GEOMETRIC REASONING IN HUMAN TOOL USE

by

Adrian Ionita (ID: 1057404)

Supervisor: Dietmar Heinke

Submitted in partial fulfilment

of the requirements for the degree of

Master of Science

School of Psychology

University of Birmingham

Birmingham, UK

September 2016

Abstract

Tool use is a fundamental human ability that distinguishes man from other species. This project investigates a potential building block for a computational model of tool use reasoning. The building block focuses on solving geometric constraints for interactions between tools and recipient objects.
Two approaches are considered:

1. An exhaustive approach using a physics engine to find compatible configurations of tool-object interaction by searching through of all possible alternatives.
2. A heuristic model which reduces the search space by using visual cues of tool and object parts.

The heuristic method employs a novel surface similarity technique which is able to assess parts having complementary geometries. Both exhaustive and heuristic models have potential for simulating human behaviour, but require further development and comparison with human behaviour.

Acknowledgements

I would like to thank Dietmar Heinke, project supervisor, for his guidance, support and making tool-use a very interesting topic to work on. Thanks also go to François Osiurak, the author of the four constraints theory, for his availability to clarifying concepts from his work. Last but not least, I would thank Rhiannon Davies for being the first subjected to proof-reading my work and providing constructive criticism.

Contents

List of Figures

Introduction

Tool use is one of the abilities that uniquely distinguishes man from other species. We refer to tools as hand held devices employed in making changes to the surrounding environment. Cases of tool use have been reported in other species, but never to the extent engaged by humans (Boysen & Himes, 1999; Harrington, 2009; Lefebvre, Reader, & Sol, 2004). As a defining attribute of human cognition, we wish to lay the foundations for a computational model of tool use reasoning.

Our model stems from the architectural framework defined by the four constraints theory (4CT) (Osiurak, 2014a). In the following we describe 4CT and its implications to the project.

The Four Constraints Theory

4CT defines tool use behaviour of healthy individuals based on empirical investigations of apraxia. Apraxia is a neurological disorder impairing a person's ability to plan and execute sequences of movements. The term covers a multitude of symptoms and levels of severity such as dyspraxia, ideomotor aparaxia, apraxia of speech. The underlying cause, however, is physical damage to the left hemisphere of the brain (Osiurak, Jarry, Lesourd, Baumard, & Gall, 2013).

Unlike competing theories, 4CT distinguishes between conceptual and production systems when describing tool use (figure 1). The production system dictates movement by transforming a person's intent into motor control while the conceptual system encapsulates the cognitive reasoning necessary for tool use. The dissertation focused on a computational model for the latter.

4CT characterises tool use situations as problem solving tasks requiring strong cognitive abilities. The four constraints of mechanics, space, time, and effort are dimensions within which problems are defined. Aiding problem solving are four cognitive processes: technical reasoning, semantic reasoning, working memory, and simulation based decision making (figure 1). Even a simple scenario like slicing bread is a reasoning task about the knife's length, sharpness, serration and the movements

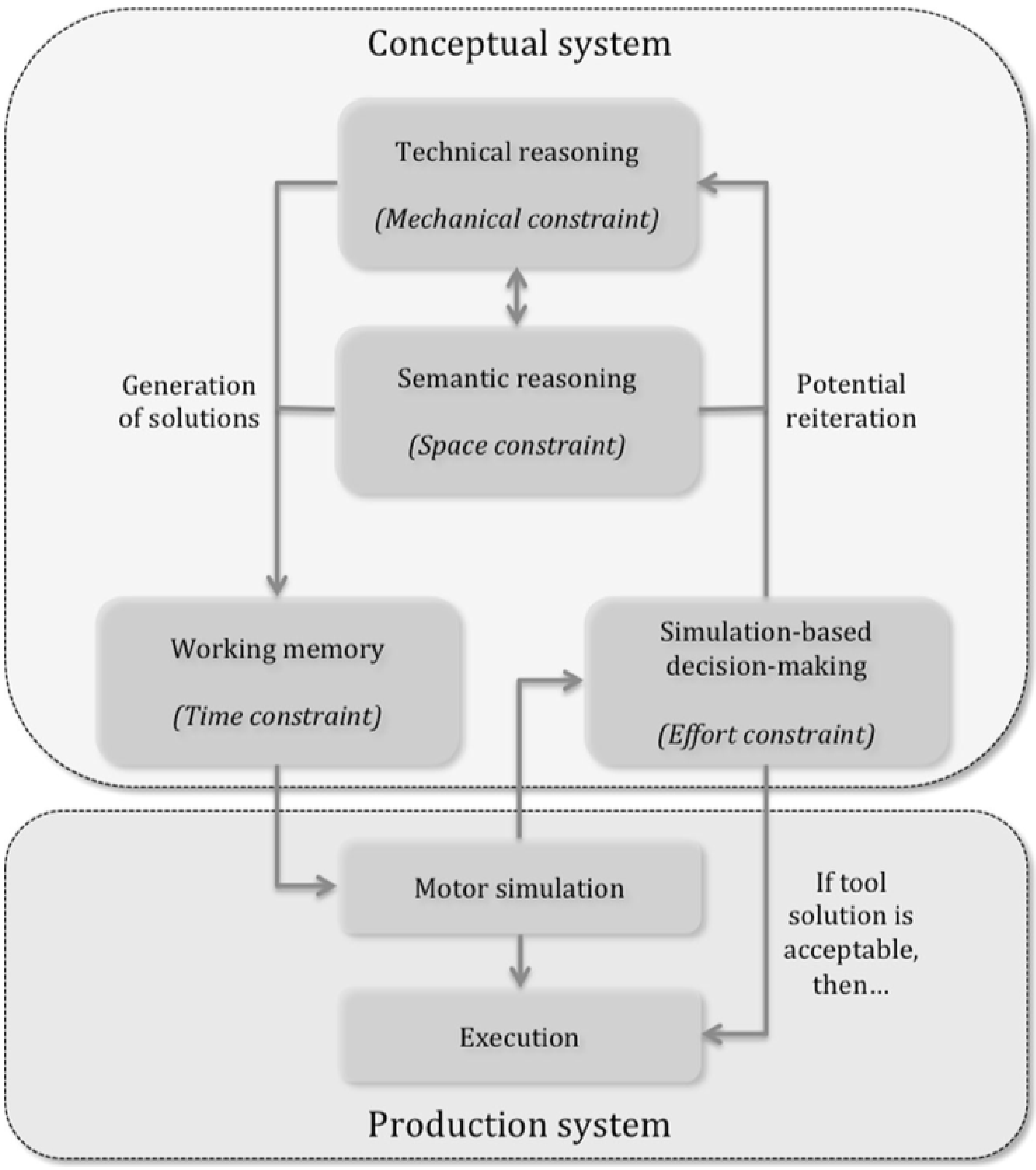


Figure 1. 4CT architecture reprinted from Osiurak (2014a)

necessary to manifest cutting effects.

Problem solving becomes more apparent in the absence of familiar tools, when subjects have to repurpose tools or fashion new ones. A distinction is made between novel tool use and familiar tool use. Semantic reasoning refers to situations when subjects are accustomed to a tool's common purpose. For example, a serrated knife can be used for cutting bread. These associations are made on prior experience without considering the knife's physical properties. In novel cases, however, technical reasoning is the inference of how the tool's properties can solve problems. For example, in the absence of a bread knife a saw would be a better cutting device than a table knife. Technical reasoning is a more general process for solving problems, although more cognitively involved.

The effort constraint and simulation based decision making refer to tool use energy cost. In nature, a simple rule of survival is that actions should require less energy than the rewards gained (Proffitt, 2006). Perceived effort would explain a user's preference for one tool use method over another. Even with multiple tools available, differences in physical properties can lead to differences in effort and dictate choices made.

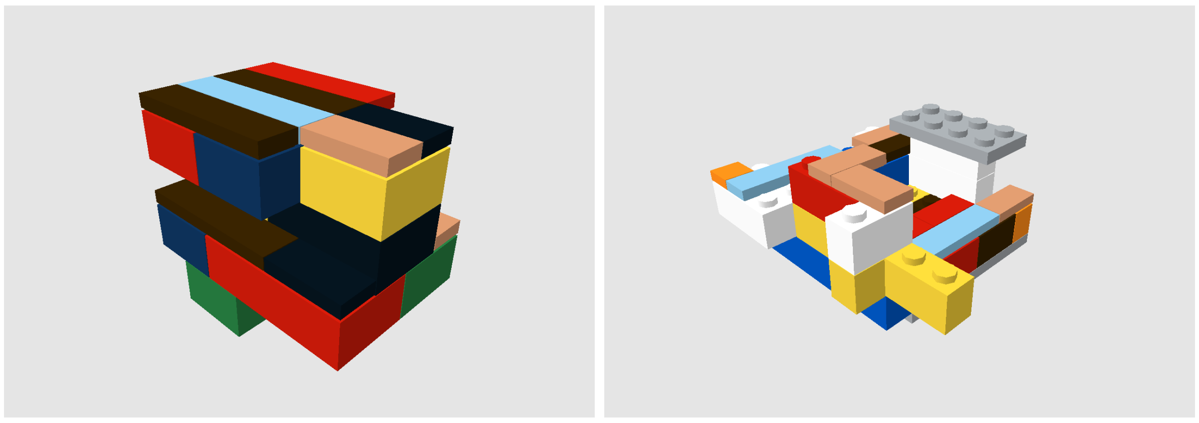
The working memory process (figure 1) is required when multiple steps must be fulfilled to achieve a goal. It describes a subject's ability to split activities into sub-goals and hold them in memory. Complex tasks such as fixing a broken radio would involve multiple steps and tool selection, however, such complex objectives are beyond the scope of this project.

