Echocardiography Guidance and Training with Mixed Reality

Edvart G. Bjerke

March 2025

1 Introduction

The aim of this thesis is to implement an echocardiography guidance and training system in augmented reality. The system is targeted at trainees, students and other unskilled practitioners. The goal is to provide a tool that is helpful both in learning about how to perform echocardiographic examinations and in actual examinations.

A pre-trained deep neural network- based model will be used to generate guidance signals for the user, and these signals will be integrated into the augmented reality system. Additionally, virtual reference model placed in the virtual environment allows the user to explore cardiac structures and familiarize themselves with standard views.

The reference models are based on cardiac MRI data. Since the project is focused on echocardiography, it's essential that the reference model also includes ultrasound data. This data should be spatially complete such that the user can view scan planes from arbitrary poses in a continuous manner. To maintain coherence, all representations (raw MRI, solid model and ultrasound) are obtained from the same patient. This ensures that context is not lost when switching between representations.

We wish to investigate whether AR technology coupled with a guidance model can enhance the learning experience of novice echocardiography users. This thesis hypothesizes that such technologies can reduce friction in the learning experience through increased interactivity and reduction of the context-switching apparent in traditional learning methods. We propose that providing immediate feedback and displaying information directly in the users field of view promotes more focused learning where the need to shift attention, both physically and mentally, is reduced. We also aim to investigate if these techniques improve the user's ability to understand and internalize the geometry of cardiac structures.

2 Research Question and Evaluation Plan

The following is a preliminary overview of the research question and evaluation plan.

To assess the effectiveness of augmented reality (AR) in echocardiography training, this thesis will investigate the following question:

How can AR-based guidance improve the learning experience of novice echocardiography users? In particular, experiments will focus on whether AR-based guidance leads to faster skill acquisition, higher success rates in obtaining diagnostically acceptable views, and improved learner confidence and engagement.

To answer this question, a user study will be conducted comparing two training methods: one based on current standard practice, and another using the proposed AR guidance system. Participants will be novice users with limited or no prior echocardiography experience, such as medical students or early-stage residents. Participants will be randomly assigned to one of the two groups.

Both groups will be instructed to obtain a set of apical views using a physical ultrasound probe in a simulation setting. The AR group will have access to live guidance through the headset, including real-time feedback and visualization of target probe positioning. The control group will follow conventional instructional material, such as handouts, diagrams, and verbal/written instructions.

Several evaluation metrics will be collected:

- **Time to acquisition:** The time taken to successfully acquire each standard view.
- **View quality:** Expert assessment of the diagnostic acceptability of each view, based on predefined scoring criteria.
- **User feedback:** A post-task survey measuring perceived learning effectiveness, system usability, and subjective confidence.

The primary outcome of interest is whether participants in the AR-guided group can learn to obtain standard views more quickly and with higher accuracy than those in the control group. Secondary outcomes include participant-reported engagement and satisfaction with the training method.

This evaluation plan will provide some quantitative evidence on the pedagogical value of our echocardiography guidance system. Insights from the user study may also inform further system improvements and broader applications in imaging training.

3 Echocardiography

Echocardiography examinations provide information about the patient's heart used to diagnose various cardiovascular disorders. Today, echocardiography is used extensively, as the procedure is low-cost and non-invasive [1]. In Transthoracic Echocardiography (TTE), the ultrasound probe is positioned externally-

on the chest of the patient. This is the most common technique for obtaining echocardiograms, and is less invasive than Transesophageal Echocardiography (TEE), where a probe is inserted into the esophagus [1]. In this thesis, we are only concerned with TTE.

In echocardiography, ultrasound waves are transmitted by a probe. The waves propagate through the medium of the patient and echoes are recorded by the probe. These echoes are processed by a set of signal processing algorithms to produce the ultrasound image. A technique known as "beamforming" is an essential part of the ultrasound processing chain. There are several imaging *modalities* used in cardiac ultrasound. The most common types are M-mode, 2D and 3D imaging. An example of 2D and 3D echocardiography is shown in figure 1

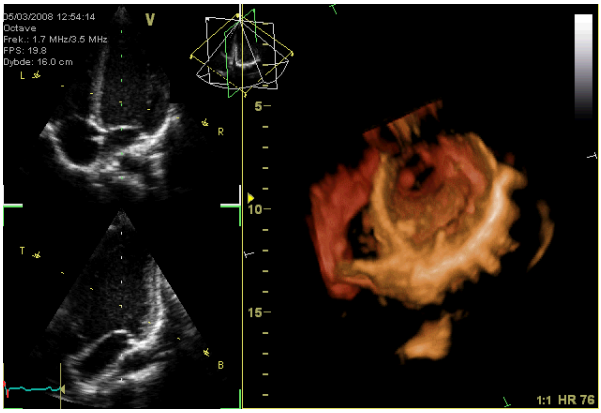

Figure 1: An example of 2D and 3D echocardiography images [2].

