Matching

The reduction of the fracture can be improved by only registering fracture areas to each other. Therefore, the first step of the matching procedure is to find these areas. Bone fragments are registered based on pairs of fracture areas.

Okada et al. [19] extract fracture lines interactively by letting users steer a 3D line tracking algorithm that operates on the 3D curvature images of the bone fragments. A fully automatic approach is proposed by Willis et al. [17] who use the intensity values of the CT scan to distinguish cortical and cancellous bone. This method is sensitive to thin cortical areas, similar to segmentation methods based on the thresholding of intensities. A combination of these two approaches is presented by Kovler et al. [12] who use both the intensity values and the the curvature of the fragment's surface to identify fracture areas. An alternative automatic approach is introduced by Fürnstahl et al. [20] who find fracture surfaces by comparing vertex normal directions. However, this method only works for reductions where the fracture surfaces are narrow. Therefore, this method is unlikely to work for acetabular fractures.

After the identification of the fracture areas, they are be paired, and the order in which they will be registered is determined. Kovler et al. [12] and Willis et al. [17] use an interactive approach and let the user identify fracture area pairs. Mangs [21] propose a relatively simple hierarchical matching based on a static fragment and the number of matching points each non-static fragment has to this fragment. Chowdhury et al. [22] use a Maximum Weight Graph Matching algorithm to identify matching fracture surfaces. In this graph fracture areas are represented by vertices. The weight of an edge between two vertices expresses their matching score. A fracture area pair has a high score if they are spatially proximal and if they exhibit complementary fracture surface characteristics.

1.3.2 Registration[1]

The literature identifies three different approaches to the registration problem, namely methods that register based on a statistical template, a physical template or, the geometry of the fragments.

Statistical approaches register the fragments of the broken bone to an anatomical atlas of a healthy bone. Moghari and Abolmaesumi [23] register fragments of a fractured humerus to an anatomical atlas of that bone in a local registration step, after which a global registration step is used to register the fragments to each other, and the template. Albrecht and Vetter [24] propose to use an SSM only if a contralateral or pre-fracture scan is not available. Their iterative method alternates between aligning the main fracture fragments to the

[1] This section is a summary of the discussion of different registration methods and how their performance compares in Section 2 in Part I.

template with iterative closest point (ICP), and adapting the statistical shape model to the individual's anatomy. An alternative to the anatomical atlases used by Moghari and Abolmaesumi [23] and Albrecht and Vetter [24] is offered by Oura et al. [25]. They predict the shape of the whole bone from its partial shape using partial least squares regression. The accuracy of the resulting prediction is similar to that of the methods based on bilateral symmetry. In general statistical methods have difficulty handling non-standard bones. Furthermore, most statistical approaches require an atlas that is representative of the patient, which may not be available.

Physical methods use a physical template to register a fracture, for example the mirrored contralateral bone or a pre-existing scan of the healthy bone. The first approach is compared to other registration methods by Okada et al. [19]. They find that this method performs worst w.r.t. the rotation error. Fürnstahl et al. [20] also used the contralateral bone, and achieved better results than Okada et al. [19], at the expense of a significantly higher runtime and more user interaction. Both of these methods assume bilateral symmetry, which is not necessarily the case. Vlachopoulos et al. [26] caution against blindly using the contralateral bone as a template for surgical reconstruction. Another disadvantage of registering to the contralateral bone is that it requires an additional CT scan of the uninjured bone, which increases the patient's radiation exposure [20].

Methods that register bone fragments based on the geometric properties of the 3D models representing them are called geometric methods. Most of these approaches use some variant of the ICP algorithm[2] [12, 17, 19, 20, 22, 27, 28]. This algorithm iteratively refines the transformation that aligns one point cloud to another by minimizing an error metric, generally some distance measure between matched pairs of points from the two point clouds. It is guaranteed to always monotonically converge to the nearest local minimum [29]. Consequently it is sensitive to the initial alignment of the fragments. A geometry based method that does not use ICP is introduced by Buschbaum et al. [30]. They register fragments to each other based on fracture lines; the lines separating the bone's strongly curved surface areas from its smooth surfaces.

References

[1] J. J. Jiménez-Delgado, F. Paulano-Godino, R. PulidoRam-Ramírez, and J. R. Jiménez-Pérez. "Computer assisted preoperative planning of bone fracture reduction: Simulation techniques and new trends." In: *Medical Image Analysis* 30 (2016), pp. 30–45. ISSN: 1361-8415. DOI: 10.1016/j.media.2015.12.005.

[2] Z. Yaniv. "Registration for orthopaedic interventions." In: *Computational Radiology for Orthopaedic Interventions.* Springer, 2016, pp. 41–70.

[3] B. J. Merema, J. Kraeima, K. Ten Duis, K. Wendt, R. Warta, E. Vos, R. Schepers, M. Witjes, and F. IJpma. "The design, production and clinical application of 3D patient-specific implants with drilling guides for acetabular surgery." In: *Injury* 48.11 (2017), pp. 2540–2547.

[2] An extensive discussion of the different aspects of this algorithm can be found in Section 3 in Part I.

[4] R. H. Schepers, G. M. Raghoebar, A. Vissink, M. W. Stenekes, J. Kraeima, J. L. Roodenburg, H. Reintsema, and M. J. Witjes. "Accuracy of fibula reconstruction using patient-specific CAD/CAM reconstruction plates and dental implants: a new modality for functional reconstruction of mandibular defects." In: *Journal of Cranio-Maxillofacial Surgery* 43.5 (2015), pp. 649–657. DOI: 10.1016/j.jcms.2015.03.015.

[5] I. Otomaru, M. Nakamoto, Y. Kagiyama, M. Takao, N. Sugano, N. Tomiyama, Y. Tada, and Y. Sato. "Automated preoperative planning of femoral stem in total hip arthroplasty from 3D CT data: Atlas-based approach and comparative study." In: *Medical image analysis* 16.2 (2012), pp. 415–426. DOI: 10.1016/j.media.2011.10.005.

[6] T. Woo, J. Kraeima, Y. O. Kim, Y. S. Kim, T. S. Roh, D. H. Lew, and I. S. Yun. "Mandible reconstruction with 3D virtual planning." In: *J Int Soc Simul Surg* 2.2 (2015), pp. 90–93.

[7] R. H. Schepers, G. M. Raghoebar, A. Vissink, L. U. Lahoda, W. J. Van der Meer, J. L. Roodenburg, H. Reintsema, and M. J. Witjes. "Fully 3-dimensional digitally planned reconstruction of a mandible with a free vascularized fibula and immediate placement of an implant-supported prosthetic construction." In: *Head & neck* 35.4 (2013), E109–E114. DOI: 10.1002/hed.21922.

[8] R. H. Schepers, J. Kraeima, A. Vissink, L. U. Lahoda, J. L. Roodenburg, H. Reintsema, G. M. Raghoebar, and M. J. Witjes. "Accuracy of secondary maxillofacial reconstruction with prefabricated fibula grafts using 3D planning and guided reconstruction." In: *Journal of Cranio-Maxillofacial Surgery* 44.4 (2016), pp. 392–399. DOI: 10.1016/j.jcms.2015.12.008.

[9] J. Pettersson, K. L. Palmerius, H. Knutsson, O. Wahlstrom, B. Tillander, and M. Borga. "Simulation of patient specific cervical hip fracture surgery with a volume haptic interface." In: *IEEE Transactions on Biomedical Engineering* 55.4 (2008), pp. 1255–1265. DOI: 10.1109/TBME.2007.908099.

[10] A. Meesters. "Three-Dimensional Computed Tomography Measurements of Acetabular Fractures." MA thesis. University of Twente, 2018.

[11] P. Fürnstahl. "Computer-assisted planning for orthopedic surgery." PhD thesis. ETH Zurich, 2010. DOI: 10.3929/ethz-a-006198365.

[12] I. Kovler, L. Joskowicz, Y. A. Weil, A. Khoury, A. Kronman, R. Mosheiff, M. Liebergall, and J. Salavarrieta. "Haptic Computer-Assisted Patient-Specific Preoperative Planning for Orthopedic Fractures Surgery." In: *International Journal of Computer Assisted Radiology and Surgery* 10.10 (Oct. 2015), pp. 1535–1546. ISSN: 1861-6429. DOI: 10.1007/s11548-015-1162-9.

[13] C. Chu, C. Chen, L. Liu, and G. Zheng. "Facts: fully automatic ct segmentation of a hip joint." In: *Annals of biomedical engineering* 43.5 (2015), pp. 1247–1259. DOI: 10.1007/s10439-014-1176-4.

[14] W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169. DOI: 10.1145/37402.37422.

[15] M. Tomazevic, D. Kreuh, A. Kristan, V. Puketa, and M. Cimerman. "Preoperative planning program tool in treatment of articular fractures: process of segmentation procedure." In: *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010*. Springer. 2010, pp. 430–433. DOI: 10.1007/978-3-642-13039-7_108.

[16] F. Paulano, J. J. Jiménez, and R. Pulido. "3D segmentation and labeling of fractured bone from CT images." In: *The Visual Computer* 30.6-8 (2014), pp. 939–948. DOI: 10.1007/s00371-014-0963-0.

[17] A. Willis, D. Anderson, T. Thomas, T. Brown, and J. L. Marsh. "3D Reconstruction of Highly Fragmented Bone Fractures." In: vol. 6512. Mar. 2007. DOI: 10.1117/12.708683.

[18] Mariana Ruiz Villarreal. *Human leg bones labeled*. Online; accessed August 14, 2018. 2018.

[19] T. Okada, Y. Iwasaki, T. Koyama, N. Sugano, Y.-W. Chen, K. Yonenobu, and Y. Sato. "Computer-Assisted Preoperative Planning for Reduction of Proximal Femoral Fracture Using 3-D-CT Data." In: *IEEE Transactions on Biomedical Engineering* 56.3 (Mar. 2009), pp. 749–759. ISSN: 0018-9294. DOI: 10.1109/TBME.2008.2005970.

[20] P. Fürnstahl, G. Székely, C. Gerber, J. Hodler, J. G. Snedeker, and M. Harders. "Computer assisted reconstruction of complex proximal humerus fractures for preoperative planning." In: *Medical Image Analysis* 16.3 (2012). Computer Assisted Interventions, pp. 704–720. ISSN: 1361-8415. DOI: 10.1016/j.media.2010.07.012.

[21] L. Mangs. "Computer-assisted fracture reduction in an orthopaedic preoperative planning workflow." MA thesis. Linköping University, 2017.

[22] A. S. Chowdhury, S. M. Bhandarkar, R. W. Robinson, and C. Y. Jack. "Virtual Craniofacial Reconstruction Using Computer Vision, Graph Theory and Geometric Constraints." In: *Pattern Recognition Letters* 30.10 (2009), pp. 931–938. DOI: 10.1016/j.patrec.2009.03.010.

[23] M. H. Moghari and P. Abolmaesumi. "Global registration of multiple bone fragments using statistical atlas models: Feasibility experiments." In: *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Aug. 2008, pp. 5374–5377. DOI: 10.1109/IEMBS.2008.4650429.

[24] T. Albrecht and T. Vetter. "Automatic Fracture Reduction." In: *Mesh Processing in Medical Image Analysis 2012: MICCAI 2012 International Workshop, MeshMed 2012, Nice, France, October 1, 2012. Proceedings*. Ed. by J. A. Levine, R. R. Paulsen, and Y. Zhang. Nice, France: Springer Berlin Heidelberg, Oct. 2012, pp. 22–29. ISBN: 978-3-642-33463-4. DOI: 10.1007/978-3-642-33463-4_3.

[25] K. Oura, Y. Otake, A. Shigi, F. Yokota, T. Murase, and Y. Sato. "Prediction of forearm bone shape based on partial least squares regression from partial shape." In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 13.3 (2017), e1807. DOI: 10.1002/rcs.1807.

[26] L. Vlachopoulos, G. Székely, C. Gerber, and P. Fürnstahl. "A scale-space curvature matching algorithm for the reconstruction of complex proximal humeral fractures." In: *Medical Image Analysis* 43 (2018), pp. 142–156. ISSN: 1361-8415. DOI: `10.1016/j.media.2017.10.006`.

[27] B. Zhou, A. Willis, Y. Sui, D. Anderson, T. Thomas, and T. Brown. "Improving inter-fragmentary alignment for virtual 3D reconstruction of highly fragmented bone fractures." In: *Medical Imaging 2009: Image Processing*. Vol. 7259. International Society for Optics and Photonics. 2009, pp. 7259-1–7259-9. DOI: `10.1117/12.810967`.

[28] T. P. Thomas. "Virtual pre-operative reconstruction planning for comminuted articular fractures." PhD thesis. University of Iowa, 2010.

[29] H. Pottmann, S. Leopoldseder, and M. Hofer. "Registration without ICP." In: *Computer Vision and Image Understanding* 95.1 (2004), pp. 54–71. DOI: `10.1016/j.cviu.2004.04.002`.

[30] J. Buschbaum, R. Fremd, T. Pohlemann, and A. Kristen. "Computer-assisted fracture reduction: a new approach for repositioning femoral fractures and planning reduction paths." In: *International journal of computer assisted radiology and surgery* 10.2 (2015), pp. 149–159. DOI: `10.1007/s11548-014-1011-2`.