A computational model should initially solve simple tool-use problems before complex ones. Our focus is therefore on single step tool and object interaction involving generic situations. These situations would be best solved through technical reasoning due to its general problem solving abilities. However, 4CT does not explain more complex reasoning than about simple object properties such as length, sharpness, abrasiveness. Crucial factors of forces and geometric constraints are simply abstracted as mechanical knowledge. A computational model would therefore have to elaborate these items before fitting into the larger 4CT framework.

The topic of this paper is solving geometric constraints through spatial reasoning in order to achieve simple tool-object interaction.

Experimental Setup

Dietmar Heinke and François Osiurak,the author of 4CT, have devised an experiment that tests human ability to use novel objects. The experiment is intended for use in both a computational model and to investigate human reasoning on tool and object geometries. Human trials were run in parallel to this project but do not constitute part of it. Nevertheless, this paper will refer to behaviour observed in these trials even though data has not yet been published.



(a) Passive object (b) Active object

Figure 2. Pair of novel tool and object models

In the experiment a subject is presented two novel objects built with LEGO blocks. One item is introduced to participants as "passive object" and may not be directly touched (figure 2a). The second item is introduced as an " active object" and must be used to lift the passive object to a new location (figure 2b). Note that the active and passive objects are analogous to tool and target objects respectively. Due to the unusual geometries, participants must reason about how to complementarily match the tool and object shapes.

A computational model mimicking human behaviour, would solve geometric constraints similarly to human subjects. Formally solving the experimental task requires consideration of force closure - object motion is controlled by tool contact forces. To shortcut the computing of forces and physical interaction one can use a physics engine. To some extent, it is believed that the human brain computes approximate Newtonian laws (Battaglia, Hamrick, & Tenenbaum, 2013). Similarly to humans, a physics engine does not compute exact friction, closure or gravitational forces, but provides realistic approximate results.

A tool use model can employ physics engines for multiple reasons:

1. The engine can serve as a source of inference over physical laws and objects

2. Solutions of tool-object interactions can be verified through simulation
3. Physical simulation is analogous to mental tool simulation from 4CT

Physics Engine Evaluation

A suitable physics engine would satisfy the following criteria:

1. The engine should have high fidelity in simulating rigid body collision. That is, the engine should faithfully reflect real world interaction of bodies involving no moving parts or deformations. As tool and object are composed of LEGO, items are best represented as rigid bodies. High fidelity is required to find realistic tool-object fitting positions, with minimal errors.
2. The engine should be available in user-friendly scientific languages (such as MATLAB or Python). Scientific languages have inbuilt algorithms that can be utilised.
3. Engine use should be inexpensive.
4. The engine should be portable to common platforms (i.e. Windows, Linux, MacOS) to encourage engagement from future researchers.
5. The engine should have good documentation for ease of development.
6. Simulations must be done on three dimensional(3D) models of tools and objects. Engines capable of only two dimensional(2D) simulations are therefore inappropriate.

In an academic context, physics engines have been used for robotic simulation of locomotion and object grasping. The fidelity of these scenarios match our experimental requirements and can serve as a basis for engine evaluation. Literature reviews have focused on commercial or open source engines dedicated to game development (Boeing & Bräunl, 2007; Hummel, Wolff, Stein, Gerndt, & Kuhlen, 2012; Roennau, Sutter, Heppner, Oberlaender, & Dillmann, 2013). Engines aimed at scientific simulation unfortunately require large usage fees (e.g. Vortex by CM-Labs and MuJoCo). Due to their target audience, game engines achieve high speed simulations at the cost of precision. There is compromise between speed and how faithfully an engine can replicate real physical interaction. For this reason, most available engines are not suitable for scientific use.

With engines undergoing constant improvement, evaluation is transient and specific to product versions. Review articles fail to specify the version of the engines evaluated. Additionally, performance results are influenced by implementation specifics. For example, reviewers can use the engine directly or through an abstraction layer that can affect performance. We therefore select the most likely engines for model use, and evaluate individually.

Two game engines emerged as candidates: *Bullet Physics* v2 and *Newton Game Dynamics* v3. A formerly popular option is Open Dynamics Engine, but should not be considered as it is no longer under active development.

Bullet Physics. Boeing and Bräunl (2007) regards Bullet as having the best overall performance. Hummel et al. (2012), Roennau et al. (2013) also mention Bullet as having the best rigid body collision, but only in the case of large object simulation. What defines objects as large is however unclear.

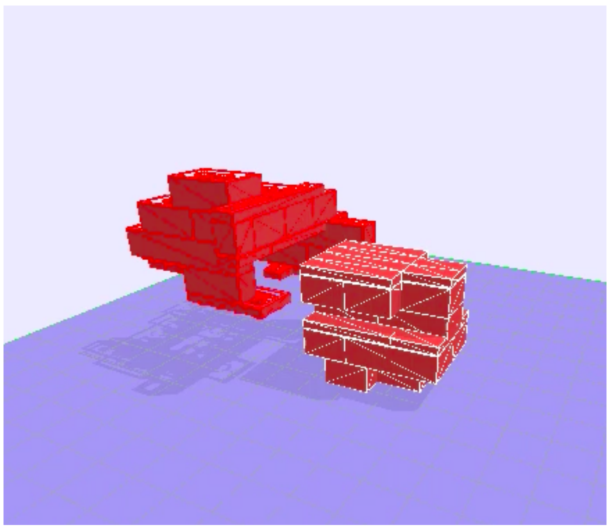


Figure 3. Tool and object models loaded into Bullet simulation demo

We test Bullet's collision detection in a demo containing simple tool and object models (figure 3). The passive object is first set on a flat surface. The tool is then positioned to slide into the object and lift it. Demo code is therefore representative of the experimental setup.

Several parameters can influence the effects of interaction between the two objects:

1. Object collision properties are more commonly represented by classes which approximate the object's shape in order to gain execution speed. A more precise approach is to use classes that represent true shape properties (figure 4).

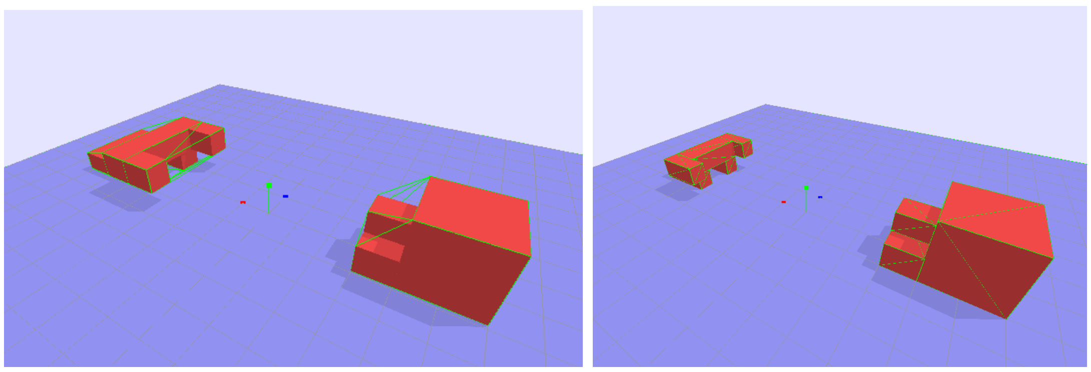


Figure 4. Comparison of approximated collision and exact collision shape represented as green lines

2. A threshold or margin of contact exists between objects. When the margin is low, shaking effects can be noticed when objects make contact. Alternatively, when the margin is large, the object's collision would not be representative as gaps would be too small to allow fitting.

3. Moving the tool to interact with the object can be done multiple ways with variable degrees of realism.

From a development perspective, the engine is easy to compile and has detailed documentation. MATLAB integration is available from third party developers, but is not well maintained, lacking functionality or failing compilation. A model using Bullet Physics would therefore have to be developed in C++.

Overall Bullet Physics suffers from a high degree of shaking when tool and object interact. When the tool is positioned to lift the object, collision points make the object bounce uncontrollably within the tool's grasp. As bounce forces amplify the object eventually bursts out of the tool's fixation. Engine parameters can be tuned to minimise these effects, but would ultimately lead to less representative simulations.

Newton Game Dynamics. Newton Game Dynamics is regarded as having better precision than Bullet, but at the cost of performance (Hummel et al., 2012). The precision gain is achieved by using a deterministic solver instead of numerical methods for integrating movement and forces over time.

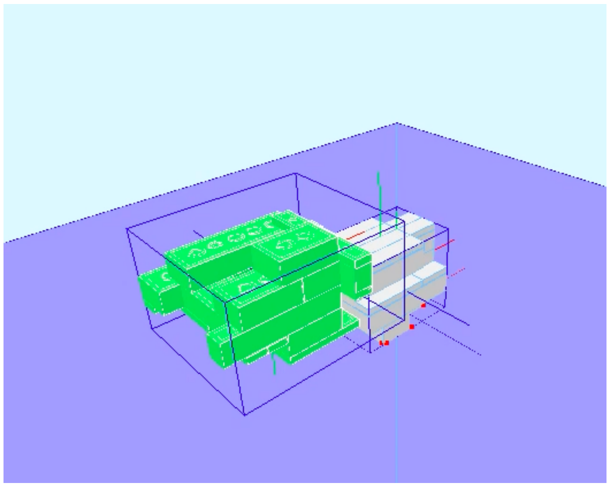


Figure 5. Tool and object models loaded into Bullet simulation demo

We setup a demo similar to Bullet. The object is set on a flat surface. The tool is then slid into the object's fitting and lifted (figure 5).

