Computer-Integrated Surgery II - Checkpoint Presentation

Head Mounted Display Integration for Orthopedic Surgery

Group 17, Zhuokai Zhao

Mentors: Sing Chun Lee, Long Qian, Alexander Barthel, Javad Fotouhi, Dr. Bernhard Fuerst, Dr. Nassir Navab

Technische Universität München

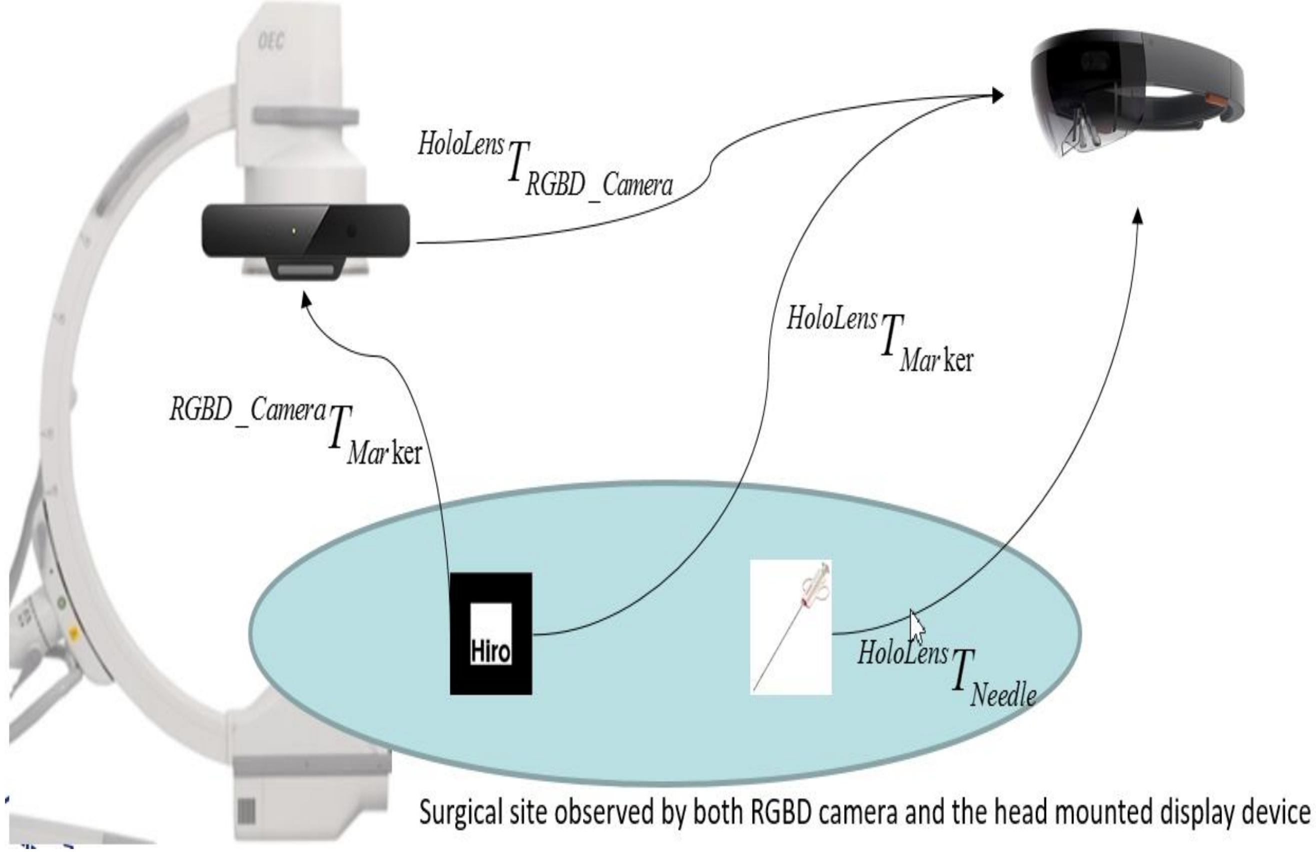
JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Project Recap– HMD in Orthopedic Surgery

- Background
  - Orthopedic surgery requires placements of screws and wires into bones
  - Current 2D imaging guidance system could be confusing and requires lots of X-ray doses
  - Augmented Reality can help!