In M-mode imaging, only a single ultrasound scan line is processed and displayed. This modality may be useful when evaluating the motion or timing of a specific structure with respect to the cardiac cycle [3]. Two-dimensional ultrasound allows the generation of cross-sectional images of the anatomy.

Three-dimensional ultrasound technology produces volumetric data of the anatomy by using two-dimensional transducer arrays. These volumes can be produced in real-time and displayed using volume rendering techniques like ray-marching or isosurface generation. One key advantage with 3D echo is that volumes can be estimated directly from measurements reducing errors intrinsic in volume estimation from cross-sectional slices [4]. In this thesis, we are mostly concerned with two-dimensional imaging as this is the most available method.

Beamforming technqiques exploit wave interference to combine ultrasound signals from separate transducers. The result of beamforming is a greyscale image visualizing anatomical structures. More specifically, the intensity of a pixel in the image correlates with the strength of the reflected echo, influenced by differences in acoustic impedance.

Due to the intricacies of acoustic wave propagation and limitations in beam-

forming, ultrasound images are generally more difficult to interpret than those obtained by Computed Tomography or Magnetic Resonance. These images typically contain significant noise and frequently exhibit visual artefacts such as *abberations* and *reverberations*. Echocardiography is specifically limited due to rib blockages, and the probes are designed such that the acoustic field of view may pass in between the ribs of the patient.

3.1 Standard Viewing Planes

The standard viewing planes are specific standardized cross-sections used to assess cardiac health. The standard views are as follows [5]:

- Parasternal long axis
- Parasternal short axis
- Apical four-chamber
- Apical two-chamber
- Subcostal chamber

These views are obtained by placing the probe at specific positions and orientations, which requires training and experience. This paradigm is useful as it provides well-defined goals for the user to reach and forms a basis for objective assessment of the quality of the examination.

For the scope of this thesis, we choose to only consider views from the apical window. This is done for simplicity, as all views can then be obtained by rotation of the probe at a single location on the thorax.

Views in the apical window are obtained by placing the probe at the *apex* of the left ventricle [5]. With the probe in the correct location for the apical window, all apical views can be obtained by rotating the probe in place

4 System Architecture and Implementation

The echocardiography guidance system is implemented as a mixed reality application, integrating several key components:

- An ultrasound machine
- The Quest 3
- A game engine with a custom rendering implementation and AR integration
- Virtual anatomical reference models from real patients, with time-varying models acquired by various imaging technologies
- A probe position/orientation tracking system

- A neural network-based guidance model
- A computer with a modern Graphical Processing Unit for rendering and model inference
- A hardware/software interaction layer to handle I/O

The game engine Unity [6] is chosen for its flexibility in rendering implementation and its AR integration system. Unity allows for custom shader injection [7] which allows us to add non-standard rendering methods such as volumetric ray casting. Also, scripts written in C# can be used to incorporate custom game/system logic.

The neural network guidance model, implemented in PyTorch [8] will receive probe data and the current echo image and predicts adjustments toward a target standard view. This prediction is visualized in the AR interface as an arrow or overlay aligned with the user's field of view. To handle the complexity of hardware, drivers, data transfer and general interoperability in such a complex system, ROS (Robot Operating System) [9] will be used.

Extracting data from the ultrasound machine itself will be done using a capture card device. The images are then distributed within the system with ROS, to be used as input to the guidance model and for display in the Unity-powered AR system.

The reference models will mainly be represented as surface meshes and scalar fields. These will be visualized in unity the built-in mesh renderer as well as custom surface and volume renderers. A custom animation system implementation will also be used to handle smooth rendering of 4D reference models.

To track the position and orientation of the ultrasound probe, AR tags will be used. AR tags are square fiducial markers in black and white with known patterns and dimensions [10]. These markers are identified in camera images using computer vision techniques and their relative position and orientation can be computer based on the pixel positions of marker keypoints.