Compared to Bullet, Newton's default behaviour exhibits realistic behaviour. The simulation shows no shaking or unrealistic dynamics. The tool and object move smoothly on established trajectories.

From a development point of view, Newton lacks the quality of documentation found in Bullet. There are thousands of lines of sample code available, but locating needed functionality is difficult. This is made worse by a strange, yet consistent API interaction.

Additionally, the engine supports unconventional 3D file formats, making it difficult to load object models. Code has to be developed to support common formats.

As a less popular engine, Newton does not have integration with other languages (e.g. MATLAB or Python). A computational model would have to be written in C++.

Even with the above development concerns, the engine's simulation precision make it a prime candidate for our model.

Exhaustive Search

Demos show how tool use can be verified through simulation. As the model aims to solve geometric constraints, one can position items in all possible configurations and select valid ones. Validity implies that the tool can lift the object, and that models do not penetrate each other's surfaces. A single configuration represents a tool's rotation and relative position to the object. That is X,Y,Z coordinates and rotations around axes (i.e. yaw, pitch, roll). The *Exhaustive Search* model finds solutions by testing all possible configurations.

Implementation

The target object is positioned on a flat surface. The tool is then placed in successive relative locations, and object collision is tested. For any collision points to be detected, one physics simulation time step has to be executed. Newton Physics allows obtaining the penetration depth of collision points. When penetration is too high, the configuration is deemed invalid as objects unrealistically penetrate each other's bodies.

If penetration depth is zero, the object and tool may not be in contact at all. Such scenarios can be removed by applying a lifting force to the tool. When the objects are in contact, lifting velocity would transfer to the target object and can be asserted.

The approach is able to verify a fitting configuration for a single physics engine update step. However, as object contact is detected through a small penetration margin it is possible that undesired vertical forces appear. Output data may therefore contain invalid solutions which require further validation.

The physics engine does not require 3D rendering when computing interaction. Exhaustive searching can run in either graphical or non-graphical modes. Graphical rendering allows researchers to investigate the correct loading of objects and 3D world dynamics. The non-graphical model allows faster computation as rendering overburden is removed.

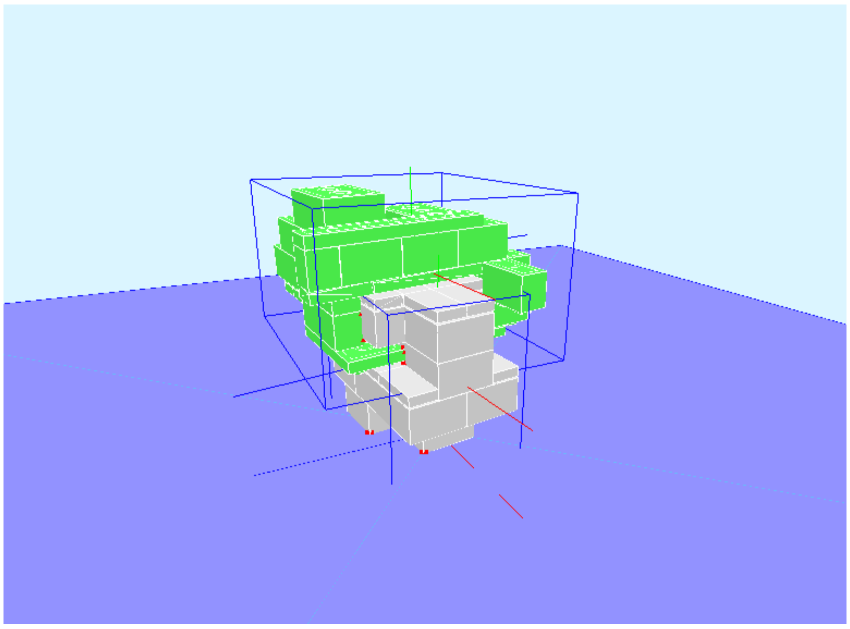


Figure 6. Exhaustive search rendering

Figure 6 captures the rendering of two objects in fitting position. Object bodies are displayed in green and grey. Collision meshes are outlined by white lines. Contact points are displayed as red dots. 3D orientation vectors give visual cues of rotation angles (red, green, blue for X, Y, Z respectively). Blue bounding boxes represent vertex extremities (i.e. min and max X, Y, Z coordinates).

Newton Dynamics has the in built ability to optimise meshes as convex decomposition. This means, complex assemblies of triangles representing simple geometries can be optimised to simpler mesh structures of the same shapes. A complex object would therefore be transformed to an assembly of simple shapes, optimising collision detection and computation. This feature may occasionally have to be tuned for pairs of tool-object models.

Discussion

When search intervals are in the vicinity of possible solutions, the model finds multiple equivalent fitting configurations. Equivalent solutions are found on the trajectory on which the tool slides into the object. However, on a single multi-core computer exhaustive searching is not feasible. The tool's position is determined by 6 degrees of freedom amounting to 10^{12} possible configurations. Even for a single tool rotation there are 10^8 configurations, requiring approximately an hour of execution.

In the implementation verifying configurations has been optimised for speed. Further improvements can be achieved by splitting the search space over multiple computers. Alternatively, the search step can be increased at the expense of finding possible solutions.

Exhaustively searching is lengthy and computationally expensive. Engine features, such as penetration depth, can be utilised in more ingenious search algorithms. However, solutions based solely on the engine's functionality remain problematically unrealistic representations of physical laws and human ability. For example, the tool can be positioned to lift an object even when no physical manner would allow it to be placed in that location.

In simulations lifting forces are applied to the tool's center of gravity. This is a strong simplification of the problem as the search may produce solutions that do not realistically permit the tool to be held.

Computational expense remains the impeding factor to the model's use. Heuristic methods can be used to optimise the search space whilst considering human ability. The following describes a visual search approach to finding possible fitting configurations. The physics simulations remain a method for testing such configurations.

Heuristics Search

In optimising the search for tool and object configurations, one could determine the regions of contact that are likely locations for good interaction. Human subjects show an understanding of the forces generated by their actions and of the geometric constraints of the physical world. For example, subjects performing our tool fitting tasks, examine the geometry of objects and stop at likely positions before attempting interaction.

Tool and object configurations, bearing high likelihood of success, would satisfy two criteria:

1. **Geometric constraints** of the two objects must be met. That means, the shapes of the two objects should adequately fit.
2. **Contact forces** generated must correspond to the intention of the actions executed. For a lifting action, the human subject would focus on points of contact that permit vertical forces to be applied.

The large number of possible tool and object configurations can be reduced by the above criteria. Time consuming simulations would therefore no longer be required (akin to mental simulation (Osiurak, 2014b)). As with human subjects, only potential solutions would be tested through the use of the physics simulation.

Geometric Constraints

The geometries of the tool and object must be compatible. In other words, either the active or passive object's geometry must fit within the gaps and edges of its counterpart. It is not necessary for the whole geometries to fit. If the tool's functional part match the gaps of the passive object, then the configuration is likely to achieve the user's intended effect.

Tighter fits, offer better transfer of energy and control over movement. A good fit would therefore require less manipulation effort. In the paradigm of 4CT (Osiurak, 2014b), the effort constraint would explain why users prefer one tool fitting solution over another. To find parts that provide tighter fits, we consider techniques of shape similarity in matching object parts.

Shape Description and Matching Techniques. Shape analysis techniques have traditionally been engaged in image processing and computer vision for detecting and tracking objects of similar features (Loncaric, 1998; Robert, Arthur, Istvan, & Sergiu, 2012; Zhang & Lu, 2004). The last two decades have provided many approaches for this type of problem (Loncaric, 1998; Robert et al., 2012; Veltkamp & Hagedoorn, 2001; Zhang & Lu, 2004). Fundamentally, shape analysis is used in detecting similarity even as objects undergo geometric transformations of rotation, translation, scaling, or shearing (figure 7).