Chapter 2

User Interface

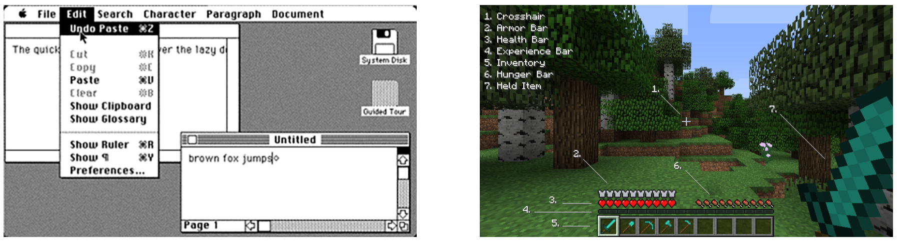
Interactions between a user and a three-dimensional environment can be distinguished into four universal tasks: system control, viewpoint control, object selection, and object manipulation [1]. System control is concerned with the interface that supports interaction between the user and the system that is not part of the virtual environment, this aspect of user interfaces (UIs) is considered in Section 2.1. The task viewpoint control, which focuses on getting around in a virtual environment while keeping track of one's whereabouts and task objectives, is the subject of Section 2.2. Section 2.3 is concerned with the first step of object manipulation: object selection. Different approaches to manipulating the selected object are discussed in Section 2.4.

2.1 System Control

Good application control techniques should be easy to learn for novices and efficient for experts. Furthermore they should allow novice users to gradually learn new ways of interacting with the system [2]. Interfaces using windows, icons, menus, and a pointer (WIMP) satisfy all these requirements [1]. An example of such an interface is shown in Figure 2.1(a).

The computer and video game industry has adapted WIMP interfaces into head-up display (HUD) interfaces for use with 3D interactive graphics. This approach places application control components in a screen space on a 2D plane, the HUD, that is displayed next to or over the 3D scene. Figure 2.1(b) shows the HUD used by the popular computer game Minecraft.

This approach uses two different user interface (UI) metaphors, namely 3D manipulation for navigation and manipulation of 3D objects, and a conventional 2D graphical user interface (GUI) for the rest of the system's functionality [1]. A disadvantage of this approach is that it requires the user to switch between two very different interface metaphors. If the shift between the two is too great the user's engagement may be broken [3].



(a) *WIMP interface* **(b)** *HUD interface*

Figure 2.1: *(a) One of the first successful WIMP interfaces, and (b) a HUD interface. Figure 2.1(a) was adapted from Hicks [4], Figure 2.1(b) from HalfOfAKebab [5].*

We only consider user interface techniques that use a two-dimensional mouse and keyboard in combination with a 'normal' display, since that configuration is pervasive in hospitals. Below we shortly contrast these devices with some of the alternatives.

One of the disadvantages of using a 'normal' display is that it does not offer the near unlimited field of view of fully immersive environments, such as the ones used by for example Harders et al. [6], Forsslund et al. [7], and Fürnstahl [8]. Since our application is only concerned with the inspection and manipulation of a small number of three-dimensional objects, this should not impact performance. Furthermore, stereoscopic three-dimensional displays can cause problems such as eyestrain, headache, fatigue, disorientation and nausea in a large number of users [1]. Finally, the use of a HUD interface instead of an immersive environment allows the use of a two-dimensional mouse.

Besides its ubiquitousness and location persistence [9], a two-dimensional mouse has several additional advantages compared to input devices with higher degrees of freedom (DOF). Firstly, it allows easy integration of two and three-dimensional environments, which is useful if a WIMP interface is used, as these are generally two-dimensional [10]. Secondly, most 2D mice are comfortable to use and do not limit the user to any particular grip. Furthermore the natural mapping from the device to the cursor reduces the cognitive load of the user [11]. One of the challenges of using a mouse in three-dimensional environments is the mapping of two-dimensional input to transformations with up to six DOF, three for rotation, and three for translation.

Mapping two-dimensional input to the manipulation of three-dimensional objects is generally done with a widget: a "visible graphic representations of an operation on, or a state of, an object, that is displayed together with that object" [12]. Hinckley et al. [13] analyzed the usability of three-dimensional rotation techniques in an orientation matching task, comparing input devices with three DOF to mouse-controlled widgets. They found that the multidimensional input devices were faster than the virtual techniques, but not more accurate. Additionally, Bérard et al. [14] compared the accuracy of a mouse with that of several 3-DOF devices in a translation task and found the mouse to be more accurate. They also observed that the free-space devices induced stress. Teather and Stuerzlinger [15] attribute the difference in accuracy observed by Hinckley et al. [13] and Bérard et al. [14] to the fact that a human hand held in free space suffers from jitter. Contrary to these results, McMahan et al. [16] found that for tasks requiring both translation and rotation in an immersive virtual environment a mouse was less efficient than devices with six DOF.

2.2 Viewpoint Control

The metaphor chosen for the control of the viewpoint strongly influences the way users interact with the application [17, 18]. Ware and Osborne [17] compared three viewpoint control technique metaphors: 'flying vehicle', 'eyeball in hand', and 'world in hand' in exploration and navigation tasks in three extremely different toy environments. The first of the metaphors gives the user the impression of flying through the environment. If the 'eyeball in hand' metaphor is used, the user fully controls the position of the camera, whereas with the 'world in hand' the viewpoint is fixed and the user manipulates the scene. Ware and Osborne [17] found that the 'world in hand' metaphor works best for the manipulation of scenes with discrete objects that are small enough to be grasped in real life. Intensive semi-structured interviewing of the subjects also revealed that simultaneous translation and rotation were perceived to be difficult in this metaphor. These results are supported by Partala [18], who compared the 'world in hand' and 'eyeball in hand' metaphors in a rotation matching task. Their subjects almost unanimously chose 'world in hand' as the best metaphor for the rotation of a single three-dimensional object. Furthermore Partala [18] also found that the 'world in hand' metaphor resulted in significantly faster task completion than the 'eyeball in hand metaphor'.

2.3 Selection

In desktop virtual environments, an object is commonly selected by positioning the cursor over it and clicking [1]. To obtain the object underneath the cursor ray-casting or off-screen rendering is generally used. This method selects the first object, from the perspective of the user, intersected by the mouse ray, i.e. the ray from the eye point through the cursor. As an extension to this method Van Emmerik [19] proposes to allow the selection of all objects in the scene intersected by the mouse ray by letting the user click multiple times. However this proposal violates the design principle for manipulation interfaces that "only visible objects can be manipulated" [15]. Within the field of medical imaging Gallo et al. [9] use a depth-enhanced cursor that ensures that the pointer always binds to the visible surface of the 3D object under investigation, independent of the occlusion of that surface by other objects along the mouse ray.

Zhai et al. [20] propose the 'Silk Cursor', which replaces the cursor with an axis aligned box covered with silk. Consequently objects behind it are occluded twice as much as those within, and objects in front of it are not occluded at all. This volume cursor outperformed a three-dimensional point cursor in target selection [20]. Problems with this technique include selecting objects beyond the users' reach, choosing an appropriate cursor size, and selecting objects in a target dense environment [21, 22]. Finally, independent of the used cursor, selection performance is increased when visual feedback is provided during the selection process [23].

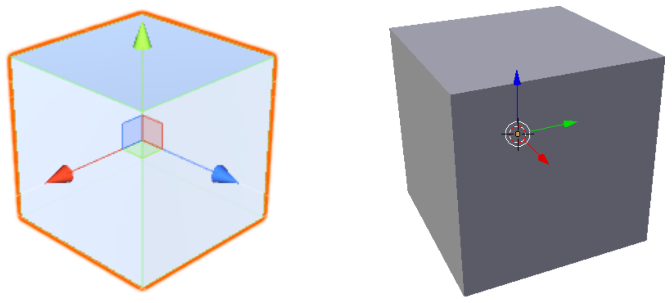
2.4 Manipulation

When transforming objects one can either translate, rotate or scale them. The first two operations are discussed in Sections 2.4.1 and 2.4.2, respectively. Scaling of individual fragments is not considered, as this operation is not available during surgery and should therefore not be possible during a virtual fracture reduction.

2.4.1 Translation

Most three-dimensional editing applications use widgets to manipulate objects [1, 24, 25]. Generally these widgets have handles, each of which lets the user control a single axis of movement. The most important advantage of this technique is that it always acts as the user expects it to [24]. However it also requires users to decompose three-dimensional movements into multiple one-dimensional translations. Furthermore it requires users to keep track of the current interaction state, which results in mode errors [24, 26]. Both Unity [27] and Blender [28] use this approach, as shown in Figure 2.2. Both applications use color to indicate the axis associated with a handle. Furthermore, Unity has extended the conventional translation widget to also allow movement along a plane, as indicated by the colored rectangles in Figure 2.2(a).

As an alternative to the conventional cursor, Houde [29] experimented with narrative cursors and bounding boxes with handles to let users move furniture around. However, this approach requires the objects to be moved to have an environment that is constraining by definition. The triad cursor introduced by Nielson and Olsen [30] replaces the conventional arrow with the axes of the local coordinate system, but still limits translations to a single dimension.



(a) *Unity* **(b)** *Blender*

Figure 2.2: *The translation widgets used by (a) Unity [27] and (b) Blender [28].*

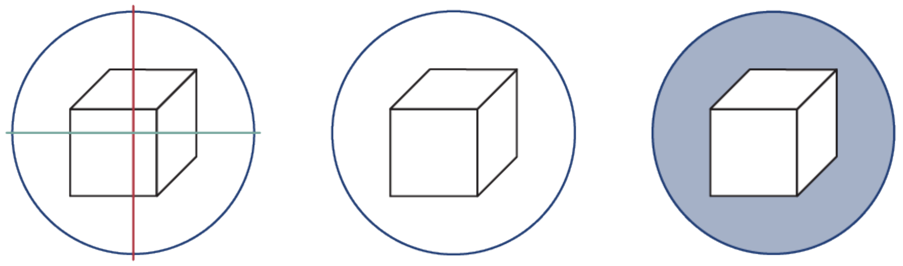
Chen et al. [31] note that users prefer translations to be constrained to a two-dimensional plane. One approach that constrains translations to a plane moves objects along the viewing plane. Although this technique is easy to implement the resulting movement is not intuitive and frequently misleading [24]. Alternatively, the movement plane can be selected to be the plane defined by the surface hit by the mouse ray when the user selected the object for selection. However, users perceive this approach as unpredictable, since they expect the translation to be independent of the cursor's position during selection [24]. A third approach is used by Unity, which allows the user to move the object within a single plane defined by two of the global or local axes [32], next to translations along a single axis.

Alternatively the movement plane can be determined using the idea that most objects attach to other objects. Oh and Stuerzlinger [24] present a sliding algorithm where the selected objects follow the cursor by sliding over the foremost surface behind it. If no such object is found it slides along the viewing plane. A comparison with traditional widgets shows that this approach enhances the subject's understanding of the object's three-dimensional position and increases the speed with which users complete a construction task. Sun et al. [33] generalized sliding by supporting object interpenetration and floating, in their shift-sliding algorithm. With this approach users may define a movement plane, to break the assumption of contact between objects. A comparison between this technique and conventional widgets revealed the former to be significantly faster.

2.4.2 Rotation

Three-dimensional modeling suites generally use one of three types of rotation widgets illustrated in Figure 2.3: discrete sliders, two-axis valuators, or virtual trackballs [34]. Chen et al. [31] observed that users prefer rotations to be constrained to a single dimension. Discrete sliders offer that constraint by providing separate elements for rotation along each of the axes of the object's local coordinate system. The sliders are presented either as separate elements [31], or projected on the object [34]. Two-axis valuators have two distinct behaviors. Moving the mouse inside the circle shown in Figure 2.3(b) rotates the object around the local x and y-axes. The object can be rotated around the z-axis by tracing an arc outside of the circle [34]. Virtual trackballs use the projection of the cursor's position onto an object-centered sphere to determine the intended rotation. Consequently, the rotation of the object mimics the result of nudging the sphere. The same technique as the one used by the two-axis valuator is used to control rotations around the z-axis [34].

In a comparison of these three types of rotation widgets in an orientation matching task Chen et al. [31] found that discrete sliders were superior in terms of accuracy and speed when rotating around a single axis. However, Rybicki et al. [34] observed that the discrete sliders were significantly less accurate and slower than the other two approaches for rotations around one or two axes. Unfortunately, Rybicki et al. [34] did not separate their results based on the number of axes around which the object was rotated. Both studies found discrete sliders to be the slowest option for rotations around three axes. Chen and Medioni [35] also observed that discrete sliders were more accurate than



(a) *discrete sliders* **(b)** *two-axis valuator* **(c)** *virtual trackball*

Figure 2.3: *Appearance of the rotation controllers generally used in three-dimensional modeling software: (a) discrete sliders, (b) two-axis valuator, and (c) trackball. Images adapted from Rybicki et al. [34].*

the other two approaches for complex rotations. In comparing these results it should be noted that their implementation of both the sliders and the virtual trackballs differ.

Chen et al. [31] found the virtual trackball to be superior to the two-axis valuator except for the speed of rotations around a single axis. In contrast, Bade et al. [36] observed the two-axis valuator to be significantly better than the virtual trackball in terms of speed in a search and shoot task. Zhao et al. [37] notes that in comparing these results we should keep in mind that the results of Bade et al. [36] are likely influenced by the fact that the two-axis valuator used a modifier key for rotation around the view vector, whereas the virtual trackball was solely mouse operated. Furthermore, it seems as though the experiments by Bade et al. [36] involved only two degrees of freedom as the final orientation was not taken into account [37]. Zhao et al. [37] did not observe differences between the two-axis valuator and the virtual trackball in terms of completion time and accuracy in an orientation matching task. However it should be noted that Zhao et al. [37] added the mouse wheel as a controller for rotation around the view vector.

