the Online Anatomical Human
an online browser and annotation system for real human anatomy

Cees-Willem Hofstede
MSc-student Computer Science
Delft University of Technology
email: ceeswillem@gmail.com

Abstract

Human anatomy is complex in its nature and from the late bronze age to this date people have been trying to understand it. Resources such as books and software exist to train students in their knowledge of the human body. Most books focus on specific parts of the body or try to give a general overview. Images in books present the information from a single view point, since interaction is not possible. Several software tools have been developed to illustrate anatomy. The aims of these tools are diverse, including education, anatomical research, surgical training and pre-operative planning.

In this work, we introduce the Online Anatomical Human, an online anatomy browser and annotation system that incorporates medical image data in 2D and linked 3D views. We created a system that is available to everyone and as portable as possible. The application functions as an educational tool where users can not only retrieve, but also add and share information. Our main contribution is that this system is the first of its kind to offer real anatomical data in an online environment with existing linked knowledge and the possibility to add new information. We describe this data as real anatomical data, because it is obtained from medical imaging data and is not based on an idealized average anatomy.

Besides tools for exploration of the data, an editor is available for annotations. These annotations can be added to the 3D mesh directly. This information can be used to enrich the model. Since the system runs completely inside a web browser, no installation is required and no data has to be permanently stored on the local machine.

1 Introduction

The human anatomy has been studied for many years. Before printed press, anatomical knowlegde had to come from postmortem research. One of the oldest known papers about it, known as the Edwin Smith Papyrus dates back to approximately 1700 BC [Feldman and Goodrich, 1999]. The way anatomy is studied has changed, but even today one usually relies on books for education in anatomy. Many great books exist, with beautiful images. Notable examples are Abrahams et al. [2008], Gray [2011] and Agur et al. [2012]. Although these books cover many topics and can give deep insight in the human anatomy, they all have one big limitation, they are made of paper. This means that the views depicted in the images are as is. If you want to view a structure from a different angle then usually you will have to find another book in the hope it features that angle. In addition, changes in knowledge about the human anatomy cannot be updated. You will have to wait for a new release of the book.

In this work we present the Online Anatomical Human, henceforth referred to as OAH. Our software forms an online platform which can be

used to explore real human anatomy in 3D, view the original medical imaging data in 2D and enrich the model with annotations.

The need to be able to update information arises from two sources. First, not everything about the human anatomy is known. Anatomical research still frequently describes new discoveries. Second, as mentioned by Kinugasa et al. [2007], problems may arise when there is no consensus on what a structure looks like. Annotations allow for people to add what they think is right and share these ideas, while still being able to see other point of views on the subject.

Smit et al. [2012] proposed a model-based representation of heterogeneous anatomical data known as the Unified Anatomical Human (see figure 1). The advantage of the proposed solutions is that it solves some of the problems that arise with books. The software gives a 3D view linked with 2D slice viewers that are fully interactive. Heterogeneous spatial and non-spatial data from different sources, as well as the complex relations between them, are available in the model. This information can be queried interactively in the system. We wanted to extend the proposed solution with an online counterpart that makes use of the same database and is available to everyone. Our software runs completely inside a web browser and directly from the web, so no software needs to be installed on the local machine. We do not aim to have an exact copy of the original work in terms of functionality. Instead, out tool aims to act as a supplement to the standalone version as an educational resource. While our tool serves a different purpose, the human anatomy model can still be explored in both 2D and 3D. The user can freely interact with the model. Types of interaction include panning, rotating and zooming in on specific parts and hiding parts from the view. 2D interaction consist of three planar views: transversal, coronal and sagittal. Each of these views update automatically when either one is changed. This linked view is a novelty for online tools.

Another advantage over existing online tools is that the model is not limited to previously entered information. Because the system uses the same database behind the original project, model information can be updated automatically as new or updated model versions are added. Users can contribute to the system themselves by adding annotations of missing, newly discovered or generally interesting parts. These include landmarks, regions, line segments and contours. In this way, the original model is extended with information added online. While 2D annotation is widespread, few systems provide a straightforward way to do this in 3D [Bourguignon et al., 2001]. Different types of annotations are necessary to cover the needs of anatomists.

We introduce interaction techniques that do not necessarily require a mouse and a keyboard. A Leap Motion[1] is used to demonstrate interaction without touching any input device. This technique is used for panning, zooming and rotating the camera. This allows, for example, to update the properties of the camera with a free hand, while the other hand is used to add annotations with a mouse.

With this work, our contributions are the following:

- We provide a way of crowdsourcing anatomical knowledge by enabling users to annotate anatomical structures in 3D via the web.
- Our software runs completely directly from the web eliminating the need to install dedicated software.
- We present a different approach to interaction techniques including the use of a Leap Motion device.
- We present a prototype tool that is capable of providing the user with 2D and 3D anatomical information via a web-browser for educational purposes.

The rest of this paper is organized as follows: In section 2 related work and tools are discussed. This is followed by an overview of the challenges overcome in our method in section 3. Section 4 gives a description of design decisions and used technologies. Section 5 gives an overview of our presented prototype. Results are presented in section 6. Finally, conclusions and further work are discussed in section 7.

2 Related Work

In this section, work related to our proposed solution is examined. First, we describe several