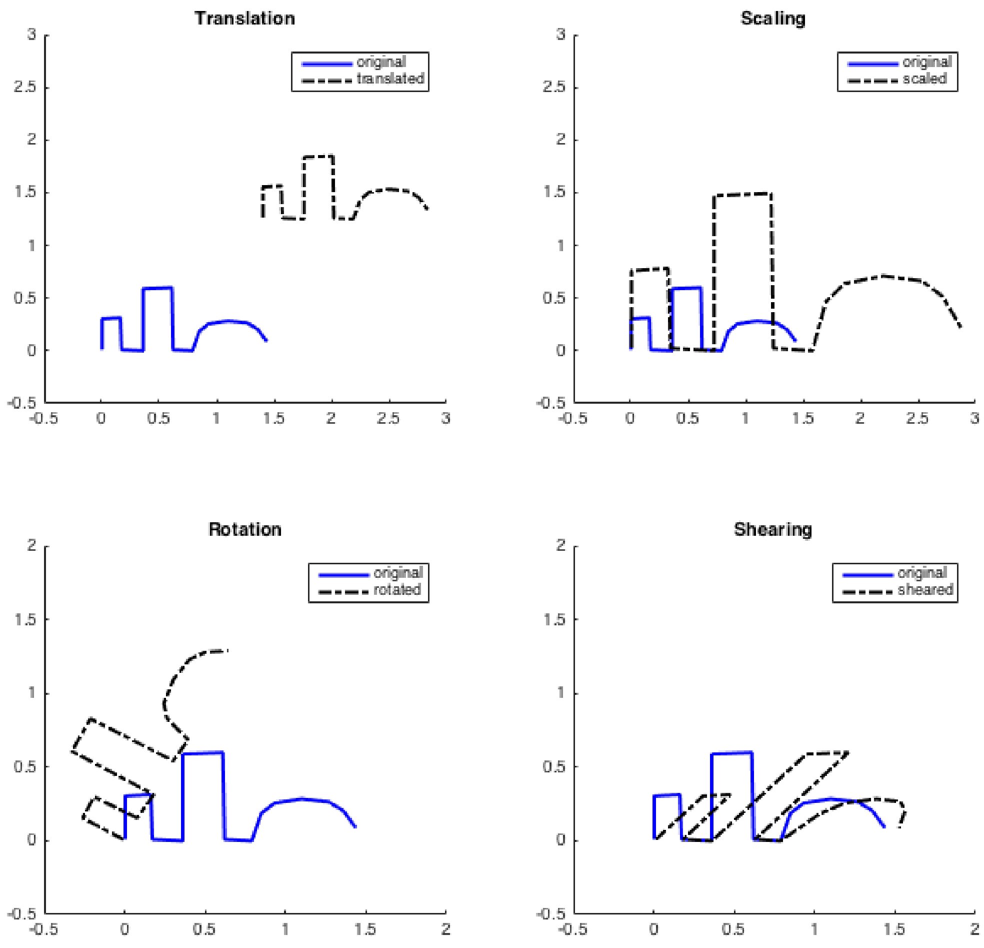


Figure 7. Graphical examples of affine transformations

Zhang and Lu (2004) distinguishes two types of techniques: contour-based and region-based. These are further divided into structural and global approaches. Figure 8 shows a comprehensive classification of prominent shape analysis techniques.

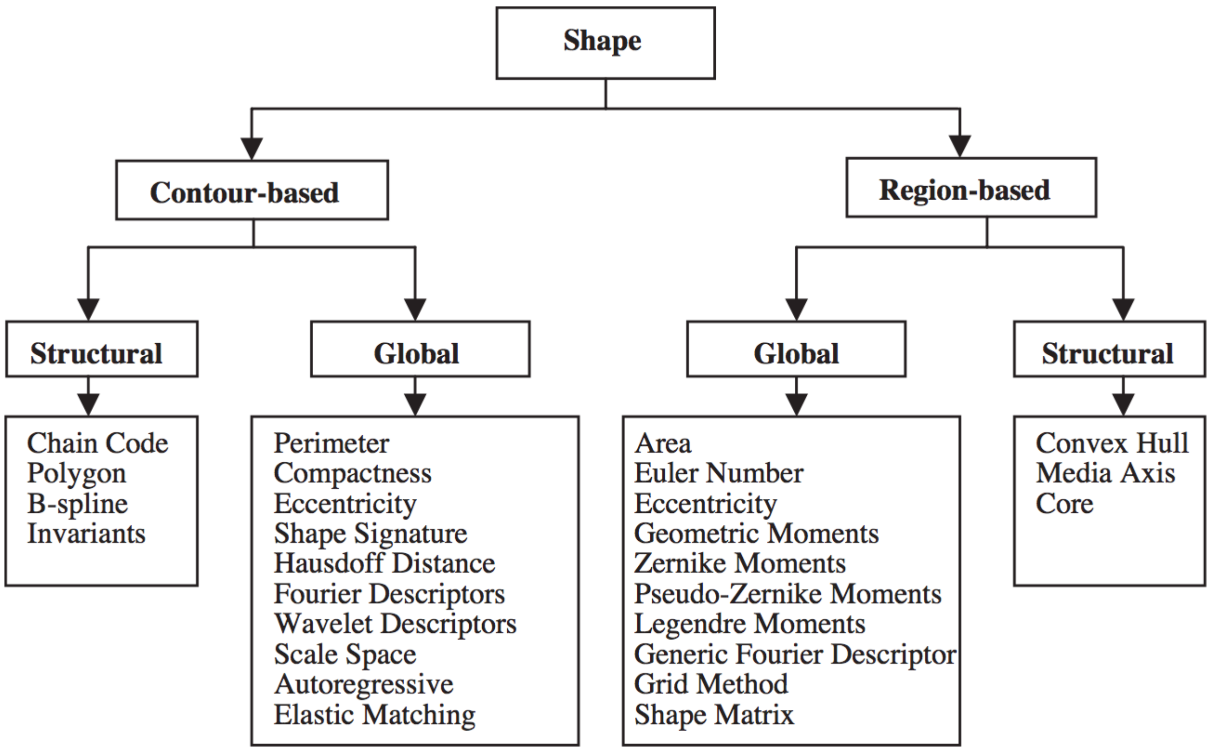


Figure 8. Classification of shape similarity techniques (reprinted from Zhang and Lu (2004))

Contour-based techniques assesses similarity by extracting features from the object's edge. In comparison, region techniques work by assessing surface level information such as: colour, gradient changes and surface medial. Techniques from both approaches have justification in human perception (Chatbri, Kameyama, & Kwan, 2016). Nonetheless, our context excludes region-based approaches as tool parts must correspond to object gaps. Gaps however do not contain surface information.

In structural approaches, shapes are considered to be composed out of primitives. For contour techniques these primitives are line segments on the boundary of an object. Primitives can be organised as either linear list of segments (Zhang & Lu, 2004), or as tree like structures that carry hierarchical information (Zhu, Zhao, & Zhu, 2015). In structural approaches, two objects are considered similar when they have the same primitive composition. As comparison, global approaches make use of shapes as a whole when assessing similarity.

Both structural and global approaches have justification in human perception (Zhang & Lu, 2004). Human subjects show a preference for features even when other shape descriptors are available (Chatbri et al., 2016). At the same time, global shape perception seems to precede local feature detection (Navon, 1977). For that reason, both global and structural approaches should be considered.

Correctly mimicking human behaviour would require more insight into human visual perception. Loncaric (1998) describes several theories of human perception, but with an emphasis on image processing. In a tool use scenario, emphasis should be given to theories describing perception as volumetric, as through the use of generalised cylinders (Dickinson & Biederman, 2014). Such theories may better explain human tool ability and can be explored in future work.

Limitations. As previously remarked, contour-based techniques contain promising approaches in solving tool-object fitting problems. It is important to consider some of the limitations of these techniques, especially with regards to human ability.

Most shape similarity techniques are tailored to analysing image based information. That is 2D projections of a 3D space. Human perception however carries depth related information. As tool use encompasses real world geometric constraints, any contour based approach would have to adapt to 3D information such as point clouds.

When solving tool use scenarios, it is important to consider a subject's point of view. The geometry of objects may hide functional parts until the object is rotated into proper perspective. Human perception is able to recognise objects from sparse information or occluded perspective (Loncaric, 1998), i.e. part of the object is hidden from view by some obstacle. Global contour matching techniques are however sensitive to these types of constraints. Structural approaches are better suited, as they identifying shapes from object's visible parts. A good contour matching approach would therefore work on object parts rather than globally evaluating similarity.

In our tool use experiment, human subjects were observed to attempt object interaction even when parts did not fully match. It is therefore possible that regions of interest are assessed probabilistically. When two parts are sufficiently similar, they are worth further investigation. A suitable contour matching algorithm would therefore have to evaluate the degree of similarity between two shapes in a uniform manner.

We next propose a novel approach for assessing object contour similarity. The approach evaluates shapes within the subject's point of view, works on 3D information, and is able to measure the degree of similarity of two parts.

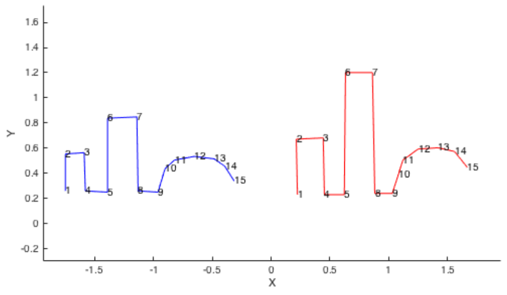
Novel Technique For Surface Similarity

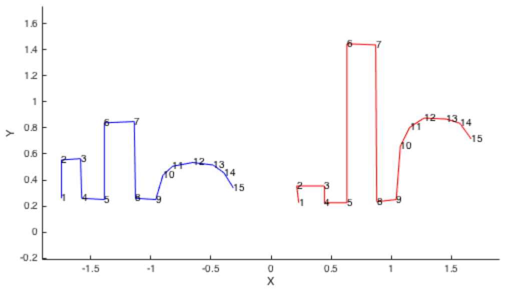
When applied to two dimensional data, contour matching is the problem of assessing similarity of lines. For three dimensional data, the task is better described as assessing similarity of surfaces. For clarity, the phrase 'surface similarity' will imply the application of the technique for either of the above cases.