5 Computer Graphics

Computer graphics (CG) is the study of digitally generating images, videos, and other types of graphical content. The type of visual content generated can broadly be categorized into photorealistic and non-photorealistic CG. The applications of CG techniques are broad- including scientific visualizations, video games, special effects and other types of informative or artistic content.

Algorithms in computer graphics typically operate on objects of a three-dimensional (3D) *scene*. The scene is a high-level description of the virtual environment used to synthesize the visual content. The objects in the scene may represent things like solid bodies, cameras, light sources or arbitrary volumetric data.

5.1 Rendering

Rendering algorithms generate digital images, given a set of objects from the scene. These algorithms are often physically-based, where *photorealistic* techniques simulate real light phenomena to produce visually realistic results.

Non-photorealistic rendering techniques are used when photorealism is n ot desired- whether for aesthetic purposes or when the objects represent something invisible to the human eye.

5.2 The Graphics Pipeline

The graphics pipeline is a high-level description of the sequence of processes used to generate an image from the scene description. The graphics pipeline is not rigidly defined, and different architectures are used to render different tyepes of data or to achieve results of a particular aesthetic.

In real-time computer graphics, the graphics pipeline can generally be viewed as three parts- the application, geometry processing and rasterization [11].

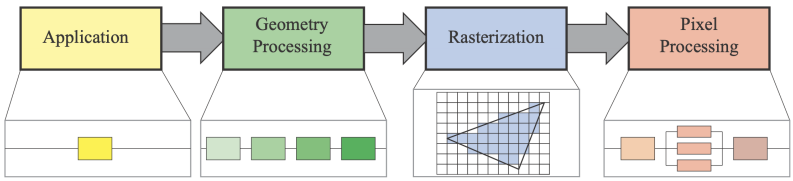

Figure 2: The graphics pipeline [11]

The application part of the pipeline generally runs on the CPU, and is concerned with defining the structure of the scene, setting up rendering commands and other pre-processing procedures. The application defines a set of geometry *primitives* which are sent further down the pipeline. These primitives are geometrical shapes like triangles and lines, defined by coordinates and indices defining the surface of the models to be processed and rendered by the GPU. The data strutures representing these primitives are typically defined in a local coordinate system referred to as *object space*, and may include additional information such as surface normals and texture coordinates. The data required in this stage depends on the algorithms employed in the *pixel processing* stage of the pipeline.

The geometry processing stage is executed on the GPU, where geometrical operations like morphing and subdividing (tesselation) may be performed. An essential part of the geometry processing stage is the *vertex shader*, which is a programmable procedure which processes each 3D vertex (position) of the given mesh.

In a typical rendering pipeline, the vertex shader transforms the vertex coordinates from *object space* to *screen space* through perspective projection, such

that objects are correctly rendered to the screen according to the *pinhole camera model*.

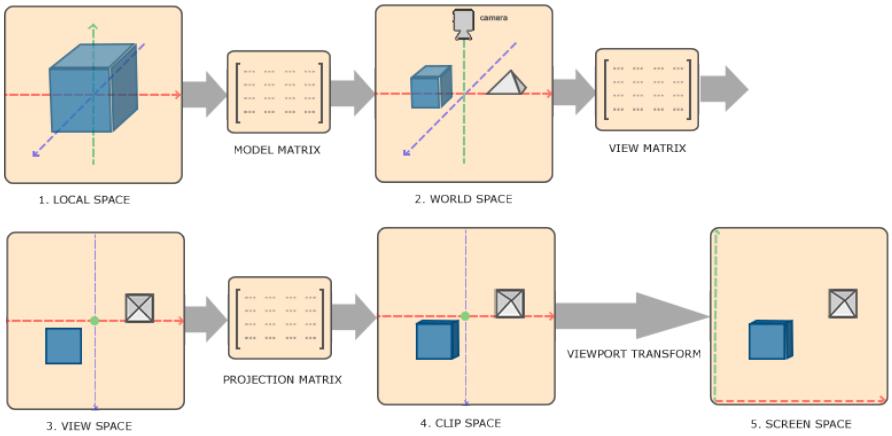

Figure 3: Representation of geometry processing in a standard vertex shader using perspective projection[12].

The rasterization part of the pipeline uses the screen space coordinates computed by the vertex shader to determine the set of pixels covered by the projection of the primitive.

Finally, the colors of the rasterized pixels are determined in the pixel processing stage. The GPU program defining the algorithm used for shading pixels is referred to as the *fragment shader*. The fragment shader is executed in parallell, for each rasterized pixel. With modern GPUs and graphics frameworks, fragment shaders are highly programmable, and different shaders may be used to render different objects.