Virtual trackballs are the most used three-dimensional rotation widget in commercial medical imaging software [9]. Different mappings from the cursor movement to rotation have been introduced by Chen et al. [31], Shoemake [38], and Bell [39]. The first mapping suffers from hysteresis [13, 38], i.e. closed loops of mouse motion do not necessarily result in closed rotation loops. Although Shoemake's mapping is a special case of the one by Chen et al., it does not suffer from hysteresis [40]. Hinckley et al. [13] found that in spite of their equal performance, users prefer Shoemake's mapping over the one defined by Chen et al. The same preference was observed by Shoemake [38] in an informal survey. Hinckley et al. [13] also found that the mapping defined by Bell results in a smoother rotation than Shoemake's. In spite of this neither Zhao et al. [37] nor Bade et al. [36] found a significant difference in performance between these two mappings. However Zhao et al. [37] rated the usability of Bell's mapping higher than Shoemake's, whereas Bade et al. [36] came to an inverse usability-based ordering. Zhao et al. [37] found Bell's mapping to be less predictable than Shoemake's. It should be noted that of all mentioned studies only Henriksen et al. [40] mention the size of the trackball. Since Shoemake's mapping is discontinuous when the cursor is moved outside of the trackball, the sphere's size is an important factor in the trackball's usability [40].

Gallo et al. [9] propose a virtual trackball that uses the geometry of the object itself as the rotation handle. In a comparison of this technique with a two-axis valuator the former was significantly faster. In spite of this, users perceived both techniques to be equally efficient, but considered the virtual trackball to be more accurate and easier to learn [9]. A possible disadvantage of this technique is that users are likely to click an arbitrary point on the object when selecting it for rotation, not realizing that the cursor position influences rotation.

References

[1] J. Jankowski and M. Hachet. "Advances in Interaction with 3D Environments." In: *Computer Graphics Forum* 34.1 (2015), pp. 152–190. ISSN: 1467-8659. DOI: 10.1111/cgf.12466.

[2] B. Shneiderman. "Direct manipulation: A step beyond programming languages." In: *Computer* 8 (1983), pp. 57–69. DOI: 10.1109/MC.1983.1654471.

[3] C. Hand. "A survey of 3D interaction techniques." In: *Computer graphics forum*. Vol. 16. 5. Wiley Online Library. 1997, pp. 269–281.

[4] J. Hicks. *40 years of icons: the evolution of the modern computer interface. Diary of a WIMP at middle age.* the verge. Mar. 2013. URL: https://www.theverge.com/2013/3/21/4127110/40-years-of-icons-the-evolution-of-the-modern-computer-interface (visited on 07/04/2018).

[5] HalfOfAKebab. *Heads-up display.* gamepedia. Nov. 2017. URL: https://minecraft.gamepedia.com/Heads-up_display (visited on 07/04/2018).

[6] M. Harders, A. Barlit, C. Gerber, J. Hodler, and G. Székely. "An optimized surgical planning environment for complex proximal humerus fractures." In: *MICCAI Workshop on Interaction in Medical Image Analysis and Visualization.* Vol. 10. Brisbane, Australia, Jan. 2007, pp. 201–206.

[7] J. Forsslund, S. Chan, K. J. Salisbury, R. G. Silva, S. Girod, and N. H. Blevins. "Design and implementation of a maxillofacial surgery rehearsal environment with haptic interaction for bone fragment and plate alignment." In: *Computer Assisted Radiology and Surgery, 2012.* Vol. 184. 2012.

[8] P. Fürnstahl. "Computer-assisted planning for orthopedic surgery." PhD thesis. ETH Zurich, 2010. DOI: 10.3929/ethz-a-006198365.

[9] L. Gallo, A. Minutolo, and G. D. Pietro. "A user interface for VR-ready 3D medical imaging by off-the-shelf input devices." In: *Computers in Biology and Medicine* 40.3 (2010), pp. 350–358. ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2010.01.006.

[10] D. Venolia. "Facile 3D Direct Manipulation." In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems.* CHI '93. Amsterdam, The Netherlands: ACM, 1993, pp. 31–36. ISBN: 0-89791-575-5. DOI: 10.1145/169059.169065.

[11] A. Ansari. “Direct 3D interaction using a 2D locator device.” MA thesis. University of South Florida, 2003.

[12] P. S. Strauss, P. Isaacs, and J. Schrag. *The Design and Implementation of Direction Manipulation in 3D*. 2002.

[13] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell. “Usability Analysis of 3D Rotation Techniques.” In: *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*. UIST ’97. Banff, Alberta, Canada: ACM, 1997, pp. 1–10. ISBN: 0-89791-881-9. DOI: 10.1145/263407.263408.

[14] F. Bérard, J. Ip, M. Benovoy, D. El-Shimy, J. R. Blum, and J. R. Cooperstock. “Did “Minority Report" get it wrong? Superiority of the mouse over 3D input devices in a 3D placement task.” In: *Human-Computer Interaction – INTERACT 2009: 12th IFIP TC 13 International Conference, Uppsala, Sweden, August 24-28, 2009, Proceedings, Part II*. Springer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 400–414. ISBN: 978-3-642-03658-3. DOI: 10.1007/978-3-642-03658-3_45.

[15] R. J. Teather and W. Stuerzlinger. “Guidelines for 3D Positioning Techniques.” In: *Proceedings of the 2007 Conference on Future Play*. Future Play ’07. Toronto, Canada: ACM, 2007, pp. 61–68. ISBN: 978-1-59593-943-2. DOI: 10.1145/1328202.1328214.

[16] R. P. McMahan, D. Gorton, J. Gresock, W. McConnell, and D. A. Bowman. “Separating the Effects of Level of Immersion and 3D Interaction Techniques.” In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST ’06. Limassol, Cyprus: ACM, 2006, pp. 108–111. ISBN: 1-59593-321-2. DOI: 10.1145/1180495.1180518.

[17] C. Ware and S. Osborne. “Exploration and Virtual Camera Control in Virtual Three Dimensional Environments.” In: *SIGGRAPH Comput. Graph.* 24.2 (Feb. 1990), pp. 175–183. ISSN: 0097-8930. DOI: 10.1145/91394.91442.

[18] T. Partala. “Controlling a Single 3D Object: Viewpoint Metaphors, Speed and Subjective Satisfaction.” In: *Human-Computer Interaction-INTERACT ’99*. Vol. 99. 1999, pp. 486–493. DOI: 10.1007/s00371-014-0963-0.

[19] M. J. Van Emmerik. “A Direct Manipulation Technique for Specifying 3D Object Transformations with a 2D Input Device.” In: *Computer Graphics Forum* 9.4 (1990), pp. 355–361. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.1990.tb00427.x.

[20] S. Zhai, W. Buxton, and P. Milgram. “The "Silk Cursor" Investigating Transparency for 3D Target Acquisition.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’94. Boston, Massachusetts, USA: ACM, 1994, pp. 459–464. ISBN: 0-89791-650-6. DOI: 10.1145/191666.191822.

[21] A. Forsberg, K. Herndon, and R. Zeleznik. “Aperture Based Selection for Immersive Virtual Environments.” In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. UIST ’96. Seattle, Washington, USA: ACM, 1996, pp. 95–96. ISBN: 0-89791-798-7. DOI: 10.1145/237091.237105.

[22] T. Grossman and R. Balakrishnan. "The Design and Evaluation of Selection Techniques for 3D Volumetric Displays." In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology.* UIST '06. Montreux, Switzerland: ACM, 2006, pp. 3–12. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166257.

[23] I. Poupyrev, T. Ichikawa, S. Weghorst, and M. Billinghurst. "Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques." In: *Computer Graphics Forum* 17.3 (1998), pp. 41–52. ISSN: 1467-8659. DOI: 10.1111/1467-8659.00252.

[24] J.-Y. Oh and W. Stuerzlinger. "Moving Objects with 2D Input Devices in CAD Systems and Desktop Virtual Environments." In: *Proceedings of Graphics Interface 2005.* GI '05. Victoria, British Columbia: Canadian Human-Computer Communications Society, 2005, pp. 195–202. ISBN: 1-56881-265-5.

[25] P. S. Strauss and R. Carey. "An Object-oriented 3D Graphics Toolkit." In: *SIGGRAPH Comput. Graph.* 26.2 (July 1992), pp. 341–349. ISSN: 0097-8930. DOI: 10.1145/142920.134089.

[26] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. "An introduction to 3-D user interface design." In: *Presence: Teleoperators and Virtual Environments* 10.1 (2001), pp. 96–108. DOI: 10.1162/105474601750182342.

[27] Unity Technologies. *Unity.* Version 2017.1.1.f1 personal. July 5, 2018. URL: https://unity3d.com/.

[28] B. Foundation. *Blender.* Version 2.7.9. July 5, 2018. URL: https://www.blender.org.

[29] S. Houde. "Iterative Design of an Interface for Easy 3-D Direct Manipulation." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* CHI '92. Monterey, California, USA: ACM, 1992, pp. 135–142. ISBN: 0-89791-513-5. DOI: 10.1145/142750.142772.

[30] G. M. Nielson and D. R. Olsen Jr. "Direct Manipulation Techniques for 3D Objects Using 2D Locator Devices." In: *Proceedings of the 1986 Workshop on Interactive 3D Graphics.* I3D '86. Chapel Hill, North Carolina, USA: ACM, 1987, pp. 175–182. ISBN: 0-89791-228-4. DOI: 10.1145/319120.319134.

[31] M. Chen, S. J. Mountford, and A. Sellen. "A Study in Interactive 3-D Rotation Using 2-D Control Devices." In: *SIGGRAPH Comput. Graph.* 22.4 (June 1988), pp. 121–129. ISSN: 0097-8930. DOI: 10.1145/378456.378497.

[32] Unity Technologies. *Positioning GameObjects. Move, Rotate, Scale, and RectTransform.* URL: https://docs.unity3d.com/Manual/PositioningGameObjects.html (visited on 07/06/2018).

[33] J. Sun, W. Stuerzlinger, and D. Shuralyov. "SHIFT-Sliding and DEPTH-POP for 3D Positioning." In: *Proceedings of the 2016 Symposium on Spatial User Interaction.* SUI '16. Tokyo, Japan: ACM, 2016, pp. 69–78. ISBN: 978-1-4503-4068-7. DOI: 10.1145/2983310.2985748.

[34] S. Rybicki, B. DeRenzi, and J. Gain. “Usability and Performance of Mouse-based Rotation Controllers.” In: *Proceedings of the 42Nd Graphics Interface Conference*. GI ’16. Victoria, British Columbia, Canada: Canadian Human-Computer Communications Society, 2016, pp. 93–100. ISBN: 978-0-9947868-1-4. DOI: `10.20380/GI2016.12`.

[35] Y. Chen and G. Medioni. “Object modeling by registration of multiple range images.” In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. Vol. 3. Apr. 1991, pp. 2724–2729. DOI: `10.1109/ROBOT.1991.132043`.

[36] R. Bade, F. Ritter, and B. Preim. “Usability Comparison of Mouse-Based Interaction Techniques for Predictable 3d Rotation.” In: *Smart Graphics: 5th International Symposium, SG 2005, Frauenwörth Cloister, Germany, August 22-24, 2005. Proceedings*. Ed. by A. Butz, B. Fisher, A. Krüger, and P. Olivier. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 138–150. ISBN: 978-3-540-31905-4. DOI: `10.1007/11536482_12`.

[37] Y. J. Zhao, D. Shuralyov, and W. Stuerzlinger. “Comparison of multiple 3D rotation methods.” In: *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011 IEEE International Conference on*. IEEE. Ottawa, ON, Canada: IEEE, 2011, pp. 1–5. ISBN: 978-1-61284-890-7. DOI: `10.1109/VECIMS.2011.6053855`.

[38] K. Shoemake. “Arcball: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse.” In: *Proc. Graphics Interface 92*. Vancouver, British Columbia, Canada: Canadian Human-Computer Communications Society, 1992, pp. 151–156.

[39] G. Bell. *Bell’s trackball*. 1993. URL: `https://web.archive.org/web/20070717090837/http://www.dispersoid.net/code/trackball.c` (visited on 11/01/2017).

[40] K. Henriksen, J. Sporring, and K. Hornbaek. “Virtual trackballs revisited.” In: *IEEE Transactions on Visualization and Computer Graphics* 10.2 (Mar. 2004), pp. 206–216. ISSN: 1077-2626. DOI: `10.1109/TVCG.2004.1260772`.

Chapter 3

Collision Detection

Numerous orthopaedic fracture reduction applications, be it for surgery preparation [1–6] or training [7], use collision detection to aid the user, especially in combination with haptic devices [2–7]. However, few mention how they detect object interference. Harders et al. [6] shortly mention that they use a specialized vector field approach. Only Scheuering et al. [1] discuss their implementation in such a way that it is reproducible.

Collision detection can be regarded as a pipeline that receives all objects in the scene as input and returns all colliding pairs of objects [8]. The first part of the pipeline, the broad phase, uses a fast test to enumerate all pairs of potentially colliding objects to be checked for exact intersection in the narrow phase [9]. Section 3.1 discusses the broad phase in more detail. In Section 3.2 we consider several approaches to the narrow phase. Section 3.4 is concerned with the challenges introduced by performing collision detection while moving the objects.

3.1 Broad Phase

The broad phase, also referred to as N-body culling, should efficiently remove pairs of objects that do not collide [9]. Kockara et al. [10] consider three categories of approaches to the broad phase: exhaustive search, coordinate sorting, and multi-level grids. The first is a brute-force approach that compares each object's bounding volume with the bounding volume of every other object. This approach requires $\mathcal{O}\left(N^2\right)$ comparisons for a scene with N objects. The topological approach, i.e. coordinate sorting, projects the objects' bounding volumes on one or more axes, e.g. the three coordinate axes. Pairs of objects whose projected bounding volumes overlap on at least one axis are labeled as colliding [9]. Multi-level grids take a spatial approach by diving the scene into a uniform grid. Only pairs of objects that share a cell of that grid are considered in the narrow phase [10].