The proposed technique can be described as a general approach for assessing data similarity. It is based on principal component analysis (PCA) and Pearson's correlation coefficient. We first describe implementation for 2D data and later discuss the 3D case with implications for tool use.

2D Similarity. For our novel technique we assumes that points taken over similar surfaces change value at the same rate and in the same direction. For example, consider two lines as in figure 9a. Pairs of points taken from similar locations on each line change Y axis value at the same time. If point number 2 on the red line increases, so does the corresponding point on the blue line. More so, compared to figure 9b, the first set of lines change value at the same rate compared to their overall Y-axis deviation. A human observer would assess lines in figure 9a to be more similar than lines in 9b, which are more similar than in figure 9c.

Lines with complex curvatures can be expressed through series of points in Cartesian space. Affine transformations of scaling(1) and translation(2) produce new points defining lines who's coordinated are linearly related to the original. Pearson's


(a) Same direction & rate (different scales)


(b) Same direction but different rates

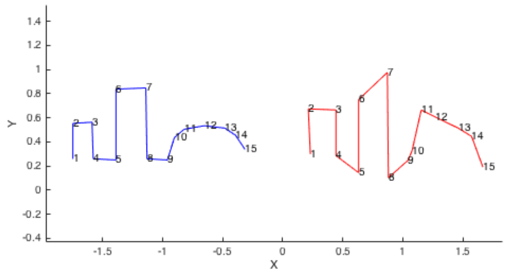

(c) Different directions

Figure 9. Lines of different similarity. From (a) to (c), shown in decreasing similarity.

correlation coefficient measures the extent to which two variables (x_i and x'_i) can be defined through a linear equation ($x'_i = a * x_i + b$).

Correlation would therefore return high similarity if two lines are linear transformations of each other.

$$\begin{bmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} a_x * x_i \\ a_y * y_i \\ 1 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i + b_x \\ y_i + b_y \\ 1 \end{bmatrix} \tag{2}$$

In the case of rotation(3) or shearing(4), the linear relationship between points is lost. Other transformations are combinations of the above or deform shapes through distortion (e.g. barrel, pin cushion transformations) . For tool use scenarios involving rigid bodies only transition, scaling and rotation are realistic geometric transformations.

Rotation is a common occurrence, and can be handled through PCA.

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ -sin(\theta) & cost(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i * cos(\theta) - y_i * sin(\theta) \\ x_i * sin(\theta) + y_i * cos(\theta) \\ 1 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i + h * y_i \\ y_i \\ 1 \end{bmatrix} \tag{4}$$

PCA returns a set of unit vectors that indicate the direction in which data has most variation. The rotation angle of a line can be removed by aligning the shape's principal components to the main X-Y axis. As affine rotations preserve variation, the principal vectors rotate by the same angles that a figure is rotated (figure 10).

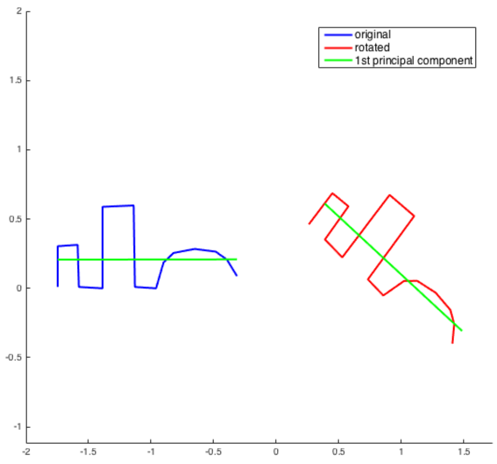


Figure 10. First principal component rotating with the figure

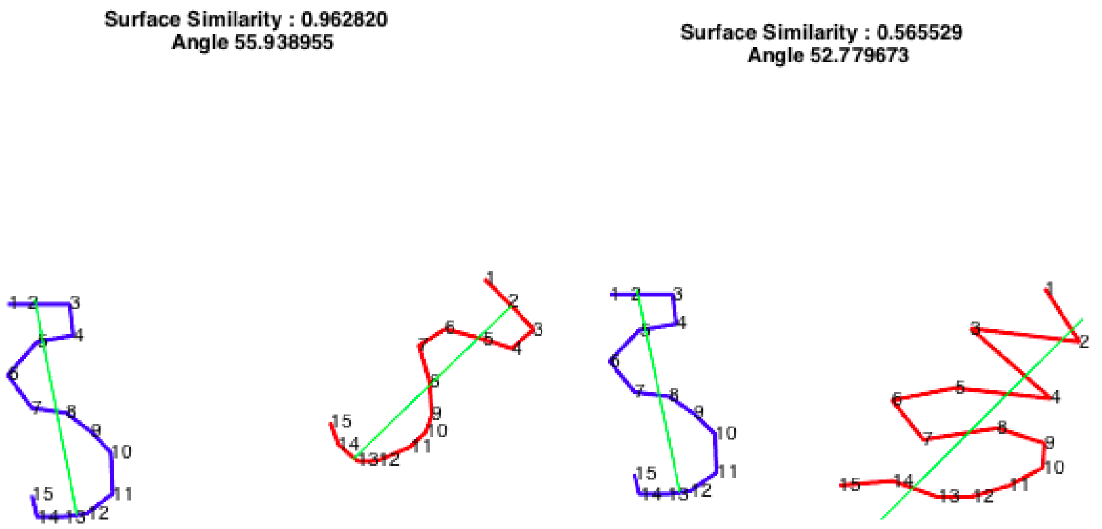
Since principal components of a set of data are orthogonal, in the case of 2D data points, it is sufficient to align just the first principal component to the X axis. Even in higher dimensionality of n components, it is sufficient to align just the first $n-1$ components.

A rotation angle(α) can be retrieved from:

$$\alpha = arctan(v_y, v_x)\ mod\ 2\pi$$

Where $\vec{v} = (v_x, v_y)$ is the first principal component vector.

Using α all points defining a figure can be rotated in order to perform correlation. Similarity is calculated as the product of correlations over the individual X-Y coordinates. Figures 11a and 11b, demonstrate the approach working on two hand drawn lines. Higher correlation is achieved for the more similar lines.



(a) Similar contours

(b) Dissimilar contours

Figure 11. Comparison of simalirty measure

Correlation values are in the range of $[-1, +1]$, where:

- values close to $+1$ represent high similarity
- values close to 0 represent dissimilarity
- values close to -1 represent high similarity but mirrored (e.g. points are taken in reverse order)

3D Similarity. The components of the approach allow it to be applied to pairs of surfaces instead of lines. As with 2D shapes, points in 3D Cartesian space (i.e. X-Y-Z coordinates) can be organised in a grid structure defining a surface mesh (figure 12).

The similarity algorithm steps remain the same as in the 2D case:

1. Using PCA calculate the principal component vectors of each surface

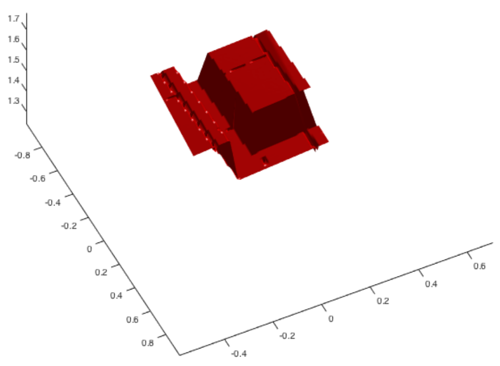


Figure 12. Example of surface mesh

2. Establish the rotation angles for aligning the first two principal component vectors to the X-Y axis
3. Rotate all point of the surfaces, using the established angles
4. Calculate the product of correlations over all dimensions (i.e. X-Y-Z)

In 3D space, calculating angles and applying rotations involves different calculations. Two angles(α, β) are required for aligning the first principal component vector against the X axis:

$$\alpha = arctan(v_y^1, v_x^1)\ mod\ 2\pi$$

$$\beta = -asin(v_z^1)\ mod\ 2\pi$$

Where $\vec{v}_1 = (v_x^1, v_y^1, v_z^1)$ is the direction vector of the first principal component.

The vector part of the second principal component must be rotated by α and β in order to extract a third angle(γ):

$$v^2 = rotY(-\beta) * rotZ(-\alpha) * v^2$$

$$\gamma = arctan(v_z^2, v_y^2)\ mod\ 2\pi$$

Where $\vec{v}_2 = (v_x^2, v_y^2, v_z^2)$ is the direction vector of the second principal component and $rotY(-\beta)$ and $rotZ(-\alpha)$ are rotation matrices around the Y and Z axis.

Surface points can then be aligned to the primary axes by multiplying them to the corresponding rotation matrices:

$$P = rotX(-\gamma) * rotY(-\beta) * rotZ(-\alpha) * P$$

Where P is the set of points defining the surface mesh.

Discussion

The surface similarity technique, formally described, possesses the properties of effective contour matching techniques. Translation and scaling invariance are immediate consequences of Pearson's correlation. Rotation invariance is attributed to properties of PCA, allowing shapes to be aligned to the same orientation. Other non-linear transformations, such as shearing, added noise, or distortion would affect similarity measure proportional to the amount of applied deformation. These attributes place the technique as a potent candidate for most computer vision tasks.

Surface simulation can not be regarded as a global matching technique. Its strength lies in assessing part similarity, although it has no mechanism for determining the structure of a given shape. If shapes are prior decomposed into parts, the technique can be used in assessing individual parts.