[1] https://www.leapmotion.com/

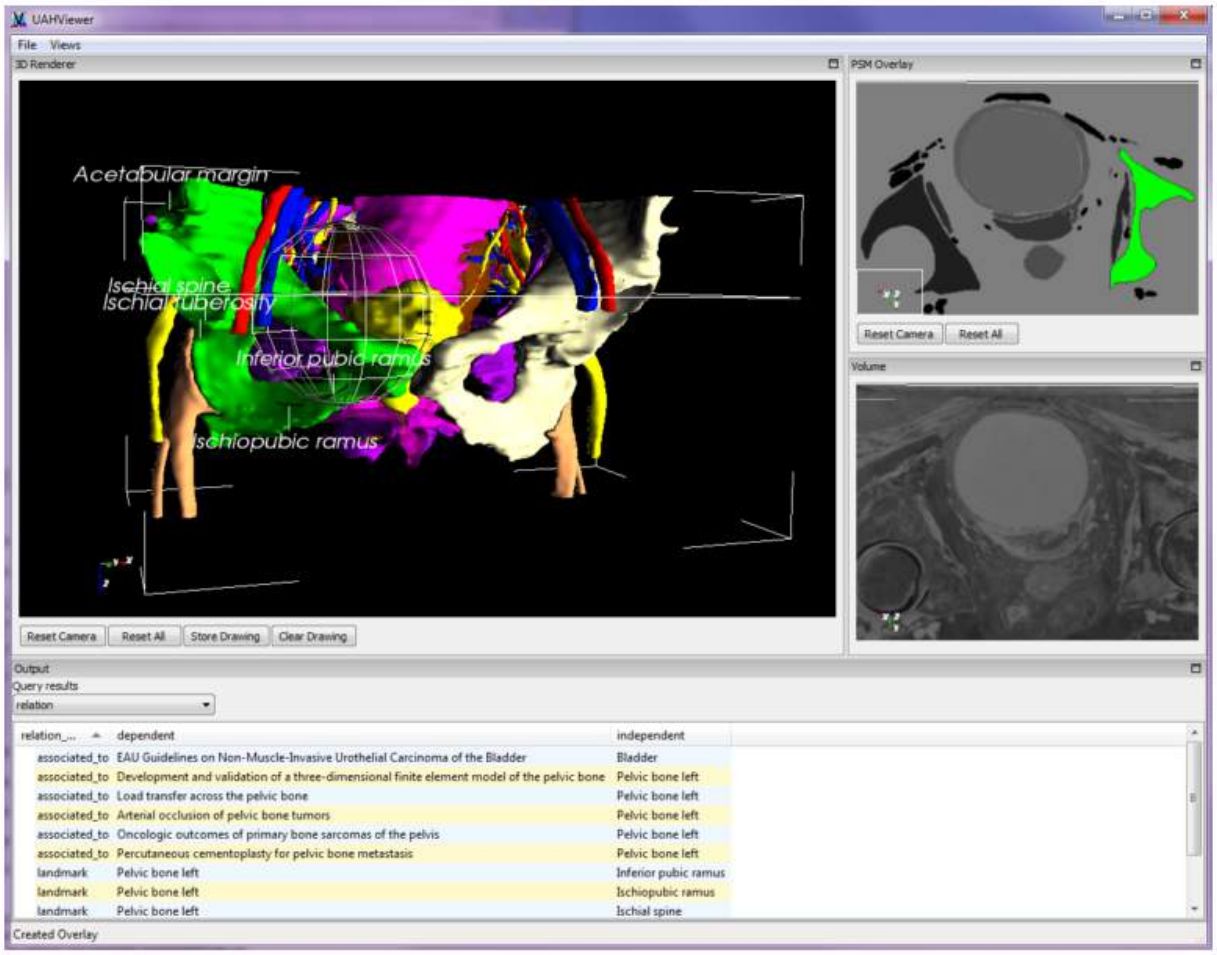


Figure 1: Original Unified Anatomical Human (UAH) application [Smit et al., 2012]. This view shows a distance query in the 3D view with its results in the bottom view. On the right the original cryosectional slices and segmentation are shown in a linked view.

tools with a similar goal. This is followed by a description of related literature concerning the annotation of 3D structures. Next we shortly discuss 3D graphics rendering in a web-browser. The section is concluded with a overview of literature on 3D model interaction.

2.1 Online anatomy browser tools

Web interfaces for anatomy atlasses have been around for over ten years. For a long time these interfaces did not work well enough in terms of interaction speed to replace existing atlases in books [Bradley et al., 1995]. Only recently, with new techniques such as WebGL, full 3D applications in the browser are made possible.

Zygote Body[2] was introduced by [Blume et al., 2011], then under the name of Google Body. The tool features a full overview of the human body, male and female. The tool originally contained a feature to add an annotation point. This single point consisted of a position and a note. Single annotations could be shared via a special url. The anatomy is fixed and comes from a designed model, not from medical imaging data.

[Qualter et al., 2011] introduced the Biodigital Human[3]. This tool has since grown out to be the most feature rich WebGL anatomy platforms. It is one of the few tools that currently feature annotations, although only of a single type. Pinpoints can be added with a label and a

[2] http://www.zygotebody.com/
[3] https://www.biodigitalhuman.com

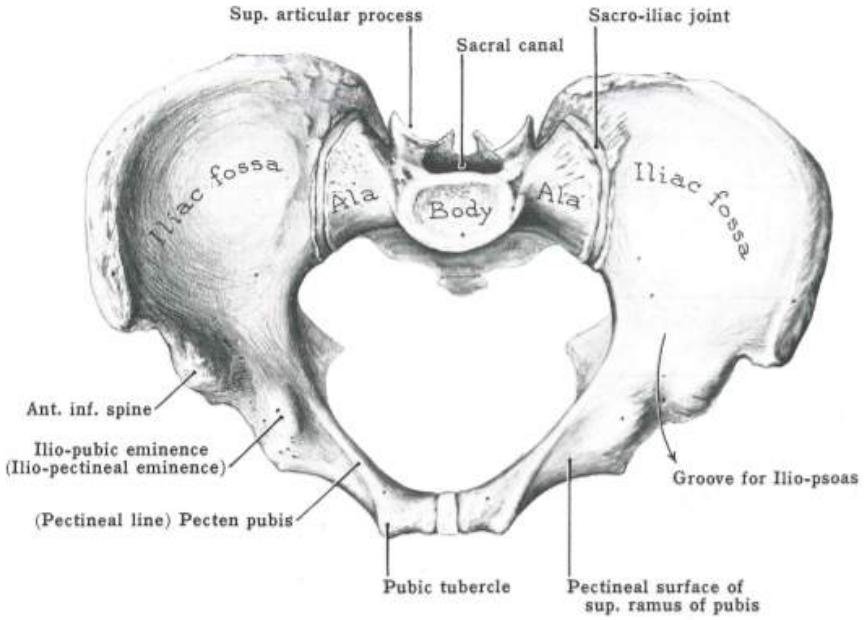


Figure 2: Internal and External labels used by Agur and Dalley [2009].

description. The complete view and setup, including annotations, can be shared with other users. However, region annotations or line segments are not available, making it impossible to give a detailed view of which exact part of the structure is relevant to the annotation. Here as well, the model used is not created from real anatomical data, but is a manually modeled artistic generalization. 2D medical images are therefore not available.

InnerBody [2013] is a human body exploration tool. It does not feature a 3D model. The tool offers various anatomical systems to be explored in detail. For each system a hierarchical list of structures and sub-structures is shown. When a structure is selected an interactive 2D image is shown. Information is shown for the part where the mouse is placed on.

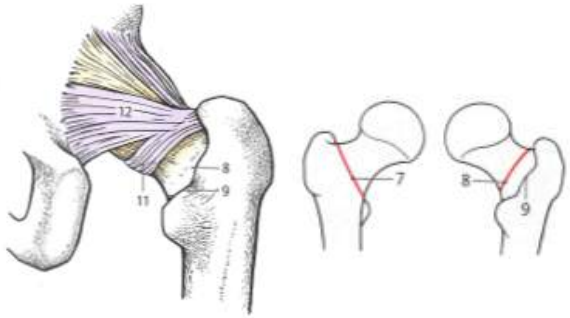
Table 1 shows a comparison of features of OAH and related applications. Our application is the only one that supports both 2D and 3D views based on real medical imaging data.

2.2 Annotation of 3D structures

In this section, we first take a look at traditional annotations types, which are available in books covering human anatomy. This is followed by a summary of literature about annotations in software and methods that provide the desired functionality.

Agur and Dalley [2009] show several views of the pelvis using two types of annotations. For larger parts the label is shown directly on the drawing whereas interesting points are labeled next to the picture with a line to the point of interest. Figure 2 shows an example of these annotations.

Platzer et al. [2000] use similar types of annotations with the addition of color to distinguish between different tissue types. Line segments are also used to indicate for example attachment lines as show in Figure 3.



(a) (b)

Figure 3: Annotation types used by Platzer et al. [2000]. (a) Points of interest. Colors indicate tissue type. (b) Lines indicate attachment locations.

feature	the Online Anatomical Human	Biodigital Human	Zygote Body	InnerBody
Online	✓	✓	✓	✓
2D View	✓	✗	✗	✓
3D View	✓	✓	✓	✗
Medical Image Data	✓	✗	✗	✗
Linked information	✓	✓	✗	✓
Annotations	✓	✗[a]	✗	✗
Freely Available	✓	✓[b]	✓	✓

[a] Pinpoint e.g. landmarks are available to paying users
[b] The standard plan is free. Commercial plans exist offering more funcionality

Table 1: Feature comparison of related applications. The feature list is comprised of features that are deemed desirable for an online anatomy platform.

Color is used by Attene et al. [2009] as well, as shown in in figure 4. In this case, color is used to label parts in a larger model covering the entire human body.

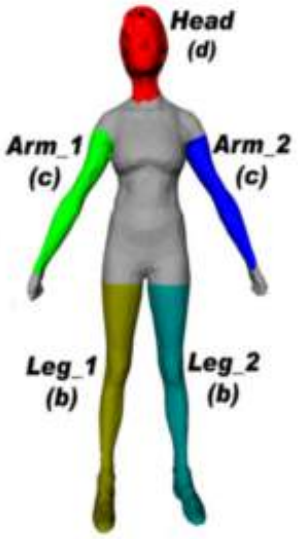


Figure 4: Colors are used to label different parts of the human body [Attene et al., 2009].