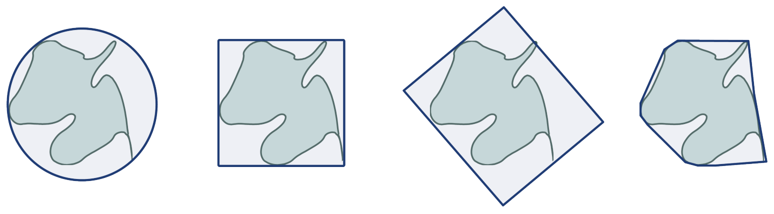
It has been shown that the minimum complexity of finding k colliding objects out of n objects is $\mathcal{O}\left(n \log^2 n + k\right)$ [9]. However due to the high constant factor, hidden by the asymptotic notation, the non-naive approaches are only profitable in scenarios with more than a hundred objects [9]. Since the scenes under consideration are very unlikely to contain that many objects, the naive approach suffices for use in a fracture reduction application. The performance of the naive approach is determined by the used bounding volumes. Section 3.1.1 discuses the different bounding volumes.

3.1.1 Bounding Volumes

Three characteristic properties of bounding volumes are: tightness, memory usage and the complexity of the intersection test. Generally there is a trade-off between these properties; a tighter bounding volume requires more memory to store and more complex intersection tests. Another consideration in the choice of a bounding volume is its rotation invariance. Several different bounding volumes are illustrated in Figure 3.1.

The simplest bounding volume shown in Figure 3.1 is the sphere. This bounding volume works best for relatively isotropic objects and fits flat geometries especially poorly [9]. An advantage of this bounding volume is that it can be stored very efficiently, since only four floating point numbers are needed [9, 12]. Not only is its memory complexity low, its distance and overlap tests are computationally efficient [9, 13]. Another strong point of this bounding volume is its rotation invariance, i.e. it does not need to be refitted after the enclosed object has been rotated. A disadvantage is that the minimal bounding sphere is difficult to find.

Axis-aligned bounding boxes (AABBs) surround the object with boxes oriented along its coordinate system's axes [14], as shown in Figure 3.1(b). The tightness of this bounding volume depends on the orientation of the enclosed object [15]; objects with a strong diagonal orientation result in AABBs with a lot of 'dead space' [14]. The two main advantages of AABBs are that they are efficient to store, and that their intersection tests are fast [9, 13]. Unfortunately, this bounding volume needs to be refitted after the bounded object has been rotated [9]. According to Kockara et al. [10] this process is sufficiently cost-effective for them to still be the preferred choice for deformable objects.



(a) *sphere* **(b)** *AABB* **(c)** *OBB* **(d)** *convex hull*

Figure 3.1: *A two-dimensional representation of the discussed bounding volumes: (a) spheres, (b) axis-aligned bounding boxes (AABBs), (c) oriented bounding boxes (OBBs), and (d) convex hulls. Images adapted from Langetepe and Zachmann [11].*

Contrary to AABBs, oriented bounding boxes (OBBs) have no constraints placed on their orientation. Consequently, they can generate a tighter fit [9, 13, 16] that does not depend on the orientation of the enclosed geometry [15]. This makes OBBs especially superior to AABBs in cluttered environments [17]. However, where an axis aligned bounding box is defined by six parameters, the definition of an oriented bounding box requires the storage of fifteen values [12]. Furthermore their intersection tests are more expensive than those of an AABB [9]. However, contrary to their axis-aligned cousin, oriented bounding boxes are rotation covariant [10].

Among the discussed bounding volumes the convex hull has the tightest fit [9]. Contrary to the other volumes, its memory requirement depends on the underlying geometry [9]. Another drawback of using the convex hull is the complexity of its intersection tests [9, 17]. However, it is rotation covariant.

Since fracture fragments are quite likely to be rotated, a rotation invariant bounding volume seems the logical choice for a fracture reduction application. Furthermore, since most bone fragments will be anisotropic an OBB seems to be the best choice of bounding volume in the broad phase. As the number of objects in the scene is likely to be quite low, the relatively high memory usage of these bounding volumes should not matter.

3.2 Narrow Phase

The narrow phase determines for each pair found by the broad phase if the objects in that pair collide. Kockara et al. [10] classifies approaches to this phase as being based on: features, simplices, the image-space, volume, and bounding volume hierarchies. The first class of algorithms works directly on the objects' geometric primitives. The most important disadvantage of feature-based approaches is that they require objects to be closed [10]. Simplex-based algorithms use the Minkowski distance between objects to detect collisions [10]. Consequently, this approach only works for convex objects [18]. Approaches based on the image-space detect collisions using the objects' discrete representation in the image space. Although this makes these approaches convenient to implement on the graphical processing unit (GPU), it may also result in erroneous representations. Furthermore, they are much slower than hierarchical approaches [10]. Volume-based approaches are built on the same idea as image-space based algorithms, but use a different method to compute the layered depth images and distance fields. These distance fields have to be recomputed after every transformation of an object. Finally, hierarchical methods use a tree of bounding volumes to approximate the object, or to decompose the space they occupy. Intersections between objects are detected by traversing the tree(s) of bounding volumes. Contrary to simplex and feature based algorithms, these methods do not impose any constraints on the objects [10]. They are often used in computer graphics, since they allow fine grained control over the trade-off between speed and precision [17].

Since fracture fragments are not necessarily closed or convex, neither feature-based nor simplex-based methods are appropriate. Furthermore, the precision of image-space based methods is too low. Volume-based methods are unsuited because they require refitting after every transformation. The generality and fine-grained control offered by bounding volume hierarchies (BVHs) makes them attractive for use in fracture reduction applications. We consider this approach in more detail in Section 3.3.

3.3 Bounding Volume Hierarchy

The narrow phase of a BVH approach typically consists of two parts: first the pairs of potentially colliding geometric primitives are found, and secondly primitive pairs are checked for collisions [9].

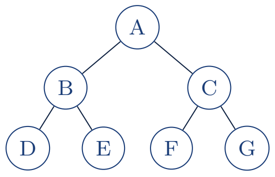
The hierarchy of bounding volumes can be used to partition the space, or the object itself. These methods are referred to as spatial and object partitioning, respectively. In the context of collision detection object partitioning is the superior approach. Firstly, because splitting polygons between bounding volumes is unavoidable while dividing the space, leading to a deeper hierarchy which reduces performance. Additionally, determining contact status with spatial partitioning is difficult if objects are near each other, since the cells of the spatial partitioning cannot cover objects' primitives tightly [10]. Finally, the spatial partitioning has to be recomputed or updated after any object has been transformed. If uniform subdivision is used, refitting can be done in constant time. However, this subdivision scheme is not suitable for scenes with objects of varying sizes [9]. Object partitioning, on the other hand, results in relatively small and tight hierarchies [10]. Furthermore, if a rotation invariant bounding volume is used, the hierarchy does not have to be refitted. Henceforth we focus on object partitioning, since that is the superior approach for our application.

The construction of an object partitioning bounding volume hierarchy is discussed in Section 3.3.1. Traversal of the hierarchies is reviewed in Section 3.3.2. Section 3.3.3 focuses on the time complexity of collision detection.

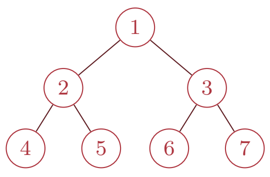
3.3.1 Construction

Bounding volumes are constructed either bottom-up, or top-down. The former starts with elementary bounding volumes of leaf nodes and merges them recursively until the root of the bounding volume hierarchy is reached. The more popular approach, top-down, starts with the object's bounding volume and divides that recursively [9].

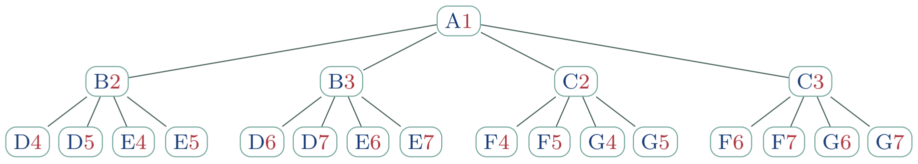
The performance of the construction process is influenced by two factors: the used splitting object, and the splitting or merging criterion. Building a tree of AABBs is cheaper in terms of both memory and computational power than a hierarchy of OBBs [17]. The construction speed of a BVH of OBBs can be increased by using heuristics to determine the approximate OBB instead of computing the optimal oriented bounding box [9].



(a) *BVH A*



(b) *BVH B*



(c) *bounding volume traversal tree*

Figure 3.2: *(c) The bounding volume traversal tree of the simultaneous recursive traversal of BVHs (a) A and (b) B. Images adapted from Weller [9].*

A classical splitting criterion is the longest side method which splits in the middle of the axis along which the bounding volume is longest [10]. Another simple method splits at the median of the longest axis of the elementary bounding boxes [9]. Neither of these methods are very robust. The surface area heuristic tries to avoid worst cases by optimizing the number of geometric primitives and the surface area over all possible split plane candidates [9]. Another more robust method aligns the splitting axis along the axis of maximum covariance. Several methods exists to determine where the axis should be split [10].

A simple approach to merging bounding volumes is to visit all nearest neighbors and minimize the size of the combined parent bounding volumes, that are at the same level. Alternative strategies are less greedy and combine bounding volumes using tilings [9].

3.3.2 Traversal

By traversing the objects' BVHs one can detect collisions between the objects. One traversal approach is a simultaneous recursive traversal of both trees, either depth-first, or breadth-first. This method traverses the trees until a leaf node has been reached in both trees, and an exact collision test between the geometric primitives, associated with the leaves, can be performed. The complete traversal algorithm results in a bounding volume test tree, as shown in Figure 3.2. Non-overlapping bounding volumes and their subtrees can be discarded to improve the algorithm's speed [9]. Recursive depth-first search is the easiest way to traverse both trees [9]. However, depending on the depth of the tree and the size of the stack this might lead to a stack overflow, in which case an iterative approach should be used [12].

For hierarchies where each node has n children, each pair of overlapping bounding volumes results in $\mathcal{O}\left(n^2\right)$ further comparisons [19]. Staircase traversal [20] only introduces $\mathcal{O}\left(n\right)$ further comparisons if two bounding volumes collide. On detecting a collision between the roots of the trees A and B this algorithm compares the nodes at height 1 of A with the root of B. Nodes that do not collide with the root of B are pruned from A. The remaining nodes at height 1 in A are tested against the nodes at the same height in B. Pairs that collide are the roots of a new sub-traversal [19].

3.3.3 Complexity

The execution time, T, of collision detection with hierarchical bounding volumes can be formulated as

$$T = N_v \cdot T_v + N_p \cdot T_p, \tag{3.1}$$

where N_v and N_p are the number of overlap tests between bounding volumes and primitives, respectively [16]. The time required for each of those tests is represented by, T_v and T_p, respectively. This formula illustrates that both the tightness of the bounding volumes and the simplicity of the overlap test influence the performance of the collision detection [13].

The tightness of the BVH influences both N_v and N_p. Gottschalk et al. [16] considered the influence of the distance between two spheres, placed both adjacently and concentrically, on N_v. They found that trees of OBBs required asymptotically fewer volume overlap tests,than those of AABBs. Furthermore, Gottschalk et al. [16] also observed that to cover a surface made up of m triangles with bounding volumes to a given tightness, AABBs and spheres require $\mathcal{O}\left(m^2\right)$ bounding volumes to achieve the same tightness that OBBs realize with $\mathcal{O}\left(m\right)$ bounding volumes [16]. Zachmann and Langetepe [15] observed that the tightness of OBBs decreases approximately linearly with the depth of the hierarchy, whereas the tightness of an AABB enclosing a surface of small curvature is almost the same as that of the parent bounding volume, making oriented bounding boxes much more attractive in terms of tightness. One disadvantage of using OBBs is that their trees are computationally more expensive to traverse than a tree of AABBs [15].

Zachmann [14] measured the execution time of collision detection of two adjacent identical objects as a function of the rotation of one object around its z-axis in increments of $0.18°$. They observed that using AABBs was significantly faster than using OBBs, and that independent of the bounding volume the initial pose of the objects strongly influenced T.

There is always a trade-off between the tightness of the BVH and the simplicity of the overlap tests. Chang et al. [13] attempt to get the best of both worlds by combining spheres, for their easy overlap tests, and OBBs, for their tightness, in a single tree. In their approach a comparison of two nodes happens in two steps, first the spheres are tested, and if they collide, the more expensive test with the OBBs is performed. A comparison between their dual approach and a BVH using OBB showed that especially for rigid bodies in static poses the dual approach is advantageous.

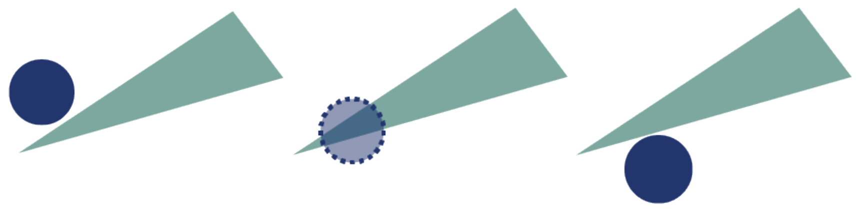


Figure 3.3: *An illustration of the tunneling effect. Although no collisions are detected at (a) time t_i or (c) t_{i+1}, the circle has collided with the rectangle (b) between t_i and t_{i+1}.*

3.4 Continuous Collision Detection

An easy approach to continuous collision detection is checking for collisions before every frame update. However, if the objects move too fast or if the time step between two queries is too large the tunneling effect could occur [9]. This effect happens if no collision is detected at time step t_i, or t_{i+1}, but a collision has occurred between the two between those two steps, as illustrated in Figure 3.3. Several techniques haven been proposed to solve this problem.