- Goal
  - Deploying HMDs with simple 2D X-ray views to orthopedic surgery
  - Use augmented reality to visualize the occluded part of the needle
  - Comparison of user’s perception of AR visualizations

Work To Date

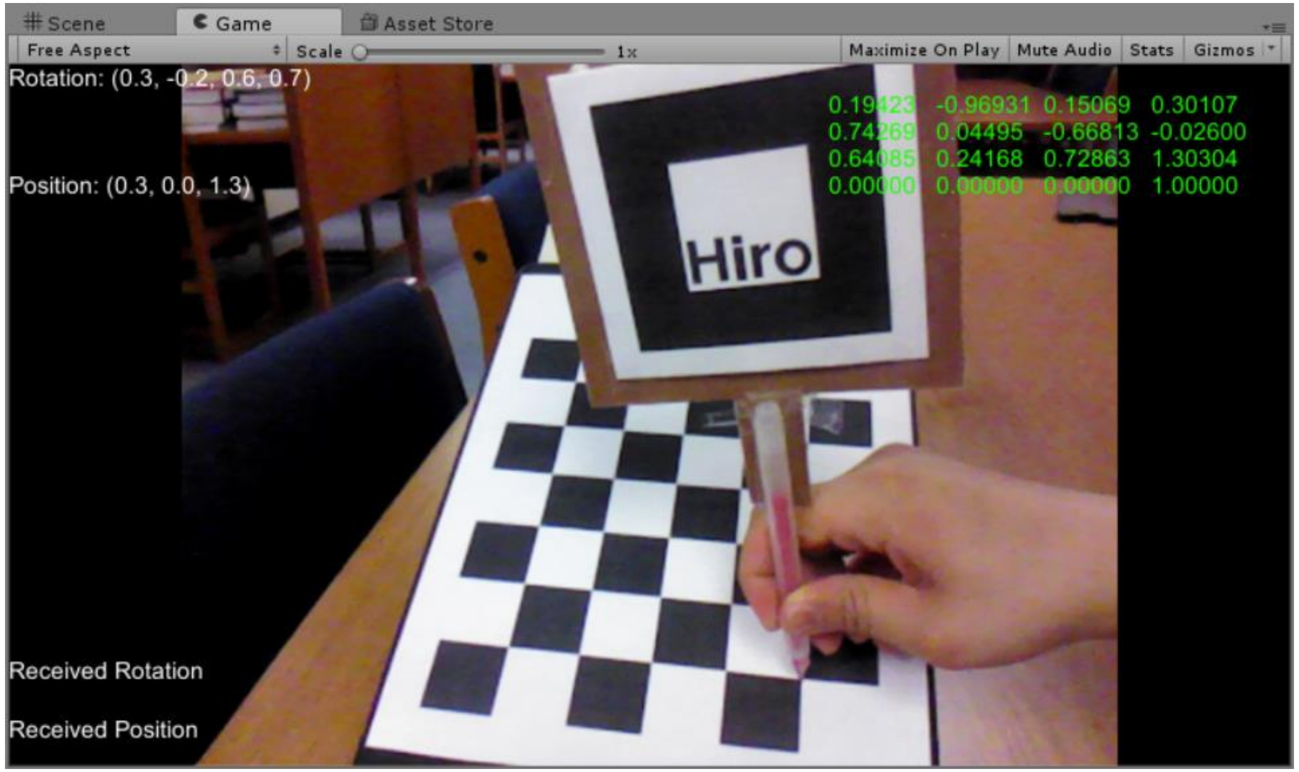


1. **Needle Tracking**
   - ARToolKit
2. **Needle Tip Position Estimation**
   - Pivot Calibration
3. **Camera & HoloLens Communication**
   - User Datagram Protocol (UDP)
4. **Patient Model Surface Detection**
   - Region-Growing Segmentation
5. **Virtual Needle Augmentation**
   - Line Model

1. Needle Tracking

- Marker attached rigidly to the top of the needle/tool
- Implemented tracking algorithm provided by ARToolKit
- Result: Transformation from marker to camera
- Validation follows after pivot calibration