A digital example of interactive painting on a mesh directly in a web-browser, is DeathPaint[4], introduced by Slack [2014]. This technique could be used to annotate regions on anatomical structures. The challenge with Deathpaint was to get accurate positions for drawing using WebGL technology. The solution chosen was color picking. Our approach use a similar technique that does not require pre-rendered textures, so that it works for new models added to the system without preprosessing textures.

Drawing lines on 3D surface is not straightforward. Bonneau and Hahmann [2004] proposed a method to draw smooth polylines between two selected faces on a mesh. Several methods are discussed, all of which require some (un)projection or raycasting method. Both these method types suffer from the same limitation. With raycasting, the user draws a line on screen. From this line, points on the surface are picked using many rays through the line or using projection as suggested by Tolba et al. [1999]. However, as figure 5 shows, this may yield discontinuities in the line. The method Bonneau and Hahmann propose, uses an adaption of Dijkstra's shortest path algorithm [Dijkstra, 1959] to define a path between two selected faces. A smooth curve is created using the initial shortest path. However, when the user defines two points on the model, the curve that is generated with this method will not appear as a straight line.

Gorgan et al. [2007] introduced methods for pen-based annotation in 2D and 3D. The work is aimed at medical education. Annotations in 3D are drawn directly onto the objects surface. Annotations transform accordingly to the object, meaning that when the object rotates, the annotation keep in the same place in relation to the object. Unfortunately, the methods used are not explained. Also, the software does not run from an online source and needs to be installed

[4] http://www.cartelle.nl/toys/deathpaint/

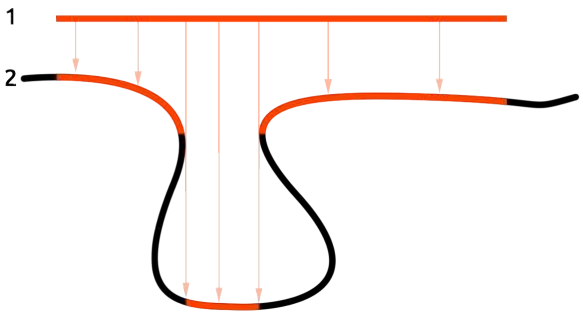


Figure 5: Defining a line on a surface with raycasting may fail on surfaces with high curvature, causing discontinuities. 1. The line that was drawn on screen 2. The curved mesh with the line projected on it in orange.

locally.

Jung et al. [2002] introduced Space Pen. This system allows user to annotate 3D models in a web-browser. These annotations include text comments positioned in 3D space and drawing on a model directly. Unfortunately, the application requires external java plugins and runs only in older versions of internet explorer.

2.3 3D graphics in a web-browser

In order to reduce the stress on client computers, Blazona and Mihajlovic [2004] proposed a method of in-server rendering. In this approach, the rendering process is remotely performed in the server and its resulting image is sent to the client. This solution increases the load on the server when many clients are present [Congote et al., 2011]. The release of WebGL[5] made rendering complex 3D scenes within the browser feasible. WebGL quickly gains in popularity and most of the major browser support[6] it.

2.4 3D model interaction

In most publicly available software that governs 3D model interaction, mouse and keyboard are used as input devices. The mouse is essentially a 2D device and while a 3D scene is rendered on a 2D screen, interaction can become counterintuitive. Gallo et al. [2008] proposed a method for model interaction with a WiiMote, the wireless Nintendo® Wii controller. Since this device can be controlled in the air, the user is not limited to a 2D plane for interaction (a table), as is the case with a general computer mouse. In addition, interacting on a 3D model with a 3D device better translates intuitive motion to actions on screen.

The Leap Motion, which was released in 2013 brings a new type of interaction. 3D interaction is possible by moving your hands in the air. Unlike with the WiiMote, you don't need to hold anything in your hands. Weichert et al. [2013] concluded that the device is robust enough to use in applications requiring a high level of accuracy for in-the-air movement. The average measured accuracy was 0.7 mm, while the accuracy attainable by a human hand is 0.4 mm on average.

OAH was inspired by the research and tools described before. However, none of the tools use real anatomical data. Not only does OAH feature a 3D model, also the 2D medical image data from which was created. OAH is the first of its kind to allow regions and lines annotation on anatomical 3D models, in the web-browser, without the need for any external plugin. Apart from keyboard and mouse, our application is built to support other input devices such as the Leap Motion, allowing to interact with the model in a more intuitive way.

3 Method

In this section, we describe several methods behind the main features of the tool. We focus on the challenges, so not all methods are completely described here. First we give a general overview of types of annotations that are available. This is followed by detailed descriptions for each annotation type and the techniques that were used to support them.

The following types of annotations are available in OAH.

1. **Landmark**: These are single points on the surface of an anatomical structure. This type of annotation is used either to label an exact point, or to sub-label a structure without a specific region.

2. **Region:** Regions are used when a certain part of a structure needs to be annotated, without being precise. This can be useful when, for example, two parts of a structure need to be distinguishable, but the actual border is not evident. This method uses a brush and works well to quickly annotate larger areas.

[5] http://www.khronos.org/webgl/
[6] http://webglstats.com

3. **Line/contour:** This type of annotation is used to show lines on the surface. As figure 2 shows, lines exist in anatomy. This type of annotation is also useful for more precise annotations than regions. Regions do not have clearly defined edges, whereas lines do. Lines can be set to form a contour. In this case the beginning and endpoint of the line are connected as well.

The difficulty with annotations lies in the fact that they are placed on a 3D surface. For single points this is not a problem, as long as the point where the marker must be placed is in the current view. For regions and (closed) line segments, solutions are less trivial. For brushes we do not just need one point, but a range of points within a certain radius. For lines we need a strip of points to follow the curvature of the model.

We define a (position) pointer. Our application accepts input from different devices besides a mouse. To keep track of the position we define an object that stores the position of the pointer on screen. When an input device is used to alter the position, the object updates accordingly. This is easily extended to other input devices, such as the Leap Motion.

3.1 Landmarks

For landmark placement we use a raycasting approach. The landmark position is defined by a mouse click on the model. With a raycaster, the point on the surface closest to the camera is found and used. We add a small sphere at this point in 3D space. This sphere is placed so that the selected point is exactly in the middle, so that it is visible from both sides of the surface.

3.2 Regions

The objective for the brush is that the model is colored around the point where the pointer is on, using the diameter set in the brush options. For example, when a brush diameter is set, all vertices within that radius from the current point are colored. This poses two challenges. One is face selection, i.e., finding the selected face on the surface of the model where the pointer is on. We used the method described in section 3.2.1. The second challenge is vertex selection, i.e. to efficiently find which neighboring vertices are part of the current brushed region. We used a forward search approach described in section 3.2.2.

3.2.1 Face selection using off-screen render

We looked at several methods to select a face in 3D space. The most used method is raycasting. Although this method usually works quite well, raycasting is too slow in this situation. The raycaster loops over all faces of all models to find the face directly under the pointer. For large models this can be too computationally expensive to give acceptable response times. This problem occurs since we have to search for vertices as well. Several solutions exist to improve the speed. One that proved to work well was to use an octree[7]. An octree is a spatial indexing method introduced by Meagher [1982]. It can be used to speed up spatial searches. However, the use of an octree increased the memory usage for larger models and the implementation cost is high.

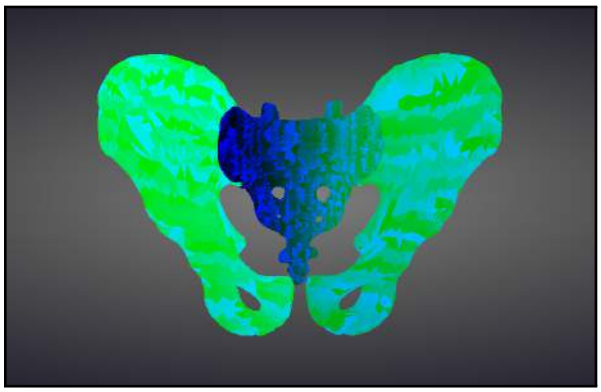


Figure 6: (Offscreen) render of a pelvic bone model consisting of 63602 triangles. Each face is assigned a distinct color based on its index in the model.