Pseudo-continuous collision detection aims to avoid the tunneling effect by performing static collision detection with a frequency that is higher than the frame rate. A smooth visual sensation requires a frame rate of at least 30 Hz, consequently the static collision query needs to be executed within approximately 33.33 ms. A clear advantage of the pseudo-continuous approach is that it allows the reuse of the static collision detection algorithm. However, even with higher sampling frequencies, contacts between thin objects may still be missed [9].

The problems of pseudo-continuous collision detection are avoided by the conservative advancement approach which advances by a certain time step that guarantees a non-penetration constraint. A different approach is to enclose the bounding volume at the beginning and end of every step with a swept volume. The downside of this method is that it requires an extra collision test [9].

References

[1] M. Scheuering, C. Rezk-Salama, C. Eckstein, K. Hormann, and G. Greiner. "Interactive Repositioning of Bone Fracture Segments." In: *Proceedings of the Vision Modeling and Visualization Conference 2001*. VMV '01. Aka GmbH, 2001, pp. 499–506. ISBN: 3-89838-028-9.

[2] P. Fürnstahl, G. Székely, C. Gerber, J. Hodler, J. G. Snedeker, and M. Harders. "Computer assisted reconstruction of complex proximal humerus fractures for preoperative planning." In: *Medical Image Analysis* 16.3 (2012). Computer Assisted Interventions, pp. 704–720. ISSN: 1361-8415. DOI: 10.1016/j.media.2010.07.012.

[3] P. Olsson, F. Nysjö, J.-M. Hirsch, and I. B. Carlbom. “A haptics-assisted cranio-maxillofacial surgery planning system for restoring skeletal anatomy in complex trauma cases.” In: *International Journal of Computer Assisted Radiology and Surgery* 8.6 (Nov. 2013), pp. 887–894. ISSN: 1861-6429. DOI: 10.1007/s11548-013-0827-5.

[4] J. Fornaro, M. Harders, M. Keel, B. Marincek, O. Trentz, G. Szekely, and T. Frauenfelder. “Interactive visuo-haptic surgical planning tool for pelvic and acetabular fractures.” In: *Studies in health technology and informatics* 132 (2008), pp. 123–125. ISSN: 0926-9630.

[5] J. Fornaro, M. Keel, M. Harders, B. Marincek, G. Székely, and T. Frauenfelder. “An interactive surgical planning tool for acetabular fractures: initial results.” In: *Journal of Orthopaedic Surgery and Research* 5.1 (Aug. 2010), p. 50. ISSN: 1749-799X. DOI: 10.1186/1749-799X-5-50.

[6] M. Harders, A. Barlit, C. Gerber, J. Hodler, and G. Székely. “An optimized surgical planning environment for complex proximal humerus fractures.” In: *MICCAI Workshop on Interaction in Medical Image Analysis and Visualization*. Vol. 10. Brisbane, Australia, Jan. 2007, pp. 201–206.

[7] D. Morris, C. Sewell, F. Barbagli, K. Salisbury, N. Blevins, and S. Girod. “Visuohaptic simulation of bone surgery for training and evaluation.” In: *IEEE Computer Graphics and Applications* 26.6 (Nov. 2006), pp. 48–57. ISSN: 0272-1716. DOI: 10.1109/MCG.2006.140.

[8] P. Hubbard. “Interactive collision detection.” In: *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*. Oct. 1993, pp. 24–31. DOI: 10.1109/VRAIS.1993.378267.

[9] R. Weller. “A Brief Overview of Collision Detection.” In: *New Geometric Data Structures for Collision Detection and Haptics*. Heidelberg: Springer International Publishing, 2013, pp. 9–46. ISBN: 978-3-319-01020-5. DOI: 10.1007/978-3-319-01020-5_2.

[10] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe. “Collision detection: A survey.” In: *2007 IEEE International Conference on Systems, Man and Cybernetics*. Oct. 2007, pp. 4046–4051. DOI: 10.1109/ICSMC.2007.4414258.

[11] E. Langetepe and G. Zachmann. *Geometric Data Structures for Computer Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006. ISBN: 1568812353.

[12] S. Gottschalk. “Collision Queries using Oriented Bounding Boxes.” PhD thesis. The University of North Carolina, 2000.

[13] J.-W. Chang, W. Wang, and M.-S. Kim. “Efficient collision detection using a dual OBB-sphere bounding volume hierarchy.” In: *Computer-Aided Design* 42.1 (2010). Advances in Geometric Modelling and Processing, pp. 50–57. ISSN: 0010-4485. DOI: 10.1016/j.cad.2009.04.010.

[14] G. Zachmann. “Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques.” PhD thesis. Technische Universität Darmstadt, July 2000.

[15] G. Zachmann and E. Langetepe. “Geometric Data Structures for Computer Graphics.” Siggraph 2003 Tutorial 16. 2003.

[16] S. Gottschalk, M. C. Lin, and D. Manocha. “OBBTree: A Hierarchical Structure for Rapid Interference Detection.” In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH ’96. ACM, 1996, pp. 171–180. ISBN: 0-89791-746-4. DOI: 10.1145/237170.237244.

[17] P. Jiménez, F. Thomas, and C. Torras. “3D collision detection: a survey.” In: *Computers & Graphics* 25.2 (2001), pp. 269–285. ISSN: 0097-8493. DOI: 10.1016/S0097-8493(00)00130-8.

[18] S. Oh and S. Hwang. “A GJK Based Real-Time Collision Detection Algorithm for Moving Objects.” In: *Advances in Cognitive Neurodynamics ICCN 2007: Proceedings of the International Conference on Cognitive Neurodynamics. ICCN 2007 Proceedings.* Ed. by R. Wang, E. Shen, and F. Gu. Dordrecht: Springer Netherlands, 2008, pp. 817–820. ISBN: 978-1-4020-8387-7. DOI: 10.1007/978-1-4020-8387-7_142.

[19] G. Bradshaw. “Bounding volume hierarchies for level-of-detail collision handling.” PhD thesis. Trinity College Dublin, May 2002.

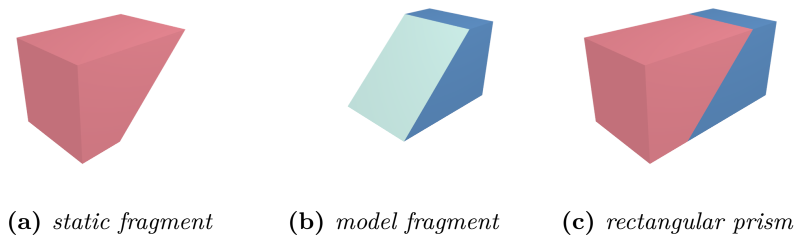
[20] I. J. Palmer and R. L. Grimsdale. “Collision Detection for Animation using Sphere-Trees.” In: *Computer Graphics Forum* 14.2 (1995), pp. 105–116. ISSN: 1467-8659. DOI: 10.1111/1467-8659.1420105.

Chapter 4

Simulated Data

Other than the actual registration, one of the challenges of research in this field is the quantification of results, since a ground truth is generally not available for clinical data. To ensure that we know the translation and rotation the registration algorithm is supposed to find, we use a simulated fracture and transform one of the fragments manually. We base this not on clinical data, but on a 3D shape that we have broken virtually. The advantage of this approach, compared to using clinical data is that we have a correct ground truth that allows us to distinguish rotation errors from translation errors.

We use these data to test the four methods presented in Section 3 of Part I: point-to-plane closed form ($\mathrm{CF}_{\mathrm{pl}}$), point-to-point closed form ($\mathrm{CF}_{\mathrm{po}}$), point-to-point iterative gradient descent ($\mathrm{IGD}_{\mathrm{po}}$), and intersection iterative gradient descent ($\mathrm{IGD}_{\mathrm{i}}$). Section 4.1 discusses the generation of this dataset and the methods used to quantify the results. The actual results are presented and discussed in Sections 4.2 and 4.3, respectively. In Section 4.4 we draw conclusions based on the previous sections, and Section 4.5 gives some ideas for future research.



(a) *static fragment* **(b)** *model fragment* **(c)** *rectangular prism*

Figure 4.1: *To generate the initial (a) static and (b) model fragment for the simulated data, we split (c) a rectangular prism of length three in two parts. The resulting fracture surface is shown in green.*

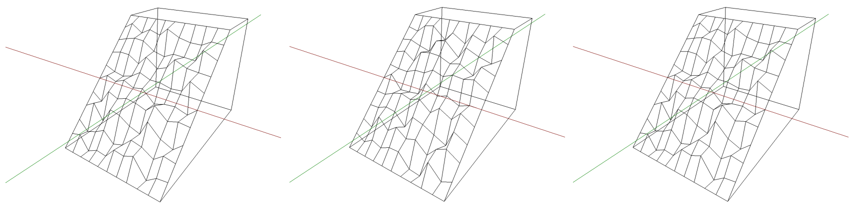

(a) a **(b)** b **(c)** c

Figure 4.2: *The three model fragments for the base fragment pairs (a)* a*, (b)* b*, and (c)* c*, with the respective seeds 2, 3, and 42.*

4.1 Experiment

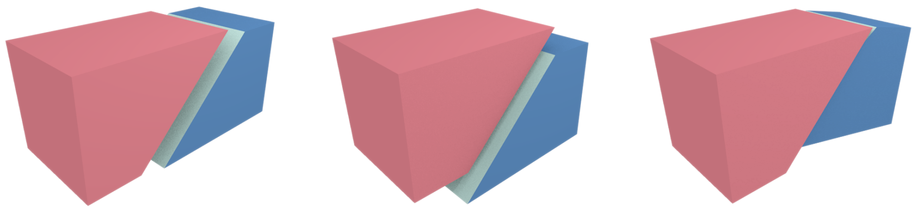
Section 4.1.1 discusses the generation of the simulated data. The quantification of the registration of the resulting fragment pairs is introduced in Section 4.1.2.

4.1.1 Data

The process used to generate the initial fragments is illustrated in Figure 4.1. We start with a $3 \times 1 \times 1$ rectangular prism. Next we split this prism with a plane that is placed such that the longest lower edges are split at one-third, and the two parallel upper edges are split at two-thirds.

We then subdivide the fracture surfaces of both the static and the model fragment into a 10×10 uniform grid of quads, using the subdivision tool implemented in Blender [1]. A uniformly sampled random offset is added in all three dimensions to the inner vertices of the static fragment, without changing the topology. The random offset are generated with the seed of the random generator set to 2, 3 and 42, resulting in the basis fragment pairs a, b, and c, respectively. The inner vertices of the model fragment are moved to ensure that the two shapes still fit perfectly. The resulting model fragments are shown in Figure 4.2. To better simulate noisy measurements we add noise, sampled from a zero-mean Gaussian distribution with standard deviations 0, 0.001, and 0.005, to the inner vertices of all fracture surfaces. Consequently, the fit between two fragments in a pair where noise has been added is not necessarily perfect. The addition of noise, with standard deviations 0, 0.001, and 0.005, creates the fragment pairs a_0, $\mathsf{a}_{0.001}$, and $\mathsf{a}_{0.005}$ from fragment pair a, respectively.

From these nine base fragment pairs we create three different datasets: S, O, and R. The model fragments in set S are moved along the vector between $\mathbf{o}_{\mathscr{X}}$ and $\mathbf{o}_{\mathscr{Y}}$, the origins of the oriented bounding boxes (OBBs) of the model and the static fragments, respectively. The new position of the origin of the OBB of



(a) S (b) O (c) R

Figure 4.3: *Examples of fragment pairs form dataset (a)* S*, (b)* O*, and (c)* R*. All pairs are based on the base pair* $\mathsf{c}_{0.001}$*. The model fragments are shown in red, the static fragments in blue, and the fracture surface in green. To generate the fragment pair Figure 4.3(a), s was set to* 1.300*. The origin of the OBB of the model fragment in Figure 4.3(b) was offset with* 0.2, −0.4, 0.2 *from its original position. Figure 4.3(c) was generated by rotating the model fragment* $10°$ *around the x-axis,* $5°$ *around the y-axis, and* $-0.1000°$ *around the z-axis.*

the model fragment is

$$\mathbf{o}'_{\mathscr{X}} = s\left(\mathbf{o}_{\mathscr{X}} - \mathbf{o}_{\mathscr{Y}}\right), \tag{4.1}$$

where $s \in \{0.70, 0.75, \ldots, 1.30, 1.35\}$. The resulting dataset consists of 126 fragment pairs. Dataset O is created by translating $\mathbf{o}_{\mathscr{X}}$ with some translation vector $\mathbf{w} \in \{-0.2, -0.1, 0, +0.1, +0.2\}^3$. The thus generated dataset has 1125 fragment pairs. Finally, dataset R is generated by rotating the model fragment $\{-10, -5, -1, -0.5, -0.1, 0, 0.1, 0.5, 1, 5, 10\}$ degrees around the x, y, and z-axes, with the center of rotation set to $\mathbf{o}_{\mathscr{X}}$. The resulting dataset has cardinality 11 979. Figure 4.3 shows a pair of fracture fragments from each of the three datasets.

For these datasets we used a fixed error threshold, which is set to 1×10^{-4}. Since both the static and the fragment models of all base pairs have 250 vertices after triangulation they are not subsampled by the application.