5.3 Volume Rendering

The traditional raster-based rendering technique is not well-suited for rendering phenomena that are volumetric in nature. Special rendering techniques have been developed to render physically occuring phenomena like smoke, clouds or fog. These techniques may also be used to visualize three-dimensional scientific data, like that captured by fluid simulation or medical imaging.

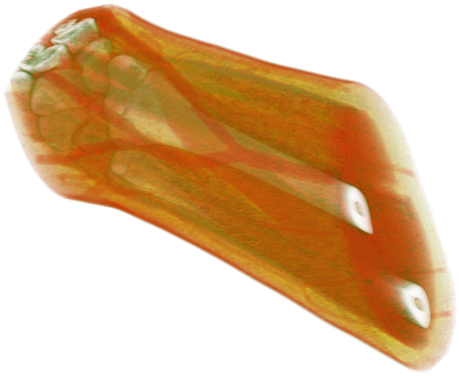

Figure 4: Volume rendering of a CT scan of a forearm

5.3.1 Cut Planes

A simple way to render volumetric data is reduce the problem to two dimensions, by only rendering a single *slice* of the data at once. This can be achieved by rendering a plane placed such that it intersects with the bounding volume of the data set. In the fragment shader, the world space position of each fragment is

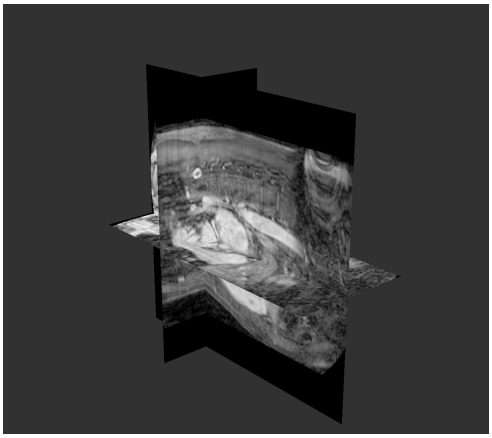

Figure 5: Volume rendering of an MRI dataset using three cut planes.

used to sample the volumetric data set. The sampled value can then be used to determine the final color of the pixel via some transfer function. This method is interpret and implement in the rasterization pipeline, requiring only a simple modification to the fragment shader.

5.3.2 Volume Ray Casting

Volume Ray Casting is a rendering technique used to directly render volumetric scalar fields. Intuitively, a ray is projected from each pixel into the volume, and the scalar field is iteratively sampled along each ray to compute the pixel values. This algorithm approximates the volume rendering rendering integral by computing the Discrete Volume Rendering Integral (DVRI)

$$I(D) = \sum_{i=0}^{n} C_i A_i \prod_{j=i+1}^{n} (1 - A_j)$$

Where I is the accumulated intensity for a given pixel with total ray depth D. C_i and A_i represent the intensities and opacities sampled along the ray, respectively. This can be extended to color images by performing the same algorithm for each channel in the color image (RGB).

If ray-casting is performed front-to-back, the algorithm takes the following form [13]

$$C_{dst} \leftarrow C_{dst} + (1 - \alpha_{dst}) C_{src}$$

$$\alpha_{dst} \leftarrow \alpha_{dst} + (1 - \alpha_{dst}) \alpha_{src}$$

Here, the *destination* color C_{dst} and opacity A_{dst} are computed by *compositing* the result from the previous iteration with the *source* values, sampled at the current ray segment. To sample arbitrary positions with the volume, trilinear interpolation is used. A shading model may also be applied during ray traversal, and this typically requires computing the gradients of the scalar field.

Transfer functions are used to map data values from the 3D scalar field to colors and opacities. The choice of transfer function depends on the desired visualization and may require expert insight into the nature of the underlying data. The steps of the volume ray casting algorithm are demonstrated in figure 6.

This algorithm can be implemented on the GPU by conforming to the traditional rasterization pipeline by using *proxy geometries*. The proxy geometries are placed such that the rasterized pixels cover the projection of the scalar field in view space. Thus, the ray casting algorithm can be implemented directly in the fragment shader used for these geometries.