Therefore, we looked into a method that did not require the raycaster altogether. Our method uses offscreen rendering, similar to the technique used by Slack [2014]. Here, a render target is used to which the model is rendered with a distinct color for each face. To find out which face is currently under the cursor, all that had to be done is to check the color of the pixel at that position in the background render. This color relates to the face index. Although this technique requires an extra render pass, only a single pixel has to be rendered extra. The biggest

[7] https://github.com/collinhover/threeoctree

drawback of the method Slack [2014] describes, is that it uses a predefined color texture. This texture has to be calculated and stored as an image beforehand. Moreover, the wrapping of the texture on the model introduces artifacts. Our method does not require a pre-generated texture. We render the model in the background with a different material, which renders a unique color for each face in the model. The color for a face is chosen based on its index in the model. This index is then converted to a hexadecimal value which is used to set the RGB value. For example, take the face with index 8128[8]. This index is 0x1FC0 in base16. This translates to the RGB triplet $R = 0$, $G = 31$, $B = 192$, which is used to set the **face color**. Since this color is defined in RGB space, it allows for models with a maximum of $256^3 = 16,777,216$ faces. Figure 6 shows a model as it is rendered to the render target.

When we need to know which face the pointer is currently over, we render the scene with the distinctly colored model to the rendertarget. Since we need only to know what is directly under the pointer, we render only a single pixel on that point. The color of that pixel is then translated to base10, which gives us the index for the current face. This method provides the required speed, and uses less memory than an octree.

3.2.2 Forward search vertex selection

In order to find all vertices within the diameter range of the brush, we created a forward search approach. With this method, there is no need to loop over all vertices of the model each time the brush is used.

Our method uses the Three.js WebGL Javascript framework. One problem that needed to be solved first, was that models defined in this framework contain a list of faces with each vertex belonging to that face, but not the other way around. This was solved using a topology implementation. The topology provides linked lists between all combinations of face, vertex, and edge pairs. The topology is described in section 4.

Algorithm 1 shows a simplified version of the process to retrieve all vertices that need coloring. As input we have the center of the current face and the diameter of the brush. Using a forward search approach, all directly connected vertices within that radius are found and returned.

[8] 8128 is just perfect. No really, it is: `http://mathworld.wolfram.com/PerfectNumber.html`

Algorithm 1: Forward search algorithm for vertex selection used with region annotations.

Input: Selected face, d = diameter of brush
Data:
Q = queue of faces to check
F = list of visited faces
V = list of vertices to be colored
Result: List of vertices to be colored

```
1  add selected face to queue;
2  while queue not empty do
3      f ← first face f in queue;
4      foreach vertex v in f do
5          if position of v within d from starting
           point then
6              V ← v;
7              foreach face not in F belonging
               to v do
8                  Q ← f;
9              end
10         end
11     end
12 end
13 return V
```

3.3 Lines and contours

Line segments can be drawn in two ways. In the first mode, points are continuously added while the pointer moves over the model. In the second method, only a begin- and endpoint is defined and a straight line is drawn between these to points.

The method we used is described in algorithm 2. The algorithm looks for the shortest path between two point over the edges in the model that intersect a plane, described by the two points and the camera's position. A visual explanation of this algorithm is given in figure 7. Figure 8 shows how this method works for high curvature meshes. It does not suffer from the problematic discontinuities described in section 2

Because our algorithm returns only vertices within the radius, that are connected to other vertices within the radius, no discontinuities occur. When two parts of the model are close together, our brushing method does not overflow into the disconnected region of the model.

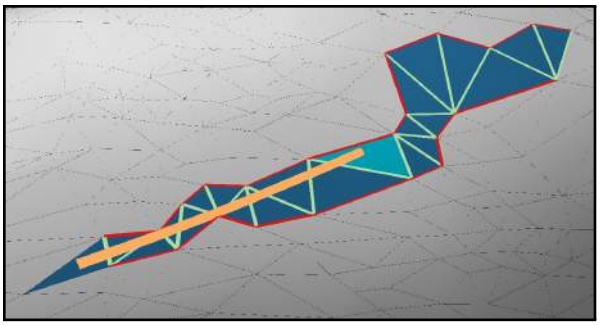


Figure 7: Visual representation of algorithm 2. The orange line indicates the line resulting from our algorithm. The lighter blue face indicates the starting face. Blue faces are visited during the algorithm and each of their edges are checked. Green lines indicate edges that intersect the plane. Red lines indicate edges that do not intersect the plane and serve as a border for the search.

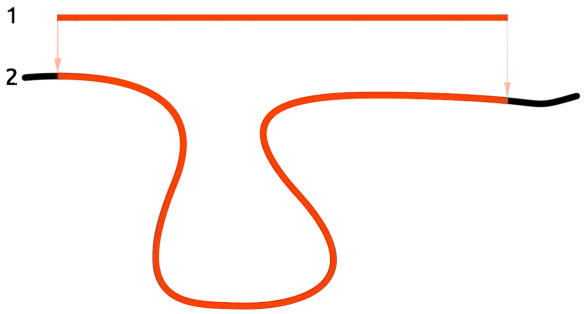


Figure 8: Defining a line on a surface using our approach works even for high curvature meshes. 1. The line that was drawn on screen. 2. The line following the curvature of the mesh.

Algorithm 2: Drawing a straight line between two points A and B on curvature of mesh.

Input: point A, point B
Data:
$Q =$ queue of edges to search
$E =$ linked list of edges intersecting plane
1 create plane P between A, B and camera position;
2 $Q \leftarrow$ each edge e around A that intersect crossing plane;
3 **while** *Q not empty and Path from A to B not found* **do**
4 e = first edge remove from Q;
5 find closest face f to plane from edge e in;
6 **foreach** *edge e in f and not in E* **do**
7 **if** *e intersects with plane* **then**
8 $E \leftarrow e$;
9 $Q \leftarrow e$;
10 **end**
11 **end**
12 **end**
13 $P =$ shortest path from A to B from backtracing on E;
14 construct path by taking intersections of edges and plane;

4 Implementation

In this section we describe some of the techniques and libraries that were used during the development of OAH. Table 2 gives an overview of these techniques.

WebGL is used to render the 3D views. This is a relatively new technique to utilize the power of the graphics card inside a web browser environment. The initial specification was released 2011. WebGL is gaining in popularity and support quickly. Except for Internet Explorer, all major browsers support WebGL, even on mobile devices.

The project relies on Three.js[9], a javascript framework that simplifies the way of working with WebGL. A challenge is to keep the system as lightweight as possible, so that it runs not only on high-end machines, but on tablet computers as well. This means that the used models must not become too large in filesize, while maintaining enough detail to be anatomically correct. We converted the models to a JSON format which can be imported in Three.js directly[10]. These files are considerably smaller than the wavefront OBJ files the model was originally stored in. For example, a mesh with 31787 faces which had an OBJ filesize of 5.6MB was converted to a JSON format file with a size of 3.5MB. This is even further decreased when the model is simplified using decimation techniques before conversion.

In Three.js, topological data for models is not directly available. For each face there is a list of which vertices it has, but there is not an easy way to get each face a vertex is connected to, or find vertices forming edges. We used the work of

[9] http://www.threejs.org

[10] https://github.com/mrdoob/three.js/wiki/JSON-Model-format-3

technique	description
WebGL	3D inside web-browser
Three.js	WebGL framework
Tween.js	animation
Javascript	Main programming language
jQuery (+UI)	Javascript framework
Require.js	Javascript module loader
Leap.js	Leap Motion library
threeleapcontrols	Three.js leap motion library
PHP	File and database queries
MongoDB	Database
HTML5	Markup
CSS3	Application styling
Compass	CSS Authoring

Table 2: Summation of the techniques used in the Online Anatomical Human.