4.1.2 Quantification

For these data, quantification is relatively straightforward, since we know the expected translation and rotation. To quantify the difference between two translations $\mathbf{t}_1$, and $\mathbf{t}_2$, we use

$$D_t\left(\mathbf{t}_1, \mathbf{t}_2\right) = \sqrt{\sum_i^3 \mathbf{t}_1^{(i)} - \mathbf{t}_2^{(i)}} \tag{4.2}$$

where $\mathbf{t}_1^{(i)}$ denotes the ith element of $\mathbf{t}_1$.

We express the distance between two rotations $\mathbf{r}_1$ and $\mathbf{r}_2$ as the inner product of their unit quaternions [2]

$$D_r\left(\mathbf{r}_1, \mathbf{r}_2\right) = 1 - \left|\frac{1}{|\mathbf{r}_1|}\mathbf{r}_1 \cdot \frac{1}{|\mathbf{r}_2|}\mathbf{r}_2\right|. \tag{4.3}$$

The range of the resulting metric is $[0, 1]$.

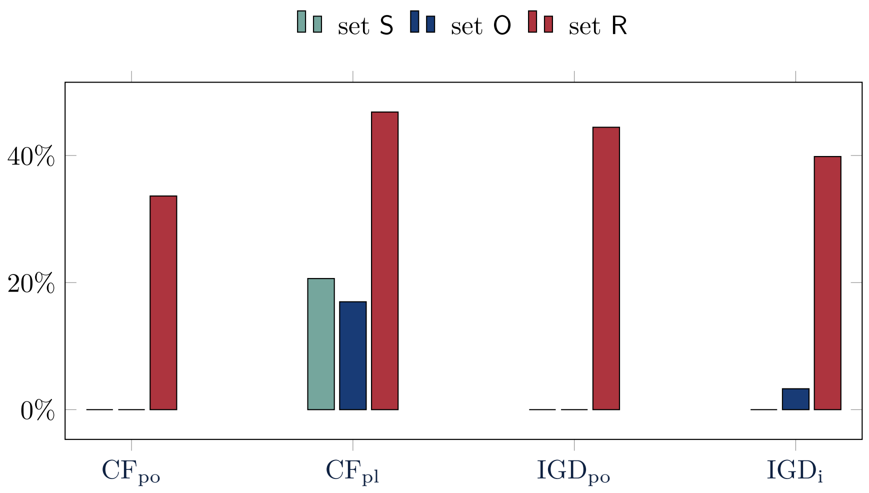


Figure 4.4: *The percentage of fragment pairs where the registration changed the morphology of the model fragment, grouped per method and dataset.*

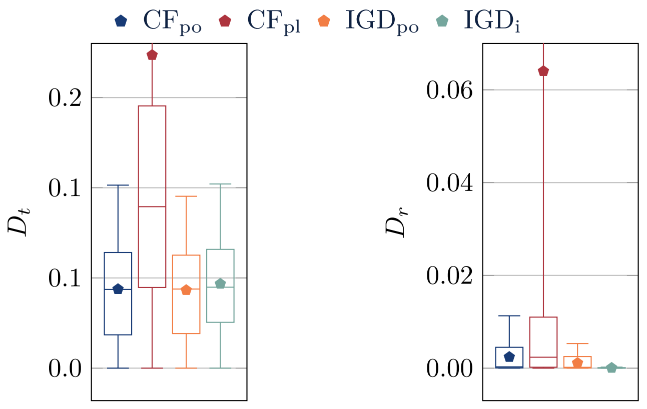
4.2 Results

The transformation applied to register the objects should be rigid, however for some model fragments, the registration algorithm has changed their morphology to that of a 2D shape, i.e. the values of the vertices in e.g. the x-dimension were all changed to 0.1349×10^{-8}. Figure 4.4 shows the percentage of fragment pairs, per dataset, whose model fragment's morphology has been changed, per registration method. We observe that all methods failed on approximately 35 % of fragment pairs from dataset R. Only the CF$_{pl}$ method failed on a significant percentage of dataset S and O. We exclude the pairs where the morphology of the model fragment was changed from the presentation and discussion of the results. In Sections 4.2.1 to 4.2.3 we present the results of the registration of the other fragment pairs for datasets S, O, and R, respectively.

4.2.1 Dataset S

Figure 4.5 shows a box-and-whisker plot of the translation and rotation error of the four registration methods on dataset S. We observe that the CF$_{pl}$ method is outperformed by the other registration methods. Based on the mean translation and rotation error of this registration method relative to its median errors, the high average error can be attributed to a limited number of very high errors. Furthermore, Figure 4.5(a) shows that the method IGD$_{i}$ introduces hardly any rotation errors. We review the influence of σ_{noise}, the standard deviation of the noise added to the vertices and the scale s on both the translation and the rotation error.

Figure 4.6 illustrates the influence of the added noise and the registration method on the translation errors in S. We observe very little influence of the noise on the results of registration methods CF$_{po}$, IGD$_{po}$, and IGD$_{i}$. This



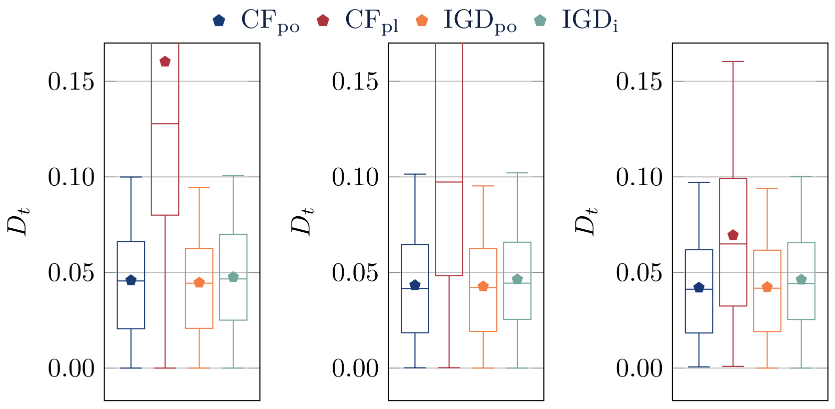
(a) *translation error* **(b)** *rotation error*

Figure 4.5: *Box-and-whisker plot of (a) the translation and (b) the rotation error on dataset* S *per dataset. The boxes show the 25th, 50th, and 75th percentile. The diamonds indicate the means, and the whiskers the minimum and maximum errors. The maximum translation (*0.8973*) and rotation error (*0.4980*) for the method CF*$_{pl}$ *are not shown.*

is confirmed by four one-way ANOVAs to compare the effect of σ_{noise} on D_t. The analysis was not significant at the significance level 0.05 for the methods point-to-point closed form ($F_{1,124} = 0.314$, $p = 0.576$), point-to-point iterative gradient descent ($F_{1,124} = 0.108$, $p = 0.744$), and intersection iterative gradient descent ($F_{1,124} = 0.0331$, $p = 0.856$). For the method CF$_{\text{pl}}$ we found a significant effect of σ_{noise} on D_t ($F_{1,98} = 11.3$, $p < 0.010$). Therefore, only the translation error of the method CF$_{\text{pl}}$ is influenced by the added noise. In Figure 4.6 the results of the four methods seem comparable for $\sigma_{\text{noise}} = 0.005$, this is confirmed by a one-way ANOVA on the influence of the registration method on the translation error ($F_{3,168} = 0.237$, $p = 0.871$). Consequently, the method CF$_{\text{pl}}$ only performs comparably to the other methods if Gaussian distributed noise with $\sigma_{\text{noise}} = 0.005$ is added to the vertices.

Figure 4.7 gives an overview of the rotation errors introduced in dataset S as a function of the registration methods and σ_{noise}. It should be noted that the rotation error of the initial alignment of all pairs in dataset S is zero. Therefore, the rotation errors shown in Figure 4.5(b) are introduced by the registration method. As for the translation error, only the rotation error of CF$_{\text{pl}}$ seems to be influenced by σ_{noise}. This is confirmed by four one-way ANOVAs that show that the influence of σ_{noise} on D_t is not significant for the registration methods CF$_{\text{po}}$ ($F_{1,124} = 0.830$, $p = 0.364$), IGD$_{\text{po}}$ ($F_{1,124} = 0.955$, $p = 0.330$), and IGD$_{\text{i}}$ ($F_{1,124} = 0.959$, $p = 0.329$). Furthermore, the registration error introduced by these methods is nearly zero, in this aspect the method IGD$_{\text{i}}$ performs particularly well. The highest rotation error introduced by this method is 2.226×10^{-4} for a pair from $\mathsf{S}_{0.005}$. For fragment pairs registered by CF$_{\text{pl}}$ the difference caused by σ_{noise} was significant ($F_{1,98} = 9.92$, $p < 0.010$). Consequently, only the rotation error generated by CF_{pl} is strongly influenced by the added noise. For $\sigma_{\text{noise}} = 0.001$ CF_{pl} only introduced rotation errors for about half of the fragment pairs. However, the introduced errors are relatively large, as can be seen in Figure 4.7(b).

Figure 4.8 shows the mean translation errors of the different registration methods as a function of s. Based on this figure, both s and the used method influence the D_t. This is confirmed by a two-way ANOVA that considers the effect of s and registration method on D_t and the interaction effect between s and the registration method. Both s ($F_{13,422} = 5.94$, $p < 0.001$) and the used registration method ($F_{3,422} = 35.7$, $p < 0.001$) had a statistically significant influence on the translation distance. The interaction effect was also significant ($F_{39,422} = 1.51$, $p < 0.050$). In Figure 4.8 we can see that significant influence of the registration method can be ascribed to the deviating results by the registration method CF_{pl}. Fourteen one-way ANOVAs with the other three registration methods as main factor, show no significant influence of the scale on the mean translation distance. We also observe the interaction effect in Figure 4.8, since the method CF_{pl} responds differently to the changes in scale than the other three methods. For example, for $s > 1$ the average translation error increases faster for CF_{pl} as a function of s, than for the other methods. In the same figure we also find that the average translation error increases as the absolute value of s increases. For $s = 0$ the average translation error should be zero, since the fragments are already perfectly aligned. Reviewing the actual values of $\overline{D}_t$ for $s = 0$, reveals that only the iterative gradient descent (IGD) methods do not introduce translation errors, CF_{po} and CF_{pl} introduce translation errors that are 2.776×10^{-4} and 4.250×10^{-4}, respectively.



(a) $\mathsf{S}_{0.0}$ **(b)** $\mathsf{S}_{0.001}$ **(c)** $\mathsf{S}_{0.005}$

Figure 4.6: *Box-and-whisker plot of the translation error per registration method for datasets (a)* $\mathsf{S}_{0.0}$*, (b)* $\mathsf{S}_{0.001}$*, and (c)* $\mathsf{S}_{0.005}$*. For dataset* $\mathsf{S}_{0.0}$ *and* $\mathsf{S}_{0.001}$ *the upper quartile and the maximum of the translation error of the method* CF_{pl} *are not shown. The absent 75th percentiles are* 0.6089 *and* 0.2218*, and the the maximum translation errors are* 0.8973 *and* 0.3808*, for* $\mathsf{S}_{0.0}$ *and* $\mathsf{S}_{0.001}$*, respectively.*

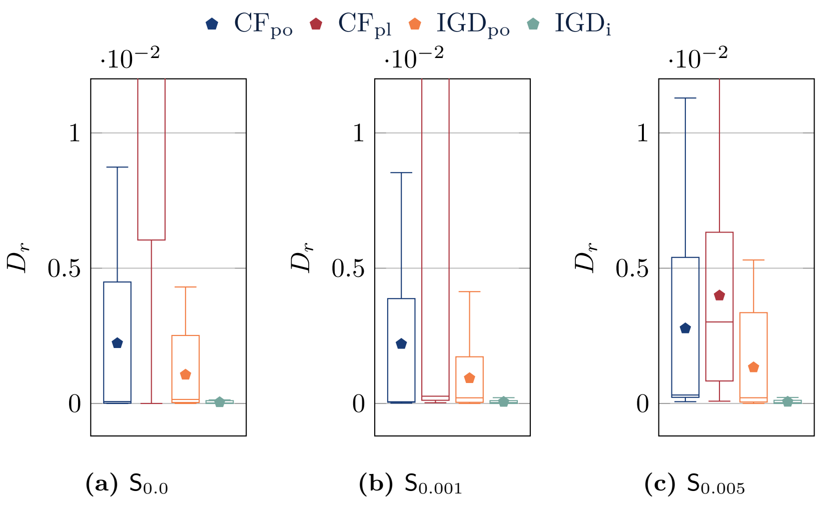


Figure 4.7: *Box-and-whisker plot of the rotation error per registration method for datasets (a)* S$_{0.0}$*, (b)* S$_{0.001}$*, and (c)* S$_{0.005}$*. The missing values for the method CF$_{pl}$ can be found in Table 4.1.*

Rotation errors were subjected to a two-way ANOVA with the main factors registration method and s. All effects were statistically significant at the 0.05 significance level. In Figure 4.9 the interaction effect ($F_{39,422} = 1.98$, $p < 0.001$) is illustrated by the differing response by the registration method CF$_{pl}$ to changes in scale for $s < 1$. Based on this figure the significance of the factors s ($F_{13,422} = 1.95$, $p < 0.050$) and registration method ($F_{3,422} = 22.3$, $p < 0.001$) are due to the rotation error the method CF$_{pl}$ introduces for $s < 1$. However, 14 one-way ANOVAs, one for each value of the factor s, without the registration method CF$_{pl}$, show that the method is a significant factor for every value of s, except for $s = 0.95$ and $s = 1.15$. Figure 4.10 shows the rotation errors introduced by the methods CF$_{po}$, IGD$_{po}$, and IGD$_{i}$. This plot shows that for $s < 0.9$, the method IGD$_{i}$ is the only method that introduces hardly any rotation error. The biggest rotation error for $s < 0.9$ caused by this method is 9.670×10^{-5} and occurs at $s = 0.7$.

				percentile		
	min	max	mean	25th	50th	75th
S$_{0.0}$	0.00	1.28e^{-1}	5.00e^{-2}	6.04e^{-3}	4.05e^{-2}	7.25e^{-2}
S$_{0.001}$	2.69e^{-5}	4.98e^{-1}	1.29e^{-1}	1.18e^{-4}	2.69e^{-4}	3.63e^{-1}
S$_{0.005}$	8.54e^{-5}	1.34e^{-2}	3.99e^{-3}	8.32e^{-4}	3.01e^{-3}	6.33e^{-3}

Table 4.1: *The minimum, maximum, mean, 25th, 50th, and 75th percentile of the rotation error of the method CF$_{pl}$ on dataset* S.