2. Needle Tip Position Estimation - Pivot Calibration

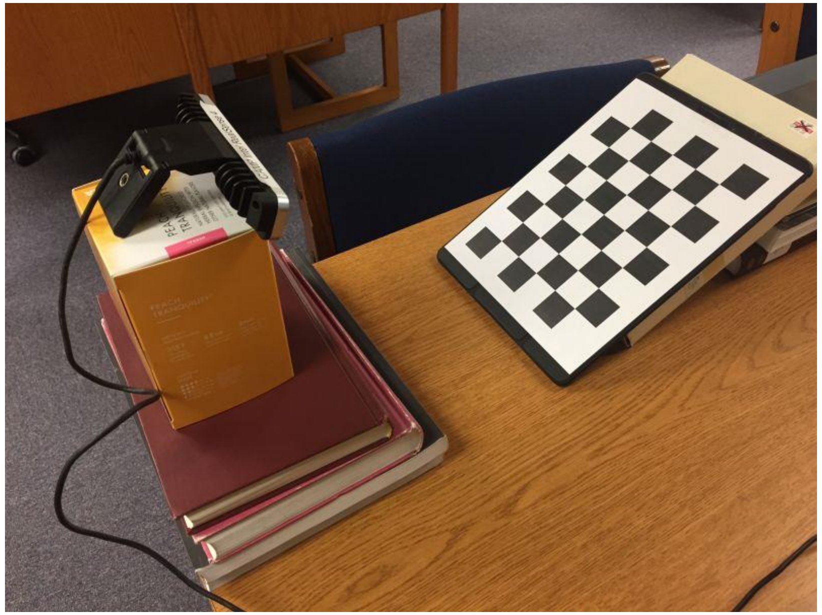


Screenshot taken in Unity

- Known:
  - Transformation from marker to camera
- Unknown/Goal:
  - Tip position with respect to camera
  - Tip position with respect to marker
- Method:
  - Keep the tip position unchanged, move the tool, save multiple marker-camera Transformations
  - Least-Square to solve for tip positions w.r.t. camera and marker

Tracking + Position Estimation Testing

- Testing Method
  - Obtain Camera Extrinsic (coordinate transformations from world to camera) through chessboard camera calibration
    - Reprojection error: ~0.12 pixels
  - Transform chessboard corners to camera coordinate – Set as ground truth
  - Compare results



Tracking + Position Estimation Results

- Results
  - X-direction: ~5.268 mm
  - Y-direction: ~7.843 mm
  - Z-direction: ~6.766 mm
  - Consistent error over different sets of experiments

- Possible Reasons
  - "Ground truth" is not perfect
  - Error from marker tracking
  - Error from pivot calibration

3. Camera & HoloLens Communication

- Inputs:
  - Common marker tracked by Intel camera and HoloLens' front-face camera
  - Transformations from the common marker to Intel camera and HoloLens
- Result:
  - Transformation from Intel camera to HoloLens
- Method:
  - Implemented data transfer using User Datagram Protocol (UDP)
- Problem left:
  - Have not tested accuracy, needs future work

4. Patient Model Surface Detection

- Four Parts:
  - Obtain Point Cloud Data from Intel RealSense F200
  - Compute normal cloud from the point cloud
  - Normal-Based Region Growing Algorithm
  - Color-Based Region Growing Algorithm

Compute Normal Cloud

- Input: point cloud
- Method: Principle Component Analysis

For each point in the point cloud

Assemble the 3x3 covariance matrix of the point and its K nearest neighbors

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p})\cdot(p_i - \overline{p})^T$$

where $\overline{p}$ is the centroid of all neighbors

Choose the smallest eigenvalues and eigenvectors of C

Re-orient the normal to make it consistent to all others. In other words, make sure

$$\vec{n}_i \cdot (v_p - p_i) > 0$$

Change the sign of the normal if needed

End loop

Point Cloud Data Generation

- Input Data: IR and RGB streaming data
- Method:

for each IR(depth) image

    for each pixel

        Get the depth value, convert to meters

        Filter out the current point if value is too large

        Deproject the image pixel to 3D point

        Transform the point from IR coordinate to RGB coordinate

        Project the point w.r.t RGB camera to RGB image pixel

        Check if the pixel is in range

    End loop

End loop

Normal-Based Region Growing Algorithm

- Input: Unsorted point cloud data
- Method:

Sort all the points by curvature, from small to large

Add an unlabeled point with min curvature to a new set called “seeds” and a new list called “Region”

Until no unlabeled point in the point cloud

Until no point in “seeds”

Put its neighbors to “Region” if the normal difference is below **angle threshold**

Put its neighbors to “seeds” if the curvature difference is below **curvature threshold**

Label the current seed as visited from point cloud, remove this seed from “seeds”

End loop

End loop

From all regions, choose all regions with very small values in terms of the x and y of their normal

Color-Based Region Growing Algorithm

- Input: Regions with average normal that has x and y value closed to 0
- Method:

Convert the region results back to unsorted point cloud

Until no unlabeled point in the point cloud

Choose a random point, label as visited. Set the point as a new Region, and a new vector “Points”

Until no point in “Points”

Pop a point from vector “Points”. Find its **K** nearest neighbors

For each neighbor

Calculate the RGB distance if the XYZ distance is smaller than **Distance Threshold**
Add to Region if the RGB distance is smaller than **Point Color Distance Threshold**