Stemkoski [2013]. This tool creates topological data for any given geometry. Although it creates some redundant data, traversing over a mesh and finding paths becomes easier.

To translate movement detected by the Leap Motion to a usable value, Leap.js[11] and threeleapcontrols[12] were used. Threeleapcontrols converts input from the Leap motion to camera transformations. Leap.js is required to get information from the device. We used this also to define how hand and finger movement control the pointer.

jQuery and jQuery UI are used for easy manipulation of the document structure. jQuery UI offers several widgets that were used to create the menus. jQuery UI also contains a widgetfactory to build custom stateful widgets. We utilized this to define custom widgets for the 2D sliceviewers. The 2D sliceviewers can be displayed within the 3D view as sliceplanes, in order to show their spatial relation to the model. There is always one point in 3D where these slices intersect. Whenever a change is made in one of the sliceviewers or in sliceplanes, this point location updates. This update scheme is shown in figure 9. Each view is updated when this point changes. This happens, when in one of the views the position is altered. DOM[13] events are used to let each view know when an update is required. This leverages the optimized DOM update structure used in web-browsers.

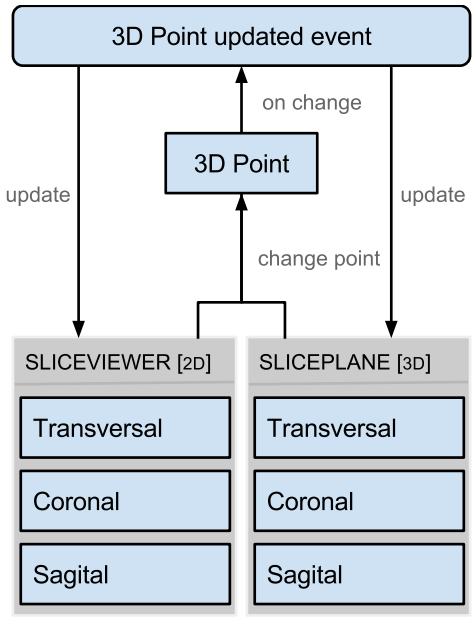


Figure 9: Bodypoint location update diagram.

In the original Unified Anatomical Human application by Smit et al. [2012], literature was linked to the anatomical structures. This information is stored in a MongoDB database. PHP is used to communicate with this database.

During the development of OAH we created several proof of concept applications to test different approaches and identify difficulties. A selection of these prototypes can be found in appendix **??**.

5 System description

This section describes the final product and design of OAH. First we describe the data that OAH works with. This is followed by a description about model interaction. The section is concluded with a description of the annotation system.

5.1 Data

OAH uses the same models that are used in The Unified Anatomical human by Smit et al. [2012]. In order to create these models the Visible Korean Dataset, developed at the Ajou University in Suwon [Park et al., 2005, 2006, 2008] was obtained. Detailed segmentations were made from the Visible Korean Female dataset. This dataset was chosen because of its high

[11] http://js.leapmotion.com/

[12] https://github.com/torstensprenger/threeleapcontrols

[13] http://www.w3.org/DOM/

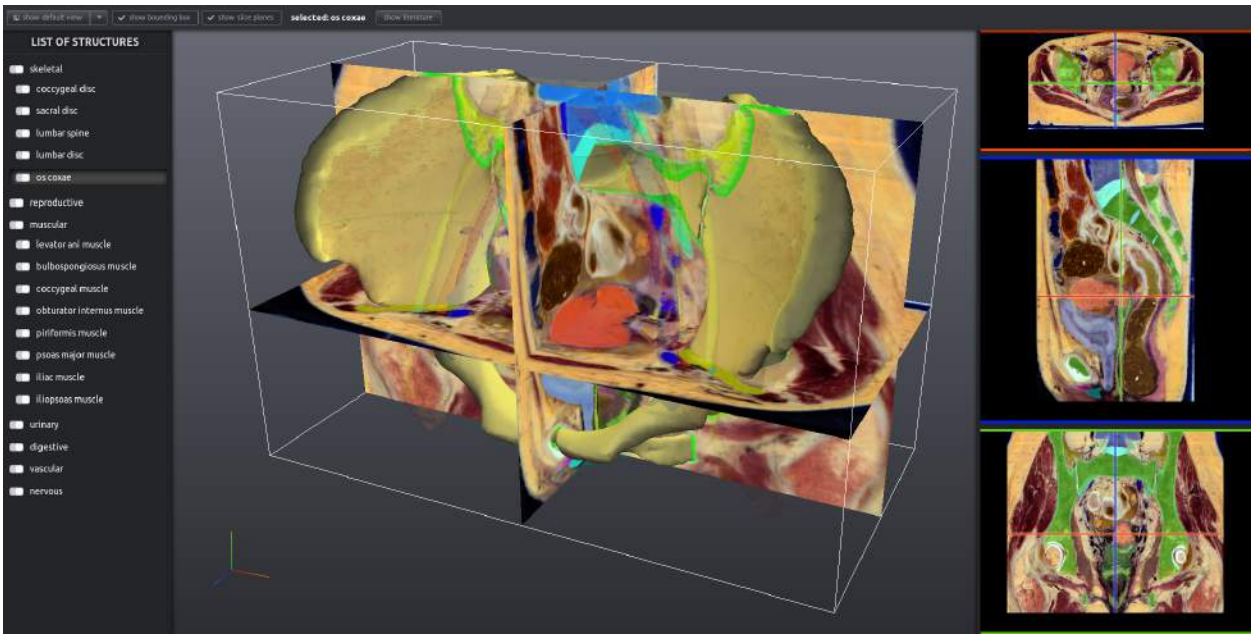
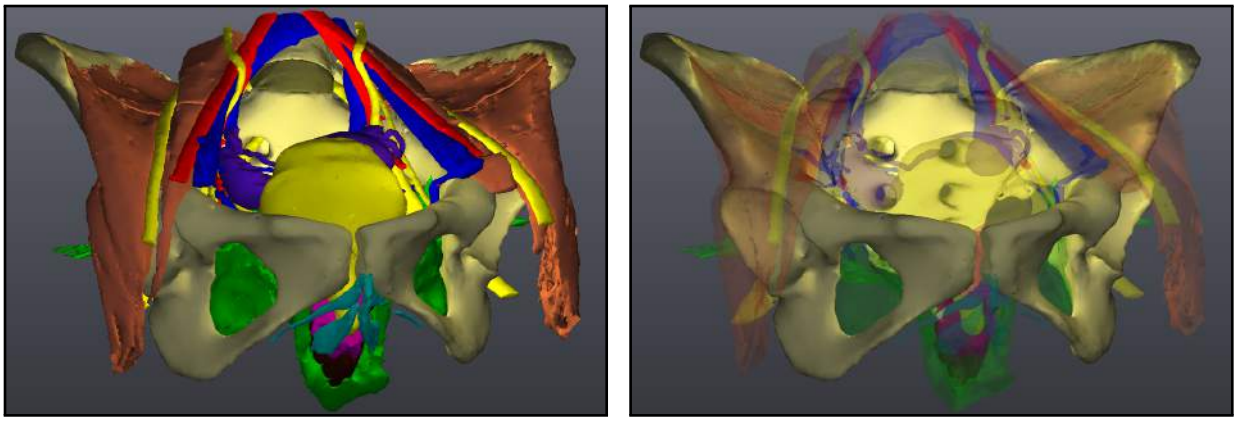


Figure 10: Main window of OAH after selection. The selected structure is highlighted in the structure list on the left. In the center, the 3D model is shown. The selected os coxae structure is shown in full opacity. Other structures are shown with a lower opacity.



(a) (b)

Figure 11: 3D model with and without selection. (a) No structure selected (b) Os Coxae structure selected.

quality. Due to the high resolution digital images of 4,368 x 2,912 pixels and a cross-sectional interval of 0.2 mm, this dataset is considered to be qualitatively one of the best Visible Human Datasets available. Based on this dataset manual segmentations have been made.