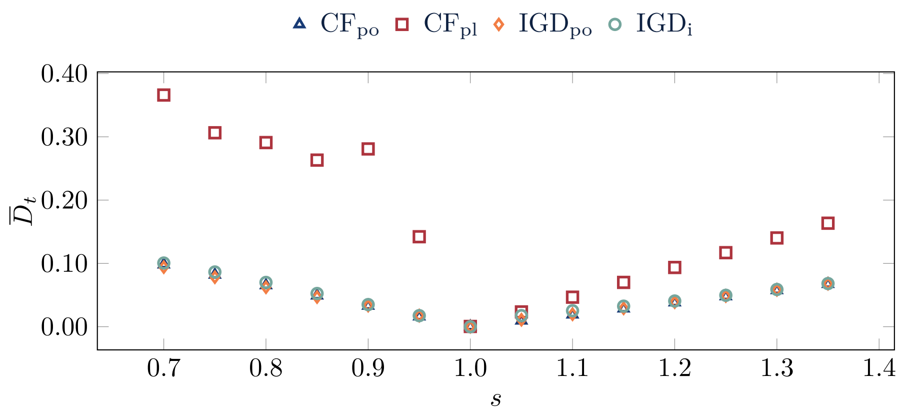


Figure 4.8: *The mean translation error, $\overline{D}_t$, for dataset* S *as a function of the scale factor s, split by registration method.*

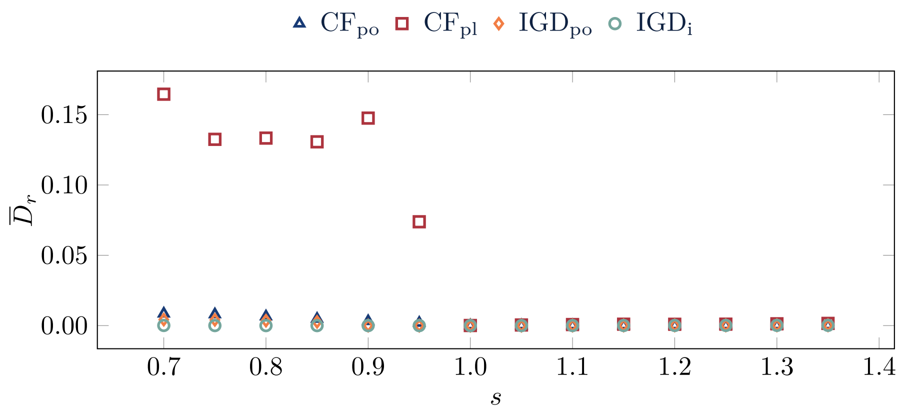


Figure 4.9: *The mean rotation error, $\overline{D}_r$, for dataset* S *as a function of the scale factor s, split by registration method.*

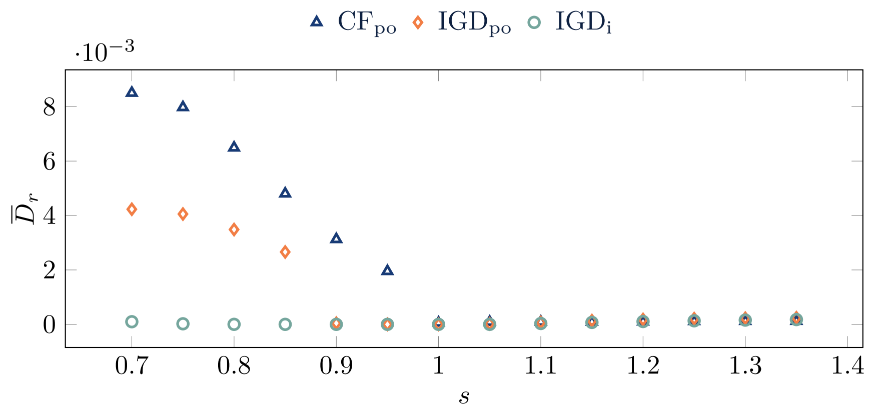


Figure 4.10: *The mean rotation error, $\overline{D}_r$, for dataset* S *as a function of the scale factor s, for the registration methods CF$_{po}$, IGD$_{po}$, and IGD$_i$.*

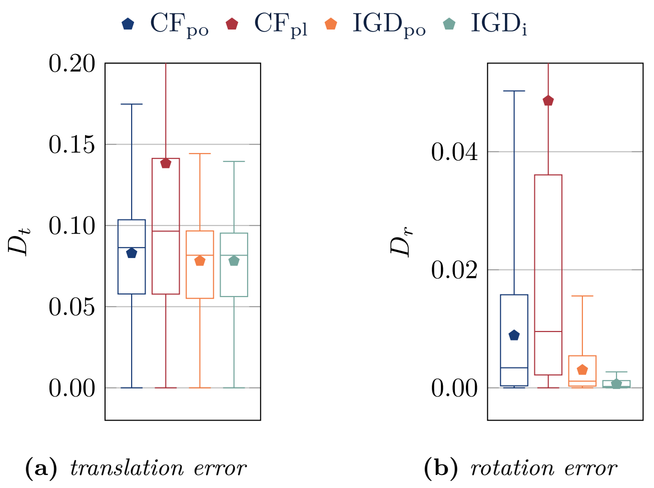


Figure 4.11: *Box-and-whisker plot of (a) the translation and (b) the rotation error on dataset* O. *The maximum translation and rotation error for the method CF$_{pl}$ are not shown, the missing values are* 1.677 *and* 0.5000*, respectively.*

4.2.2 Dataset O

Figure 4.11 shows the performance of the four different registration methods on dataset O. In these plots we observe that only the minimum translation and rotation error of the method CF$_{\text{pl}}$ are on par with the other registration methods. Furthermore, we find that the variance of this method is higher than that of the other methods. With regard to the translation error, registration with the method IGD$_{\text{i}}$ seems to slightly outperform both IGD$_{\text{po}}$ and CF$_{\text{po}}$. Furthermore, the minimum error of each registration method is zero, suggesting that all of them have achieved a near perfect registration for at least one fragment pair. The IGD methods outperform the closed-form methods w.r.t. D_r. The rotation error of the method IGD$_{\text{i}}$ is in general a lot lower than those associated with the other registration methods. The maximum error of the method IGD$_{\text{i}}$ is nearly the same as its minimum error, indicating consistent performance over all fragment pairs in dataset O.

To investigate the influence of the noise on the performance of the different registration methods we use Kruskal-Wallis tests within the registration methods on the factor σ_{noise}. The added noise did not influence the translation error of the methods CF$_{\text{po}}$ ($H_2 = 5.73$, $p = 0.057$), IGD$_{\text{po}}$ ($H_2 = 3.09$, $p = 0.213$), or IGD$_{\text{i}}$ ($H_2 = 3.81$, $p = 0.148$). Only the translation errors of the method CF$_{\text{pl}}$ ($H_2 = 25.1$, $p < 0.001$) are influenced by the noise at the significance level 0.05. The difference in translation error between O$_{0.0}$ ($\mu = 0.200$, $\sigma = 0.148$) and O$_{0.001}$ ($\mu = 0.130$, $\sigma = 0.152$) was significant if the method was registered with CF$_{\text{pl}}$ ($t_{397} = 5.27$, $p < 0.001$). The difference between O$_{0.0}$ and O$_{0.005}$ ($\mu = 0.114$, $\sigma = 0.109$) was also significant for this method ($t_{302} = 7.08$, $p < 0.001$). Only the difference between O$_{0.001}$ and O$_{0.005}$ was not significant ($t_{671} = -1.61$, $p = 0.109$). This suggests that its translation error decreases with the addition of more noise, but that the standard deviation of the added

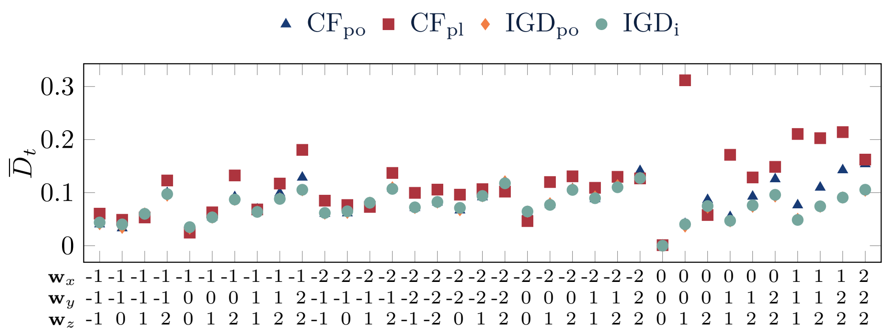


Figure 4.12: *The average translation error per method and offset* $\mathbf{w}$ *for dataset* $\mathsf{O}_{0.005}$. *The offset, scaled by a factor ten, is shown in three rows below the* x*-axis, the first row denotes the offset in the* x *direction, the second in the* y *direction and the third in the* z *direction.*

noise does not matter much.

We use the same approach to investigate the influence of σ_{noise} on the rotation error. For this error metric the Kruskal-Wallis tests reveal a significant influence of the factor σ_{noise} for all registration methods. Post-hoc tests reveal that the difference in rotation error for the zero-noise condition and the two noise-added conditions were significant for all methods except for the CF$_{\text{po}}$ method. The other methods all showed significant differences in performance between O_0 and $\mathsf{O}_{0.001}$ on the one hand and O_0 and $\mathsf{O}_{0.005}$ on the other hand. No significant differences in performance were found between $\mathsf{O}_{0.001}$ and $\mathsf{O}_{0.005}$ for any registration method. For all registration methods, except the method CF$_{\text{po}}$, the average rotation error decreased as noise was added.

We investigate the influence of the offset vector $\mathbf{w}$ on the translation error with the results of $\mathsf{O}_{0.005}$. Figure 4.12 shows the mean translation error per offset vector $\mathbf{w}$ for each registration method. Firstly, this graph shows that the difference in performance between the CF$_{\text{po}}$ and the IGD methods is quite likely due to a few patterns. Most of these cases are found on the right side of the graph, where the offset vector has mostly relatively large positive values. Furthermore, for all methods except CF$_{\text{pl}}$, it seems as though for each fixed value of $\mathbf{w}_x$ the translation error increases as the y and z components increase. This effect is especially visible for $\mathbf{w}_x \geq 0$.

Only the addition of noise, not its standard deviation, significantly influenced the rotation error of all levels. Therefore we combine the results of $\mathsf{O}_{0.001}$ and $\mathsf{O}_{0.005}$ into $\mathsf{O}_{0.001 \cup 0.005}$ and consider the rotation errors of the registration methods on that set separately from those on $\mathsf{O}_{0.0}$.

Figure 4.13 shows the rotation error per offset for dataset $\mathsf{O}_{0.0}$. In this figure we observe that the performance of the method CF$_{\text{pl}}$ ($\mu = 0.0926$, $\sigma = 0.119$) is worse than that of the other methods. A t-test reveals that the rotation errors introduced by this method are significantly worse than those caused by the second worst method ($t_{7345} = 9.71$, $p < 0.001$), point-to-point closed form ($\mu = 0.008\,60$, $\sigma = 0.009\,74$). Furthermore, for a subset of offset vectors, the method CF$_{\text{po}}$ performs worse than the IGD methods. It is quite likely due to these

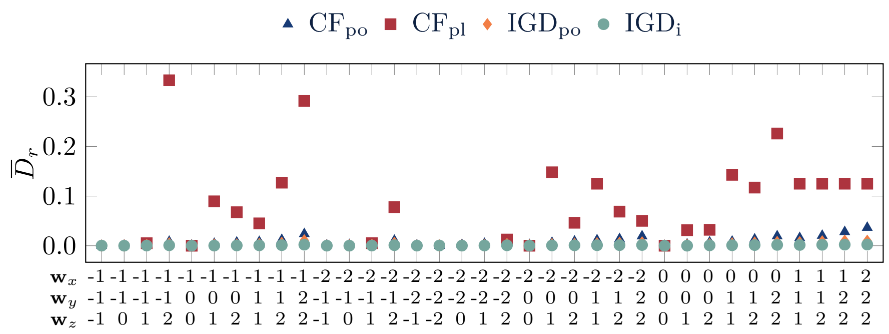


Figure 4.13: *The average rotation error per method and offset* $\mathbf{w}$ *for dataset* $\mathsf{O}_{0.0}$*. The offset, scaled by a factor ten, is shown in three rows below the x-axis, the first row denotes the offset in the x direction, the second in the y direction and the third in the z direction.*

offsets that the rotation errors introduced by CF$_{po}$ are significantly different from those introduced by IGD$_{po}$ ($t_{508} = -8.60$, $p < 0.001$) and IGD$_i$ ($t_{380} = -15.5$, $p < 0.001$). A t-test reveals that the methods IGD$_{po}$($\mu = 0.003\,89$, $\sigma = 0.004\,19$) and IGD$_i$($\mu = 0.000\,769$, $\sigma = 0.000\,845$) are significantly different at the 0.05 significance level ($t_{15405} = 14.1$, $p < 0.001$).