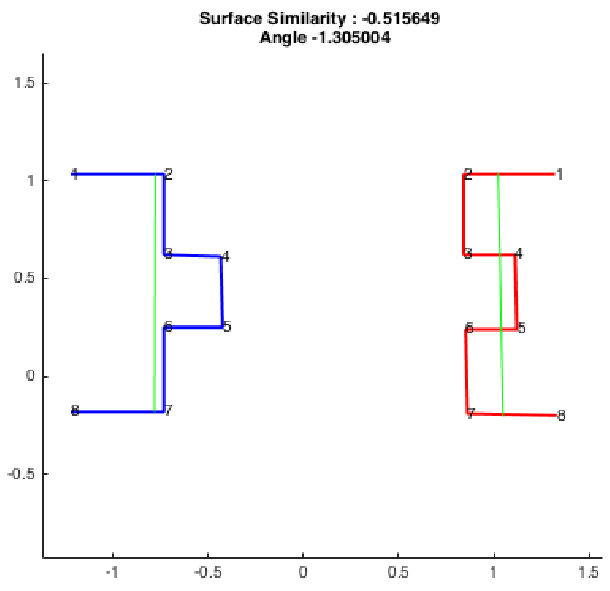


Figure 13. Complementary surfaces having negative similarity measure

For the purpose of tool fitting, surface similarity carries additional helpful properties. The nature of tool use requires attention to complementary features rather than similar. Negative similarity measures would indicate complementary surfaces (figure 13). Care should be given as this property depends on the size of the visual field. For example, in figure 13 if points 1 and 8 were removed, the surfaces would have a positive similarity value.

Tool and object interaction require that intended parts are properly aligned. Surface similarity can extract the necessary angle difference between complementary or similar shapes. However, PCA orthogonal vectors are sometimes reversed, the resulting angle differences can occasionally be incorrect by 180°.

Both complementarity and angle differences are highly desired traits for tool use problems.

In the case of tool use, the surface similarity algorithm can be computationally simplified. Human subjects physically rotate objects so that important features face their perspective (figure 14). As such, analysed surfaces would have already aligned to the same orientation. PCA rotations are therefore unnecessary in practice, and can lead to a simplified form of the algorithm.

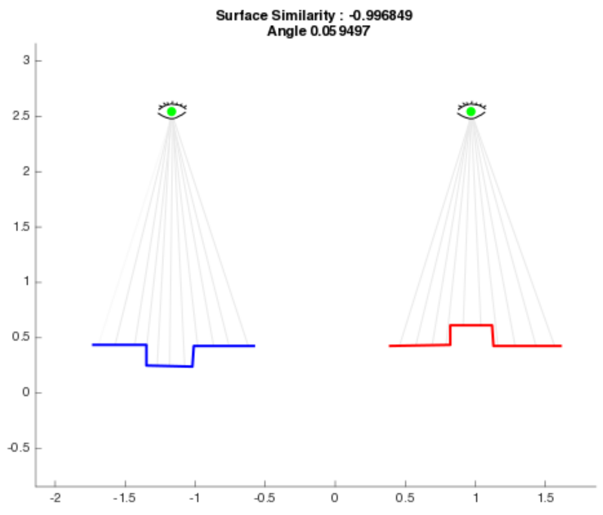


Figure 14. In practice, surfaces are already aligned to the user's point of view

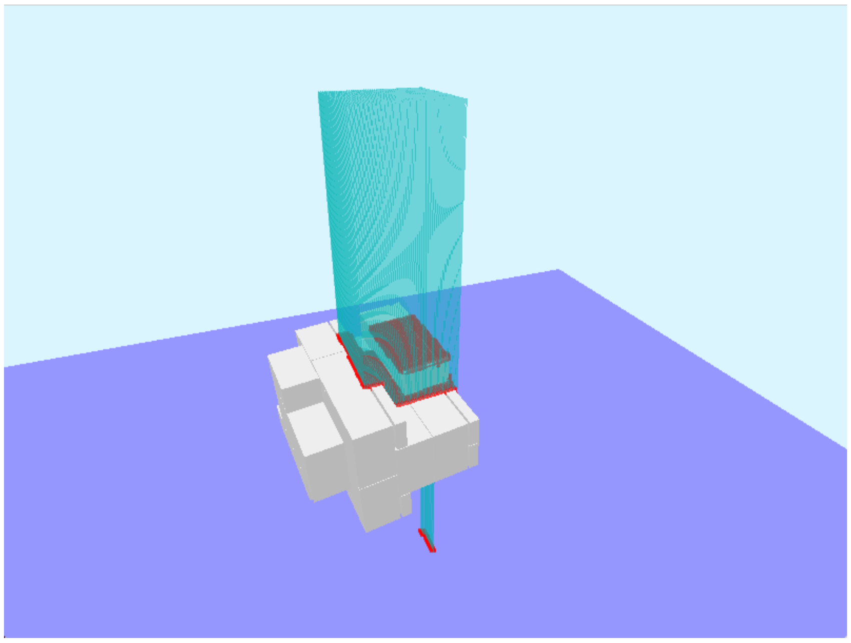


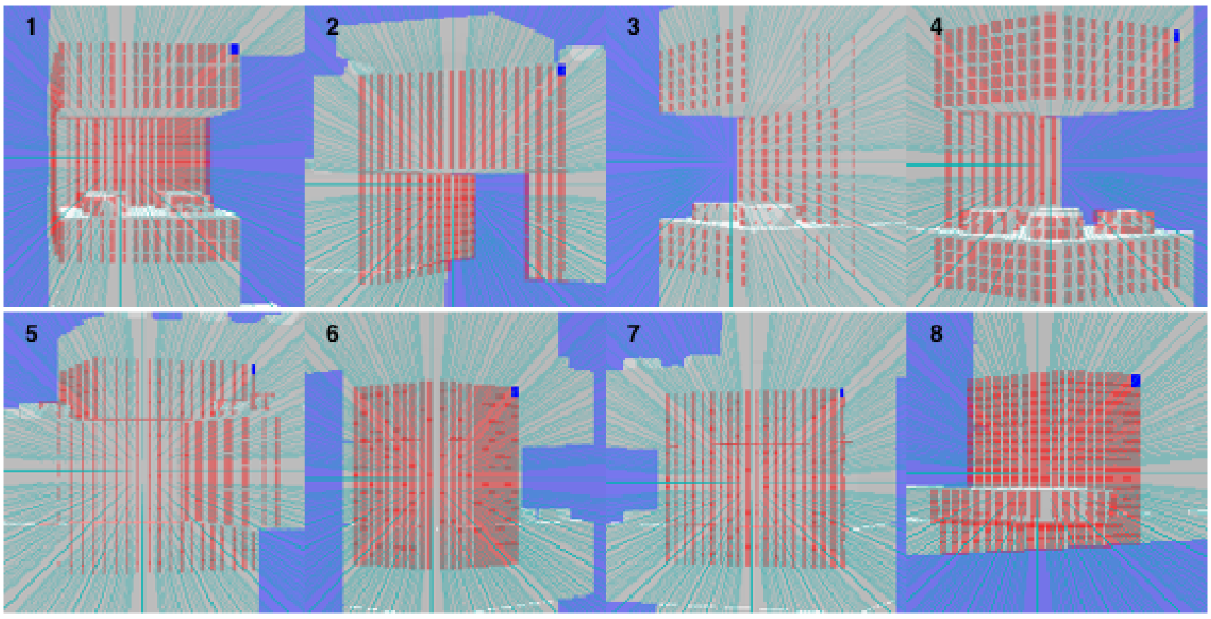
Figure 15. Ray casting to extract surface points

Evaluation

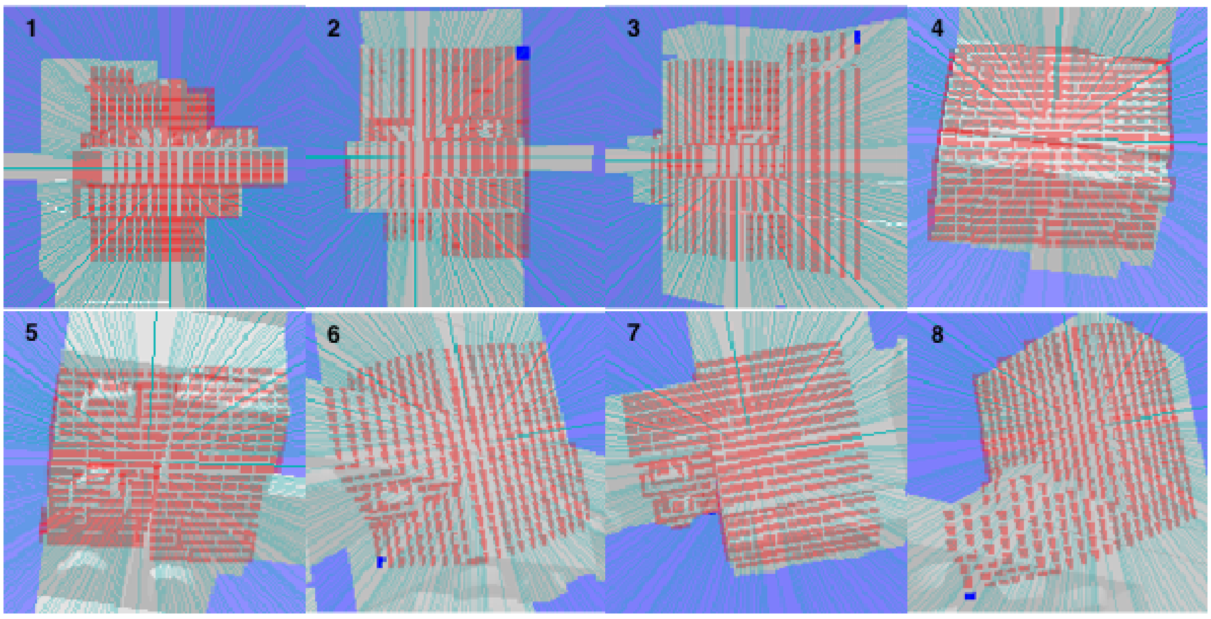
The effectiveness of surface similarity was assessed using a physics simulator for a tool use scenario. Human depth perception and perspective was simulated through ray casting, by extracting object 3D surfaces. Ray casting is a feature of physics engines which mimics the propagation of light. Surface are extracted by casting a grid of parallel rays over an object model as in figure 15).