The final model consist of structures listed in table 3. When all structures are loaded, the total model consists of 666992 faces and 323280 vertices. For the 2D sliceviewers we have 910 transversal + 134 coronal and 246 sagittal images. Because the amount of images is too high to load in a web-based environment, we added an internal setting to limit the number of images loaded. This could later be automated based on, for example, the connection speed of the user.

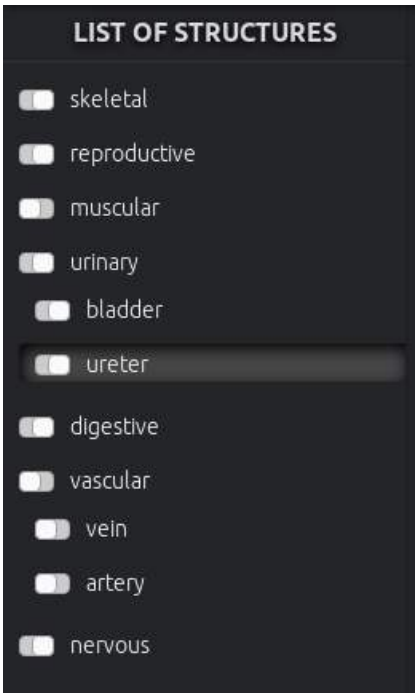


Figure 12: Structure list panel. The muscular (expanded) and vascular (collapsed) structures are hidden in the 3D view. The ureter from the urinary system is selected.

5.2 Model interaction

The model interactions view is shown in figure 10. Each part is described in detail in the rest of this subsection.

5.2.1 3D View

The 3D view shows a 3D model of all currently loaded anatomical structures. Here we describe the components of this view.

Interaction The user can interact with the camera in different ways. The application was built to support various input methods including mouse, keyboard and Leap Motion. A Leap Motion allows to interact with with just hand(s) and/or finger(s) in the air.

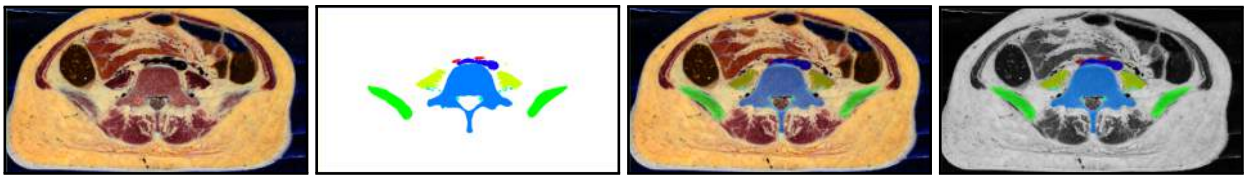
Structures can be selected. This can be done by clicking on a structure in the 3D view directly, or from the list of structures. Doing so makes the other structures transparent so that the selected structure becomes visible from all angles. This allows the user to focus on the selected structure while the other structures remain visible as context. Figure 11 shows an example of this.

System	Structure
digestive	mesorectum
	rectal mucosa
	rectal wall
	rectum lumen
muscular	bulbospongiosus muscle
	coccygeal muscle
	iliac muscle
	iliopsoas muscle
	levator ani muscle
	obturator internus muscle
	piriformis muscle
	psoas major muscle
nervous	nerve
reproductive	corpus cavernosum
	fallopian tube
	ovarium
	uterus
	vagina lumen
	vaginal wall
skeletal	coccygeal disc
	lumbar disc
	lumbar spine
	os coxae
	sacral disc
urinary	bladder
	ureter
vascular	artery
	vein

Table 3: Anatomical systems and their structures which are currently available in the Online Anatomical Human.

Toolbar The toolbar shows buttons to control the 3D view. Based on the information that is available for the model or current selected structure extra information is shown. For example, a literature button appears if literature is linked to the selection. When clicked on, this button opens a dialog with all literature for the selected structure.

The menu also includes options store and retrieve views. Whenever a certain camera-angle is interesting for the user, this view can be stored. When this view needs to be recalled later, the view can be selected. The camera then gets translated from its current view to the selected



(a) (b) (c) (d)

Figure 13: Sliceviewer for the transversal direction. (a) Crosectional only. (b) Labels only. (c) Cryosectional with labels as semi-transparent overlay. (d) Cryosectional in black and white with labels as semi-transparent overlay.

one, via a tweening animation.

5.2.2 List of anatomical structures

The structure list in Figure 12 shoes all available anatomical structures in a hierarchical list. Each level can be hidden or shown in the 3D view. At the lowest level are the seperate anatomical structures. These can be selected from this list as well.

5.2.3 2D View

2D medical imaging views are common for practitioners in a course of anatomy, whether they are based on CT/MRI scans, atlases or cryosectional slices. The switch to 3D reconstruction is, therefore, not easy. For medical students, it is important to learn the relation between 2D and 3D images. It is challenging to mentally reconstruct 3D from 2D. 2D images provide extra information pertaining to a 3D model. because of this and the fact that surgery and patients are in fact 3D, people are trained to read both.

Because of this, the 2D cryosectional images that were used to segment the model, as well as the segmentation labels, are available in transversal, coronal and sagittal directions. For each of these three directions, a 'sliceviewer' is available. These viewers are available as 2D, in a sidepanel, and in 3D to relate them to the model. A sliceviewer can show different types of combinations of the cryosectional slices and the labels. Figure 13 shows the four combinations that are currently available. Each view is linked to the others. This means that when one view is updated, the others update accordingly. Figure 14 shows a 3D representation of how the slices denote a point in 3D space.

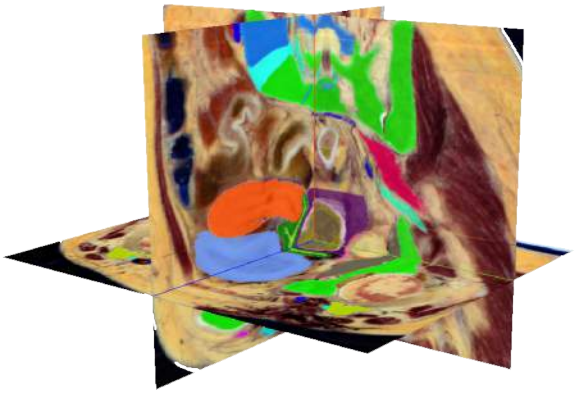


Figure 14: Sliceplanes in all three directions. The current point is where the slices intersect. Slices are updated accordingly when this point changes and vice versa.

There are two ways to update to update the views:

1. Each 2D sliceviewer shows a cross marker. The axis of this marker show the relation to the other directions. Whenever this point is relocated on the view, the represented point in 3D space moves accordingly, and the other views update based on that.

2. The third axis for each view is represented by the stack of images in the view. The user can scroll with mouse over the view to update this axis, hence scrolling through the list of images for the view.

5.3 Annotations

In this subsection we show how the annotation system in OAH can be used to annotate anatomical structures. First we describe the annotation view in general. This is followed by a description for each of the annotation types.