Figure 4.14 shows that the method CF$_{pl}$ ($\mu = 0.0373$, $\sigma = 0.0929$) also performs worse w.r.t. the rotation error than the other methods on dataset $\mathsf{O}_{0.001\cup0.005}$. The difference in introduced rotation error between the methods CF$_{pl}$ and CF$_{po}$ ($\mu = 0.009\,02$, $\sigma = 0.0107$) is significant at the 0.05 significance level ($t_{761} = -8.22$, $p < 0.001$). With the introduction of noise, the method CF$_{po}$($\mu = 0.009\,02$, $\sigma = 0.0107$) is still outperformed by the methods IGD$_{po}$ ($t_{508} = 15.8$, $p < 0.001$) and IGD$_i$ ($t_{380} = 21.6$, $p < 0.001$). Between the two IGD methods, IGD$_{po}$($\mu = 0.002\,60$, $\sigma = 0.002\,88$) and IGD$_i$($\mu = 0.000\,552$, $\sigma = 0.000\,605$) the difference in performance w.r.t. D_r is significant ($t_{818} = 19.0$, $p < 0.001$).

Both Figure 4.13 and 4.14 show that especially the closed-form solutions are

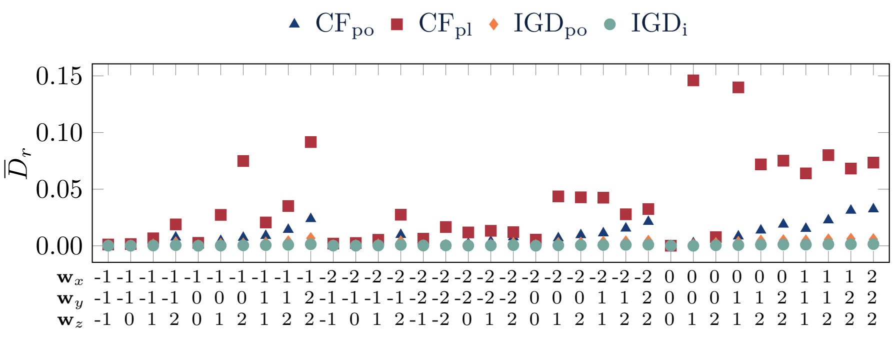


Figure 4.14: *The average rotation error per method and offset* $\mathbf{w}$ *for dataset* $\mathsf{O}_{0.001\cup0.005}$*. The offset, scaled by a factor ten, is shown in three rows below the x-axis, the first row denotes the offset in the x direction, the second in the y direction and the third in the z direction.*

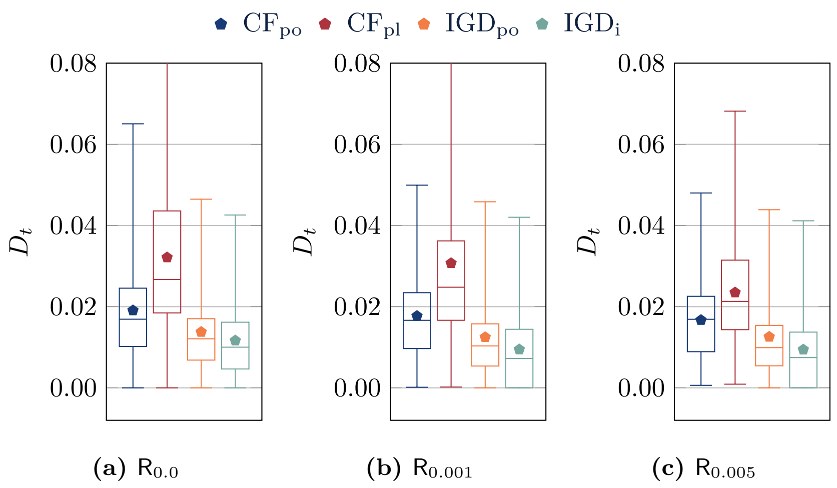


Figure 4.15: *Box-and-whisker plot for the translation error per registration method for datasets (a)* $\mathsf{R}_{0.0}$*, (b)* $\mathsf{R}_{0.001}$*, and (c)* $\mathsf{R}_{0.005}$*. The maximum errors of the method CF$_{pl}$ for datasets* $\mathsf{R}_{0.001}$ *and* $\mathsf{R}_{0.005}$ *are not shown, they are* 0.2934 *and* 0.4776*, respectively.*

influenced by the translation vector, independent of the addition of noise. Although the offset vectors for which the worst results were achieved are the same for $\mathsf{O}_{0.0}$ and $\mathsf{O}_{0.001\cup0.005}$, we could not observe a direct relationship between the elements of the offset vector and the performance of the registration methods.

4.2.3 Dataset R

Figures 4.15 and 4.16 show box-and-whisker plots of respectively the rotation and translation error of dataset R for each registration method per value of σ_{noise}. A Kruskal-Wallis test on the factor noise indicates that σ_{noise} has a significant influence on the translation ($H_2 = 163$, $p < 0.001$) and the rotation error ($H_2 = 96.7$, $p < 0.001$). In Figure 4.15 we observe that the translation error decreases for all methods as the added noise increases. Furthermore, for $\sigma_{\text{noise}} > 0.0$ the minimum translation error of the method IGD$_{\text{i}}$ is nearly zero. Comparing the translation and rotation errors for the different registration methods within $\mathsf{R}_{0.0}$, $\mathsf{R}_{0.001}$, and $\mathsf{R}_{0.005}$ with Kruskal-Wallis tests shows that the differences between the registration methods are significant, within all noise conditions. Furthermore, using t-tests we have found that the difference in performance w.r.t. both the translation and the rotation error, within the noise condition is significant at the 0.05 significance level for every registration method pair.

We use a Kruskal-Wallis test to investigate the influence of the factor rotation, which has 286 levels, on D_t and D_r. For both the translation ($H_{285} = 8970$, $p < 0.001$) and the rotation error ($H_{285} = 16\,900$, $p < 0.001$) the effect of the rotation is significant. If we look within σ_{noise} and method, we find that for all methods, for all noise levels, the differences in performance w.r.t. both the

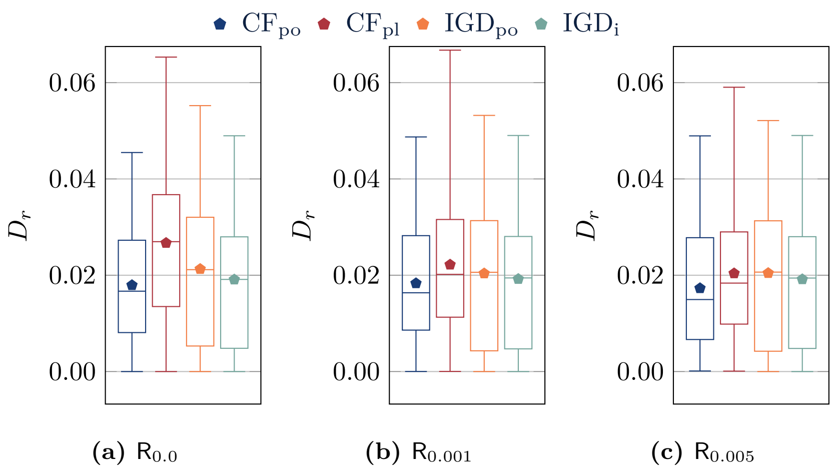


Figure 4.16: *Box-and-whisker plot for the rotation error per registration method for datasets (a)* R$_{0.0}$*, (b)* R$_{0.001}$*, and (c)* R$_{0.005}$*. The maximum rotation errors of CF$_{pl}$ are too big to be shown for* R$_{0.0}$ *and* R$_{0.001}$*. The missing values are* 0.065 31 *and* 0.066 74*, respectively.*

translation and rotation error are significant, with one exception. On dataset R$_{0.005}$ the rotation is not a significant factor w.r.t. the translation difference if the method CF$_{pl}$ is used.

4.3 Discussion

Unfortunately all registration algorithms changed the morphology of a number of model fragments for every dataset. The fact that this happened for all registration methods on all datasets suggests that it is due to a property of the registration method that is shared between all four methods. The changed morphology indicates that at least one of the applied transformation matrices was not linear. One possible reason for this are under and overflow errors, due to Unity being limited to single precision floating point numbers. This hypothesis is further supported by the fact that non-linear transformation matrices did not occur for the clinical data, which have a different scale. The fragments in the clinical dataset are between a factor 7 and 70 bigger than those in the simulated dataset. Consequently, scaling the simulated fragments could be an easy solution to the problem.

One thing that stood out in Section 4.2.1 was that the introduced rotation errors were worse when $\mathscr{X}$ was moved into $\mathscr{Y}$, i.e. when $s < 1$, for all methods except the IGD$_{i}$ registration method. This shows that the addition of the intersection term is valuable when the fragments intersect. This indicates that our adapted correspondence finding method does not completely solve the problems that occur when registering two intersecting shapes, as otherwise the performance of the registration methods would not have been influenced by the sign of s. The lack of significant difference in introduced rotation errors between methods for $s = 0.95$ is caused by the jump in performance by IGD$_{po}$. We have no explanation for this, or the lack of significant difference at $s = 1.15$.

One striking effect we observed for dataset O when considering the translation error as a function of the translation vector in Figure 4.12 was the increase in $\overline{D}_t$ that occurred when $\mathbf{w}_x$ was fixed and the other components of $\mathbf{w}$ were increased. One likely explanation is that the length of the offset vector matters. However, comparing offset vectors of the same length in Figure 4.12 shows that vectors of differing length have errors of the same order of magnitude.

For dataset S and O the rotation error for the initial alignment of the pairs was zero. Most methods introduced some rotation error. The worst rotation errors were introduced by the closed-form methods, of those two methods $\mathrm{CF}_{\mathrm{pl}}$ performed the worst. The IGD methods hardly introduced any rotation errors. The addition of the intersection term clearly helped, since the method $\mathrm{IGD}_{\mathrm{i}}$ outperformed $\mathrm{IGD}_{\mathrm{po}}$ in this aspect. Furthermore, the rotation errors introduced by this method were the least sensitive to noise. The other methods, especially $\mathrm{CF}_{\mathrm{pl}}$, introduced significantly smaller rotation errors on dataset S and O when more noise was added.

One strange effect that occurred for all datasets and all registration methods is that their performance w.r.t. the translation and rotation error did not deteriorate if noise was added. For some methods the addition of noise even improved their results. We expect that this addition of noise positively influences the correspondence finding process.

Finally, in general we have found that the method $\mathrm{CF}_{\mathrm{pl}}$ works better if some noise is added to the data. However even if noise is added this method generally performs worse than the other methods. This indicates that the assumption that the transformation is near linear does not hold for these simulated data.

The other closed-form method, $\mathrm{CF}_{\mathrm{po}}$, was outperformed by the IGD methods on all three datasets w.r.t. both the translation and rotation error, except for dataset $\mathsf{R}_{0.005}$. On this dataset the method $\mathrm{CF}_{\mathrm{po}}$ outperformed the methods $\mathrm{IGD}_{\mathrm{po}}$ ($t_{3926} = 5.22$, $p < 0.001$) and $\mathrm{IGD}_{\mathrm{i}}$ ($t_{2323} = -7.64$, $p < 0.001$), if we consider the mean performance. In spite of their higher average error, the IGD methods had lower rotation errors than the method $\mathrm{CF}_{\mathrm{po}}$.

Within the iterative gradient descent methods, the method $\mathrm{IGD}_{\mathrm{i}}$ outperformed the method $\mathrm{IGD}_{\mathrm{po}}$ on all three datasets, for all three noise levels in terms of the rotation error. The translation errors of the two IGD methods did not differ significantly, except for datasets $\mathsf{R}_{0.0}$, $\mathsf{R}_{0.001}$, and $\mathsf{R}_{0.005}$. On these datasets the method $\mathrm{IGD}_{\mathrm{i}}$ performed significantly better than other IGD method. As the standard deviation of the added noise increased, the difference in performance between these two methods increased as well.

4.4 Conclusion

Strangely the performance of the different registration methods did not suffer due to the addition of noise, instead some of them even improved their performance w.r.t. the translation and rotation error.

In general, the IGD methods performed best on the simulated data. The performance of the general IGD method, IGD_{po}, can be further improved by the addition of a term that punishes intersections.

4.5 Future Work

The strange effect of noise on the performance of the different registration algorithms in the simulated datasets should be investigated further. One approach would be to store the correspondences found by registration methods whose quality can then be compared between different levels of noise. However this requires the definition of a metric for the quality of a correspondence, which to the best of our knowledge is not available.

Future research should look into solving the problem of the changed morphology, and verify that numerical accuracy is indeed the cause.

We did not find any strong relations between the applied offset and rotation and the performance of the registration methods. However, our results clearly show that the relative rotation and offset of the fragments influence the registration algorithm. Further research should look into finding the maximum rotation and translation distance that the registration methods can handle.

References

[1] B. Foundation. *Blender*. Version 2.7.9. July 5, 2018. URL: `https://www.blender.org`.

[2] D. Q. Huynh. “Metrics for 3D rotations: Comparison and Analysis.” In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164. DOI: `10.1007/s10851-009-0161-2`.