A first test verifies the effectiveness of the algorithm and the impact of PCA to the similarity measures. 8 pairs of tool and object surfaces were extracted (figure 16a,16b). The first set of 4, were chosen to represent parts that are of interest to human subjects. A second set have no tool fitting value. Figure 17 shows a grid of correlation values for each combination of surfaces. In either the PCA or simplified form, the algorithm returns complementary matching for pairs in the first set of surfaces. Other combination achieve 0 or positive similarity, making them less possible tool fitting candidates. The results match intuition of object parts that would provide good fitting.

It is worth observing that a non-PCA algorithm tends to return more false positive matches. Yet, because of its reduced complexity, we employ the simplified algorithm in a heuristic visual search implementation.



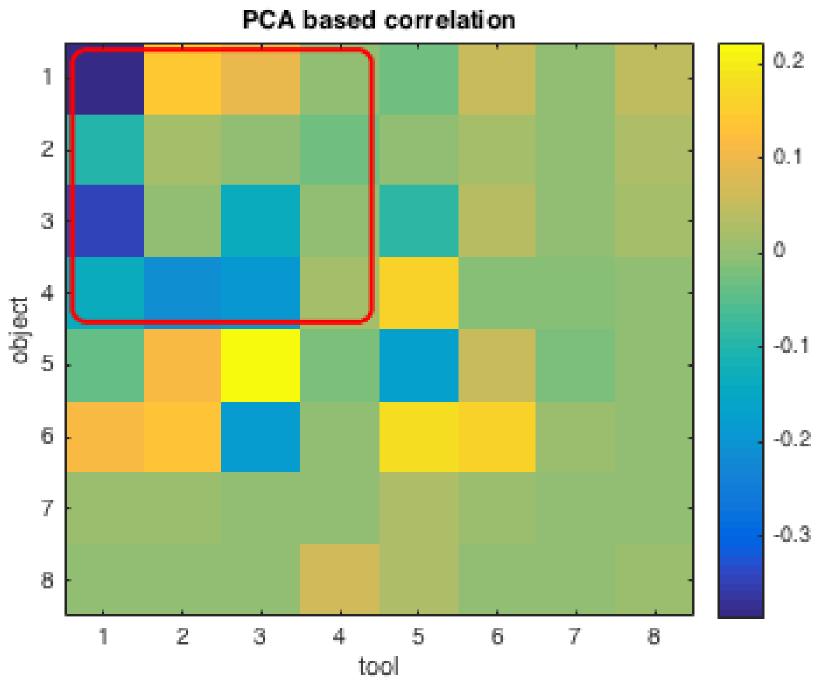
(a) Object surfaces



(b) Tool surfaces

Figure 16. Surfaces extracted from both tool and object

As previously mentioned,tool and object fitting can be optimised by visually detecting compatible parts. Visual search simulates the act of finding potential solutions inside the physics simulator. Given a tool and object, the search randomly selects surfaces to assesses complementarity. When a pair is found, the simulation pauses and lets a human assessor verify validity (figure 18).



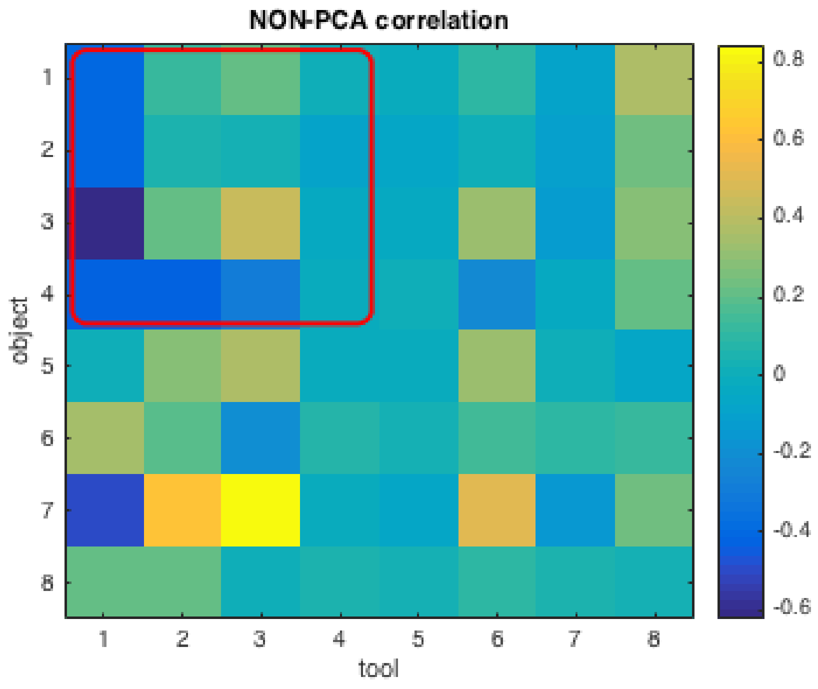


Figure 17. Complementary correlation occurs mostly between parts of interest

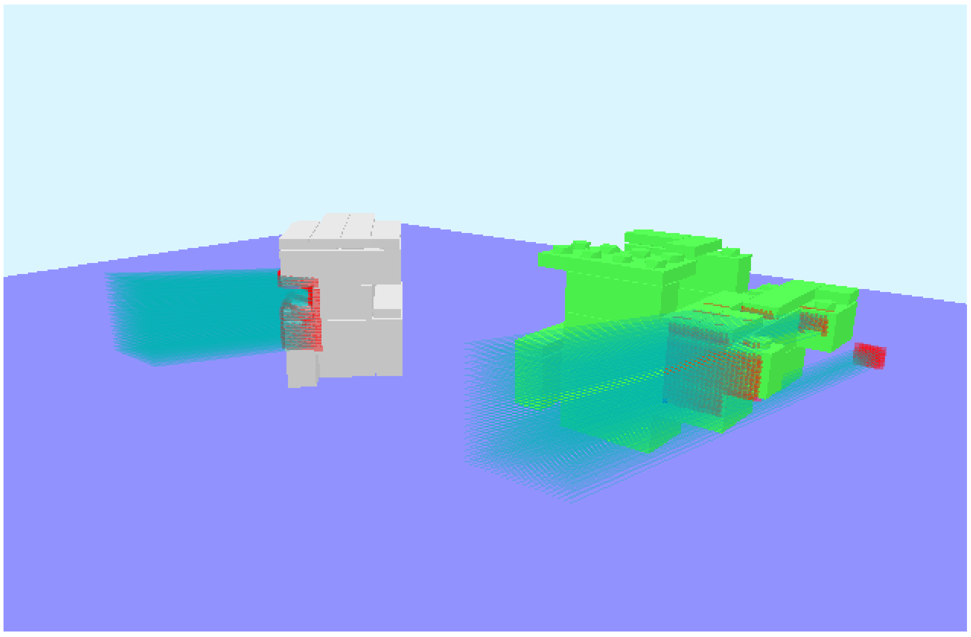


Figure 18. Complementary surfaces found by visual search

Due to project time constraints the effectiveness of visual search was not formally assessed. Future investigation will have to compare results to human data, potentially carrying eye tracking information.

Additionally, further improvements can be made to the implementation:

1. The search contains no strategy to decompose objects into parts
2. Ideally, sub-regions of similarity should be located within larger surfaces
3. The size of analysed surfaces was arbitrarily chosen and can be adjusted for better results
4. Matched surfaces should be verified by fitting the tool and object
5. The PCA based algorithm can be used to remove false positive results

Conclusion

This project has developed two approaches for a computational model of geometric reasoning in human tool use. The models would solve a task involving a pair of novel objects in which one object is used to lift and place the other at a new location. The task requires solving geometric constraints for tool-object interaction, and is best described as a process of technical reasoning.

In an exhaustive search model, a physics engine was used to find valid tool-object interaction by searching through all possible configurations. Although the search returns valid solutions for the lifting task, the model proves computationally unfeasible and unrealistic of human ability. Apart from long execution times the model would return solutions that are not physically possible to reach.