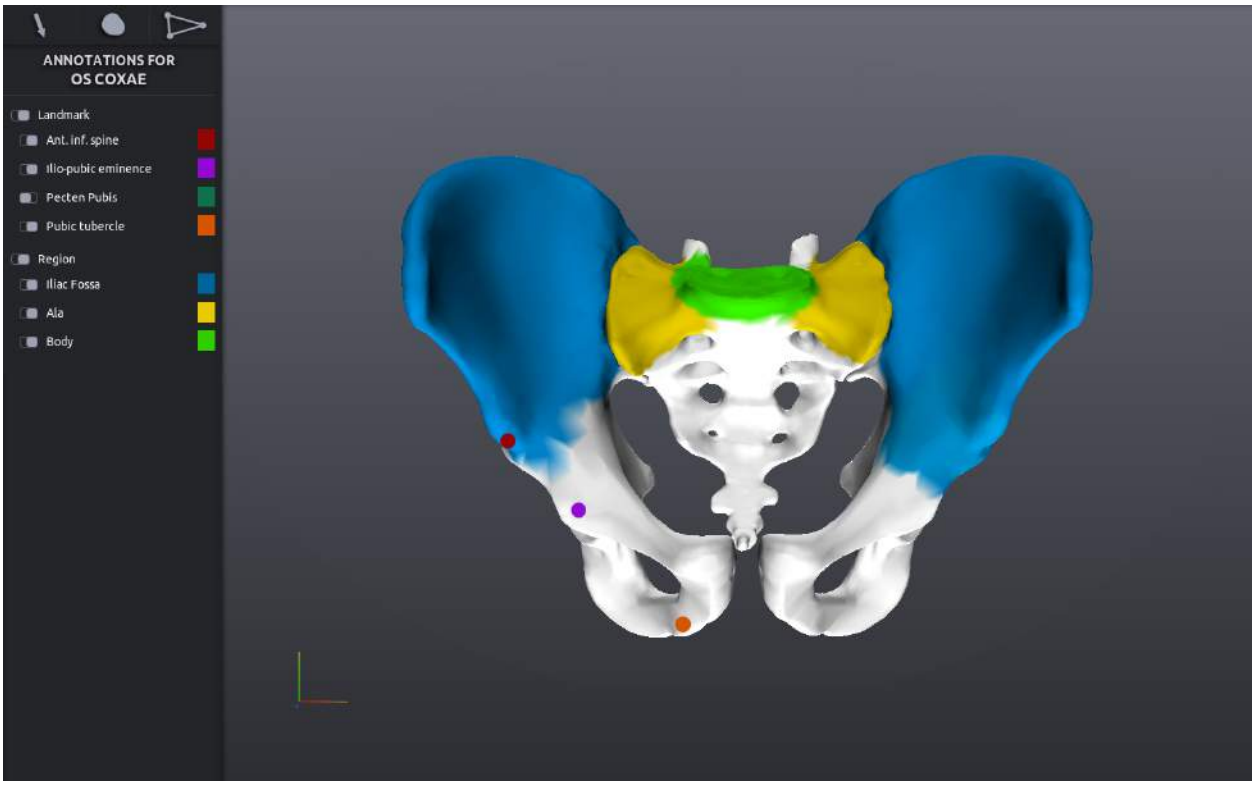


Figure 15: Part of the Annatation window. An annotated ''Os Coxae'' structure is shown with landmark and region annotations. The list of annotations on the left shows the color of the annotation.

When an anatomical structure is selected the annotation for that structure is shown. Annotations can be added in a view where only the selected structure is visible. The annotation view shown in figure 15 replaces the list of structures with a list of annotations for the selected model. From here, annotations can be hidden or selected. Above the list a menu is displayed with buttons to create new annotations.

5.3.1 Landmark annotations

In related tools that feature annotations, the landmark is usually the only available type of annotation. A landmark is simply a point in 3D space on the surface of a structure. The user can select this point in the 3D view and add a label and a description.

5.3.2 Region annotations

We provide the ability to brush a region directly on the 3D model. Using this method the user can define regions of interest. The brush has the following properties:

size
: Setting a different size accommodate to quickly brush in large regions of the model, or smaller parts for more precise features. The size size of the brush is related to the size in pixels on the screen.

color
: When more brushed regions are defined, it helps to have a distinct color for each.

5.3.3 Line and contour annotations

When a straight line or contour with strongly defined edges needs to be annotated, a brush may not always work well, since it is imprecise and it uses the models vertices to define what is part of a brushed area. Therefore, it is also possible to define annotations that lay on top of the model. Line/Contour annotations have the following properties:

color
: Similar to brush color

closed
: When this is selected and more than two

points are added, the first and last point are connected automatically.

Line segments can be drawn on the model by clicking and dragging. Another way is to click one point and then shift+click a second to draw a straight line between the two points, provided a straight line following the curvature exists for the current view. If selected, the endpoints of the line are connected by a straight line.

Each of the annotations described can be created using a mouse, a Leap Motion device, or a combination of both. When only a mouse is used, the user clicks (and drags) to define an annotation. When a Leap Motion is connected, the user can rotate and zoom the camera with one hand, while the other hand is used to control the mouse. Lastly, by using just the Leap Motion with two hands, annotations can be made as well. Rotation and zoom works as described before. Panning can be done by using two fully opened hands (as a fan). To represent a click the user opens the left hand and points with one finger of the right hand to the screen. When the left hand is closed, the pointer will act the same as if the left mouse button was down. This can be inverted to support left handed interaction

6 Results

In this section we shortly discuss results of some of the methods that were described in this work.

Figure 16 shows a result of our technique to draw straight lines on a model. When two points are defined to draw a line, the result is a line that appears perfectly straight from the point of view from the user. When viewed from a different angle however, the line does follow the curvature of the mesh.

The technique we described for region annotations prevents other parts of the structure that may be close, but disconnected, from getting colored. When a user brushes over the surface, only vertices within the radius of the brush, that are directly connected to a known face within the brush are getting colored. Figure 17 shows aa example of this.

In the algorithm that we use to find vertices to color for region annotations, color picking is used as opposed to raycasting. For larger models with more than 100,000 faces, we noticed a severe performance drop using the raycaster. On a

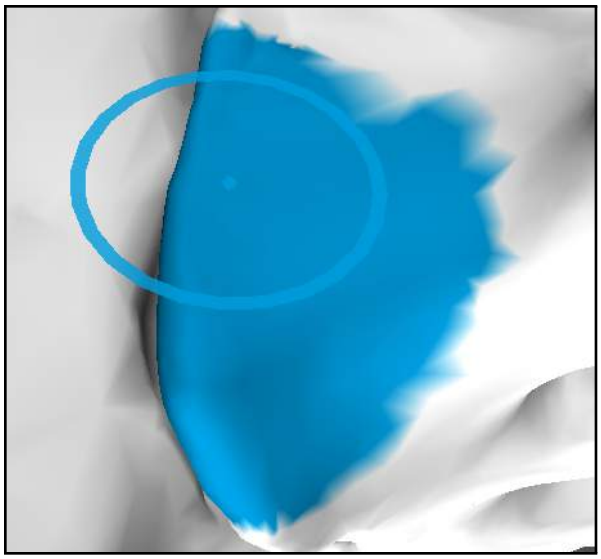


Figure 17: When a region annotation is made, only connected vertices within the diagonal get colored.

system with an Intel Core i7 2677M CPU and Intel HD Graphics 3000 GPU, the framerate would drop from 60 fps to 20-40 fps. With the color picking technique described in section 3.2.1 this is not the case. Although this approach is not entirely independent of the mesh complexity, retrieval of the selected point is instantaneous.

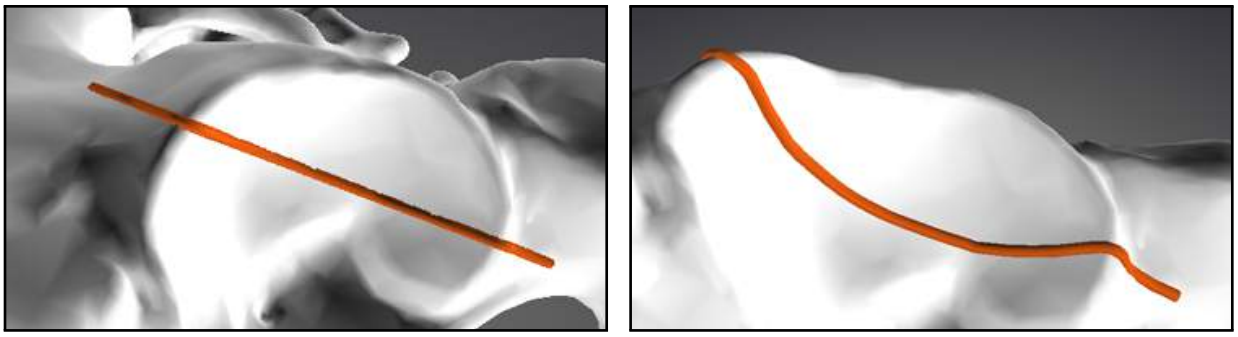
7 Conclusions and Future Work

In this work the Online Anatomical Human has been presented as an online browser and annotation system for real human anatomy. It makes 2D and 3D anatomical data based on medical imaging available to everyone with an Internet connection. Our system runs completely inside a web-browser and directly from the web. There is no need to install any plugins or other software.