5.4 Isosurface Rendering

In isosurface rendering, a triangulated mesh is generated directly from the volume data. The mesh is formed by generating vertices at volume cells whose data includes a specific value. This value is refered to as the *isovalue* The

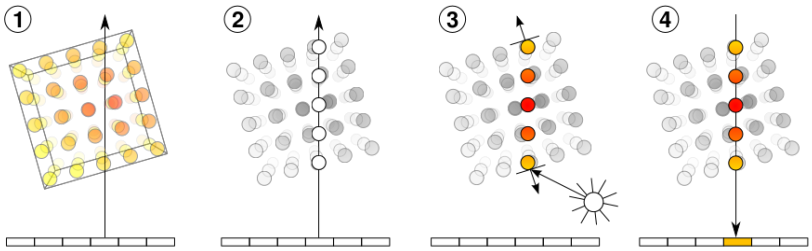

Figure 6: Volume ray casting. The generated ray for a given fragment is shown in (1), passing through the discrete scalar field. In (2), data values are sampled along the ray. Then, a transfer function and shading model are applied at each sample (3) to compute color and opacity values. Finally, the pixel value is computed through compositing (4) [14].

mesh can then be rendered using the traditional rasterization pipeline for rendering surfaces, and lighting models can be applied to achieve proper shading. This method is generally cheaper in rendering cost as the geometry is directly rendered through projection and rasterization rather than by direct volume rendering such as with ray casting.

A well-known isosurface generation technique for scalar fields is the marching cubes algorithm [15]. This algorithm generates coherent 3D surface meshes from 3D scalar fields and is often used for visualization of medical imaging data such as that obtained by MRI. An example visualization is shown in figure 7

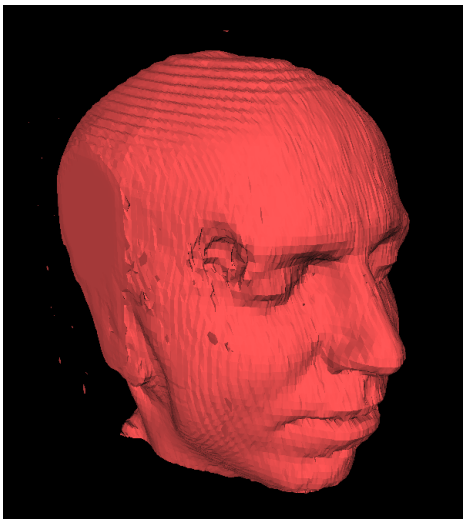

Figure 7: A visualization of a mesh generated by the marching cubes algorithm on MRI data of a human head.

References

[1] S. Omerovic and A. Jain, *Echocardiogram*. Treasure Island (FL): StatPearls Publishing, updated 2023 jul 24 ed., 2025.

[2] K. Lenes, "Apikal4d.gif: 3d-loop echocardiogram from the apex." https://commons.wikimedia.org/wiki/File:Apikal4D.gif, 2008. Licensed under CC BY-SA 3.0; accessed 2025-06-26.

[3] T. Saul, S. D. Siadecki, R. Berkowitz, G. Rose, D. Matilsky, and A. Sauler, "M-mode ultrasound applications for the emergency medicine physician," *The Journal of Emergency Medicine*, vol. 49, no. 5, pp. 686–692, 2015.

[4] B. N. Shah, "Echocardiography in the era of multimodality cardiovascular imaging," *Biomed Research International*, vol. 2013, p. 310483, 2013. Epub 2013 Jun 26. PMID: 23878804; PMCID: PMC3708397.

[5] C. M. Otto, *Textbook of Clinical Echocardiography*. Philadelphia, PA: Elsevier, 6 ed., 2018.

[6] Unity Technologies, "Unity Real-Time Development Platform." https://unity.com, 2025. Accessed: 2025-06-26.

[7] Unity Technologies, "Shader programs - unity manual." https://docs.unity3d.com/6000.1/Documentation/Manual/Shaders.html, 2025. Accessed: 2025-06-26.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009. Available: https://www.ros.org.

[10] National Research Council Canada, "Artag - background and description of the original implementation," 2004. Accessed: 2025-06-26.

[11] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering*. CRC Press, 4 ed., 2019. [Originally published in 2008].

[12] J. de Vries, "Figure from learnopengl: [vertex transforms]." Online, 2025. Licensed under CC BY-NC 4.0. Accessed: 28-Mar-2025.

[13] J. Krüger and R. Westermann, "GPU-based interactive visualization techniques," in *Proceedings of IEEE Visualization 2006*, pp. 13–20, IEEE, 2006.

[14] F. Hofmann, "Volume ray casting," 2011. CC BY-SA 3.0, via Wikimedia Commons.

[15] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, pp. 163–169, ACM, 1987.