Heuristic approaches were considered for a second model based on visual cues of object and tool parts. The underlying hypothesis was that parts which fit would share similar geometries. A novel surface similarity technique was presented to assess object parts from 3D data points. Due to project time constraints, the approach was not fully implemented into a model demonstrating lifting solutions. Only the visual search step of the process was implemented, but requires further investigation as it employs a simplified algorithm which produces false positive matches. Visual search results should also be formally verified against observed human behaviour.

Both the exhaustive search and heuristic models can be improved to provide better results. The exhaustive search's simulation time can be reduced by running code on multiple machines and aggregating results. The heuristic model can be improved through better analysis of objects geometries, but also by considering forces generated by user actions (i.e. lifting). Nevertheless, even with the above improvements, model results require formal comparison to observed human behaviour performing similar tool-use tasks.

Appendix A - Compiling

The following section describes how to compile and run the computational model on Windows and Linux operating systems. Original source-code and documentation can be found at https://github.com/iceiony/4ConstraintsTheory The below installation steps are meant to be run using a terminal window.

Linux and **Unix** based operating systems require GCC compiler version 4.6 or earlier.

1. Install **git** and clone the code repository:

   ```
   apt-get update
   apt-get install git-core

   git clone --depth 1 https://github.com/iceiony/4ConstraintsTheory.git
   cd 4ConstraintsTheory
   git submodule init
   git submodule update
   ```

   If the folder ./ComputationalModel/newton-dynamics is missing, clone the physics engine repository:

   ```
   cd ./ComputationalModel
   git clone --depth 1 https://github.com/MADEAPPS/newton-dynamics.git
   ```

2. Install **cmake version 3.2** (which is not available on Ubuntu by default):

   ```
     apt-get install software-properties-common
     apt-get-repository ppa:george-edison55/cmake-3.x
     apt-get update
     apt-get install cmake
   ```

3. Install **GLFW3** library:

   ```
     add-apt-repository ppa:keithw/glfw3
     apt-get update
     apt-get install libglfw3
     apt-get install libglfw3-dev
   ```

4. Install **GLEW** library:

   ```
     apt-get install libglew-dev
   ```

5. Install **tinyxml** library:

```
apt-get install libtinyxml-dev
```

6. The computational model source code is located in the ./ComputationalModel subfolder. To compile it run:

```
cmake .
make -j4
```

Windows operating systems require the Visual C++ compiler which part of Visual Studio package. The Windows solution contains pre-compiled packages required for graphic rendering. As these packages were compiled using VisualStudio - 2015, other viersions of the compiler would not work.

1. Install **git** using GitExtensions-2.48-SetupComplete.msi During installation the following options should be chosen :

   MsysGit should be installed

   OpenSSH authentication should be chosen instead of Putty

2. After installation open **Git Bash** available in the start menu. Clone the code repository:

```
git clone --depth 1 https://github.com/iceiony/4ConstraintsTheory.git
cd 4ConstraintsTheory
git submodule init
git submodule update
```

   If the folder ./ComputationalModel/newton-dynamics is missing, clone the physics engine repository:

```
cd ./ComputationalModel
git clone --depth 1 https://github.com/MADEAPPS/newton-dynamics.git
```

3. Install **VisualStudio 2015** community edition, with the **Visual C++** option enabled.

4. Open the visual studio solution located in:

```
./ComputationalModel/vs\_2015/ComputationalModel/ComputationalModel.sln
```

The solution allows building any of the computational model executable. Output is directed to:

```
./ComputationalModel/vs\_2015/ComputationalModel/bin
```

Successful compilations would result in 4 executable files:

- ExhaustiveSearchGUI.exe
- ExhaustiveSearch.exe
- ExtractSurfacePointsGUI.exe
- VisualSearchGUI.exe

Appendix B - Software Glossary

This section describes the functional use of the computational model software. For the graphic version of the software (files ending in GUI), camera view-point can be controlled using **W,A,S,D** keyboard keys and **Mouse Click & Drag**.Execution of the 3D simulation can be paused and resumed by pressing the **P** key.

ExhaustiveSearchGUI.exe performs an exhaustive search of the tool and object fitting. It can be invoked from command line with the tool and object model files in 3DS file format. If no parameters are used, the executable will default to loading obj51.3ds and obj52.3ds .

```
./ExhaustiveSearchGUI.exe obj51.3ds obj52.3ds
```

On startup, the tool to object fitting is paused. Execution can be resumed and visualised in 3D by pressing the **'P'** key. Potential fitting position are written to **results.csv**.

ExhaustiveSearch.exe performs an exhaustive search without the use of a graphic user interface. As no 3D rendering is shown, the execution is considerably faster. All output is written to **results.csv**

ExtractSurfacePointsGUI.exe utility software for extracting points on the surface of an object in current view. The model of an object must be passed as parameter in 3DS format.

```
./ExtractSurfacePointsGUI.exe obj51.3ds
```

Surface points are recorded every time physics simulation is paused (by hitting the **P** key). Output is written to the **./surfaces** subfolder with one file per extracted surface.

VisualSearchGUI.exe performs a visual search of surfaces that would potentially offer good tool to object interaction. Tool and object models can be passed as parameters. If the program is invoked with no arguments, obj51.3ds and obj52.3ds are loaded.

```
./VisualSearchGUI.exe obj51.3ds obj52.3ds
```

During execution the tool and object are selectively rotated. For every rotation, random sub-surfaces are selected and checked for correlation. When correlation value is strong, execution is paused for a human observer to asses validity.

For analysing the feasibility of the novel visual correlation technique, Matlab was used. Scripts can be found in the **./CorrelationTetsts** subfolder. As a convention, files starting with a capital letter are a point of interest. They represent different tests executed for 2D and 3D surface correlation.

References

Battaglia, P. W., Hamrick, J. B., & Tenenbaum, J. B. (2013, October). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, *110*(45), 18327–18332. doi:10.1073/pnas.1306572110

Boeing, A. & Bräunl, T. (2007). Evaluation of real-time physics simulation systems. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07*. doi:10.1145/1321261.1321312

Boysen, S. T. & Himes, G. T. (1999, February). Current issues and emerging theories in animal cognition. *Annual Review of Psychology*, *50*(1), 683–705. doi:10.1146/annurev.psych.50.1.683

Chatbri, H., Kameyama, K., & Kwan, P. (2016, June). A comparative study using contours and skeletons as shape representations for binary image matching. *Pattern Recognition Letters*, *76*, 59–66. doi:10.1016/j.patrec.2015.04.007

Dickinson, S. J. & Biederman, I. (2014). Geons. *Computer Vision*, 338–346. doi:10.1007/978-0-387-31439-6_431

Harrington, M. (2009, July). Captive rooks master tool use. *Lab Animal*, *38*(7), 220–220. doi:10.1038/laban0709-220a

Hummel, J., Wolff, R., Stein, T., Gerndt, A., & Kuhlen, T. (2012). An evaluation of open source physics engines for use in virtual reality assembly simulations. *Lecture Notes in Computer Science*, 346–357. doi:10.1007/978-3-642-33191-6_34

Lefebvre, L., Reader, S. M., & Sol, D. (2004). Brains, innovations and evolution in birds and primates. *Brain, Behavior and Evolution*, *63*(4), 233–246. doi:10.1159/000076784

Loncaric, S. (1998, August). A survey of shape analysis techniques. *Pattern Recognition*, *31*(8), 983–1001. doi:10.1016/s0031-2023(97)00122-2

Navon, D. (1977, July). Forest before trees: the precedence of global features in visual perception. *Cognitive Psychology*, *9*(3), 353–383. doi:10.1016/0010-0285(77)90012-3

Osiurak, F. (2014a, April). What neuropsychology tells us about human tool use? the four constraints theory (4ct): mechanics, space, time, and effort. *Neuropsychology Review*, *24*(2), 88–115. doi:10.1007/s11065-014-9260-y

Osiurak, F. (2014b, April). What neuropsychology tells us about human tool use? the four constraints theory (4ct): mechanics, space, time, and effort. *Neuropsychology Review*, *24*(2), 88–115. doi:10.1007/s11065-014-9260-y

Osiurak, F., Jarry, C., Lesourd, M., Baumard, J., & Gall, D. L. (2013, August). Mechanical problem-solving strategies in left-brain damaged patients and apraxia of tool use. *Neuropsychologia*, *51*(10), 1964–1972. doi:10.1016/j.neuropsychologia.2013.06.017

Proffitt, D. R. (2006, June). Embodied perception and the economy of action. *Perspectives on Psychological Science*, *1*(2), 110–122. doi:10.1111/j.1745-6916.2006.00008.x

Robert, V., Arthur, C., Istvan, S., & Sergiu, N. (2012, August). Efficient real-time contour matching. *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*. doi:10.1109/iccp.2012.6356185

Roennau, A., Sutter, F., Heppner, G., Oberlaender, J., & Dillmann, R. (2013, November). Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots. *2013 16th International Conference on Advanced Robotics (ICAR)*. doi:10.1109/icar.2013.6766527

Veltkamp, R. C. & Hagedoorn, M. (2001). State of the art in shape matching. *Principles of Visual Information Retrieval*, 87–119. doi:10.1007/978-1-4471-3702-3_4

Zhang, D. & Lu, G. (2004, January). Review of shape representation and description techniques. *Pattern Recognition*, *37*(1), 1–19. doi:10.1016/j.patcog.2003.07.008

Zhu, Y., Zhao, Y., & Zhu, S.-C. (2015, June). Understanding tools: task-oriented object modeling, learning and recognition. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2015.7298903