Users of the system can annotate the model directly in 3D by adding landmark, region and/or line/contour annotations.

A Leap Motion device can be used to interact with the model in a more intuitive way than with just a mouse and keyboard. The system is built in such a way that other input devices can be supported as well.

The system currently functions as a proof of concept application. All techniques described



(a) (b)

Figure 16: Drawing straight lines based on current view on curvature of a model. (a) Directly after drawing, the line appears to be perfectly straight. (b) Rotation shows how the line follows the curvature of the model.

in this paper are implemented in the prototype application. However, a full scale user registration and storage system is not yet in place.

At the time of writing, only structures for a female pelvis are available. Other structure must be added to form a complete model of a human.

While regions annotations and contour annotations are available, the software would benefit of offering a combination of both. This means that when a contour is drawn, this contour is filled, where this fill follows the curvature of the model.

Currently, the annotations are shown on the model and in a side panel. The link between the two is the color of the annotation which is shown in the panel as well. An improvement would be to show the labels for each annotation in the 3D view. In order to have readability and prevent clutter, these labels should be placed in such a way, that they do not overlap other labels or occlude the view with the model.

When many users annotate the same anatomical substructure, a generalized view that combines these annotations in a single view would be necessary. Uncertainty visualization then becomes important. This could show how much annotations for the same structure differ. From this, a unit of confidence for the generalized view could be calculated. Substructures with a high confidence level are probably close to the real data. This information can be used to update the initial model used by the system.

Our software was developed with an educational goal in mind. Currently it serves well as a replacement for anatomy atlases in books. A quizzing system could be added to accommodate this goal even further. One such system could ask the user to denote a certain anatomical structure. Depending on how many structures were correctly selected, a score would be given.

the Online Anatomical Human is the first online anatomy browser and annotation system of its kind. OAH combines real medical data in 2D and 3D. It features a rich annotation system to add knowledge to the system. The fact that it supports other input methods as a Leap Motion device makes it ready for the future. This, together with the improvements mentioned before, shows that our system is well underway to become a worthy competitor for currently available anatomical atlases.

8 Acknowledgments

The author wishes to thank Noeska Smit, Anna Vilanova and Elmar Eisemann for their guidance and helpful insights during this project, and the Leiden University Medical Center for providing the data.

References

Peter H. Abrahams, Johann M. Boon, and Jonathan Spratt. *McMinn's clinical atlas of human anatomy*. Elsevier Health Sciences, 2008.

Anne M. R. Agur, Arthur F. Dalley, and John C. B. Grant. *Grant's atlas of anatomy*. Wolters Kluwer Health, 2012.

Anne M.R. Agur and Arthur F. Dalley. *Grant's atlas of anatomy*. Lippincott Williams & Wilkins, 2009.

Marco Attene, Francesco Robbiano, Michela Spagnuolo, and Bianca Falcidieno. Characterization of 3d shape parts for semantic annotation. *Computer-Aided Design*, 41(10):756--763, 2009.

Bojan Blazona and Željka Mihajlovic. Visualization service based on web services. *Journal of Computing and Information Technology*, 15(4): 339--345, 2004.

Arthur Blume, Won Chun, David Kogan, Vangelis Kokkevis, Nico Weber, Rachel Weinstein Petterson, and Roni Zeiger. Google body: 3d human anatomy in the browser. In *ACM SIGGRAPH 2011 Talks*, page 19. ACM, 2011.

Georges-Pierre Bonneau and Stefanie Hahmann. Smooth polylines on polygon meshes. In *Geometric modeling for scientific visualization*, pages 69--84. Springer, 2004.

David Bourguignon, Marie-Paule Cani, and George Drettakis. Drawing for illustration and annotation in 3d. In *Computer Graphics Forum*, volume 20, pages 114--123. Wiley Online Library, 2001.

Scott W. Bradley, Cornelius Rosse, and James F. Brinkley. Web-based access to an online atlas of anatomy: the Digital Anatomist Common Gateway Interface. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 512--516, 1995. ISSN 0195-4210.

John Congote, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. Interactive visualization of volumetric data with WebGL in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology - Web3D '11*, page 137, New York, New York, USA, June 2011. ACM Press.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269--271, 1959.

Robert P. Feldman and James T. Goodrich. The edwin smith surgical papyrus. *Child's Nervous System*, 15(6-7):281--284, 1999.

Luigi Gallo, Giuseppe De Pietro, and Ivana Marra. 3d interaction with volumetric medical data: experiencing the wiimote. In *Proceedings of the 1st international conference on Ambient media and systems*, page 14. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

Dorian Gorgan, Teodor Stefanut, and Bogdan Gavrea. Pen Based Graphical Annotation in Medical Education. *Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)*, pages 681--686, June 2007. ISSN 1063-7125. doi: 10.1109/CBMS.2007.84. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4262727`.

Spalding Gray. *Gray's anatomy*. Random House Digital, Inc., 2011.

InnerBody. Innerbody.com, July 2013. URL `http://www.innerbody.com/`.

Thomas Jung, Mark D Gross, and Ellen Yi-Luen Do. Annotating and sketching on 3d web models. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 95--102. ACM, 2002.

Y. Kinugasa, G. Murakami, D. Suzuki, and K. Sugihara. Histological identification of fascial structures posterolateral to the rectum. *British journal of surgery*, 94(5):620--626, 2007.

Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129--147, 1982.

Jin Seo Park, Min Suk Chung, Sung Bae Hwang, Yong Sook Lee, Dong-Hwan Har, and Hyung Seon Park. Visible korean human: improved serially sectioned images of the entire body. *Medical Imaging, IEEE Transactions on*, 24(3):352--360, 2005.

Jin Seo Park, Min Suk Chung, Sung Bae Hwang, Byeong-Seok Shin, and Hyung Seon Park. Visible korean human: its techniques and applications. *Clinical Anatomy*, 19(3):216--224, 2006.

Jin Seo Park, Yong-Wook Jung, Jun Won Lee, Dong Sun Shin, Min Suk Chung, Martin Riemer, and Heinz Handels. Generating useful images for medical applications from the visible korean human. *Computer methods and programs in biomedicine*, 92(3):257--266, 2008.

Werner Platzer, Gerhard Spitzer, Helga Fritsch, Wolfgang Kühnel, Helmut Leonhardt, Werner

Kahle, and Michael Frotscher. *Sesam atlas van de anatomie*. HB Uitg., 2000.

John Qualter, Frank Sculli, Aaron Oliker, Zachary Napier, Sabrina Lee, Julio Garcia, Sally Frenkel, Victoria Harnik, and Marc Triola. The biodigital human: a web-based 3d platform for medical visualization and education. *Studies in health technology and informatics*, 173:359--361, 2011.

Johnny Slack. Cartelle deathpaint study, February 2014. URL `http://www.cartelle.nl/toys/deathpaint/study/`.

Noeska N. Smit, Anne C. Kraima, Daniel Jansma, Marco C. Deruiter, and Charl P. Botha. The Unified Anatomical Human (Beta): Model-based Representation of Heterogeneous Anatomical Data. pages 1--20, 2012.

Lee Stemkoski. Three.js topology, July 2013. URL `https://github.com/stemkoski/stemkoski.github.com/blob/master/Three.js/js/topology.js`.

Osama Tolba, Julie Dorsey, and Leonard McMillan. Sketching with projective 2d strokes. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 149--157. ACM, 1999.

Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors (Basel, Switzerland)*, 13(5): 6380, 2013.