End loop

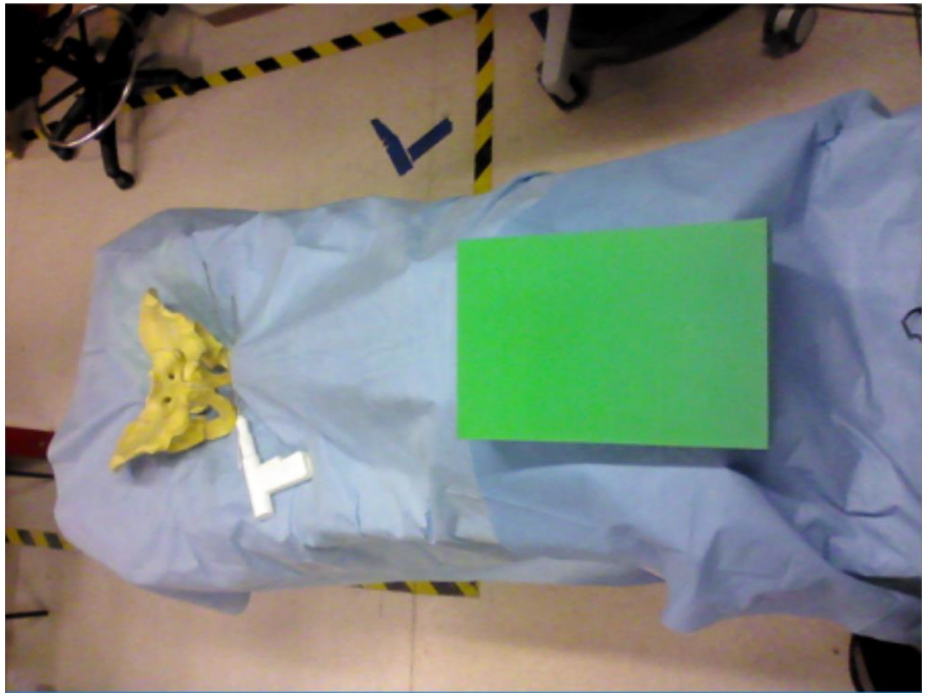
End loop

End loop

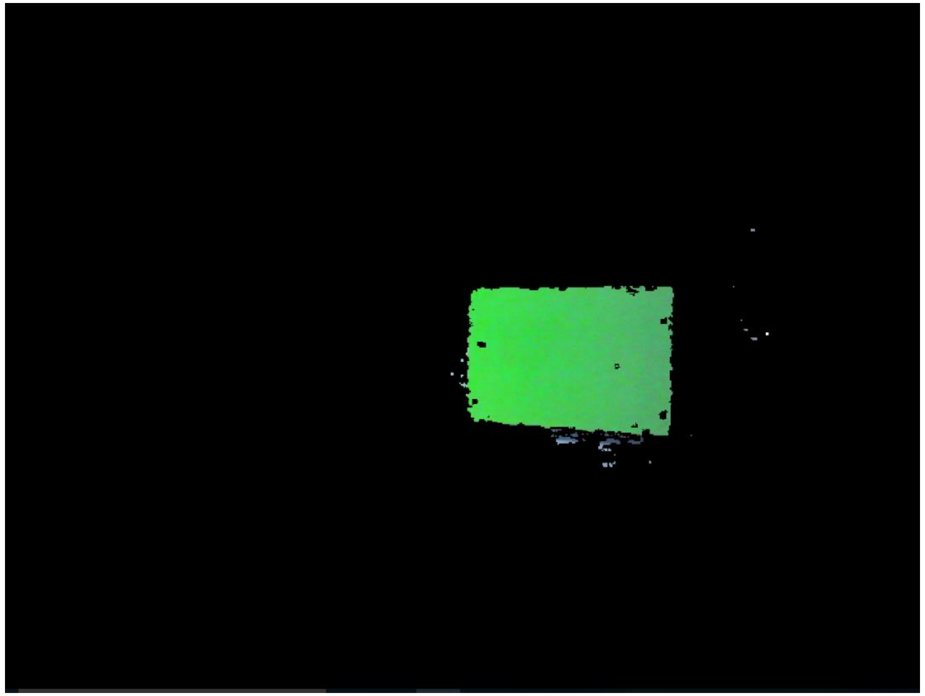
Choose the final region with average R, G, B value in the threshold range

Surface Detection Results

- RGB Streaming



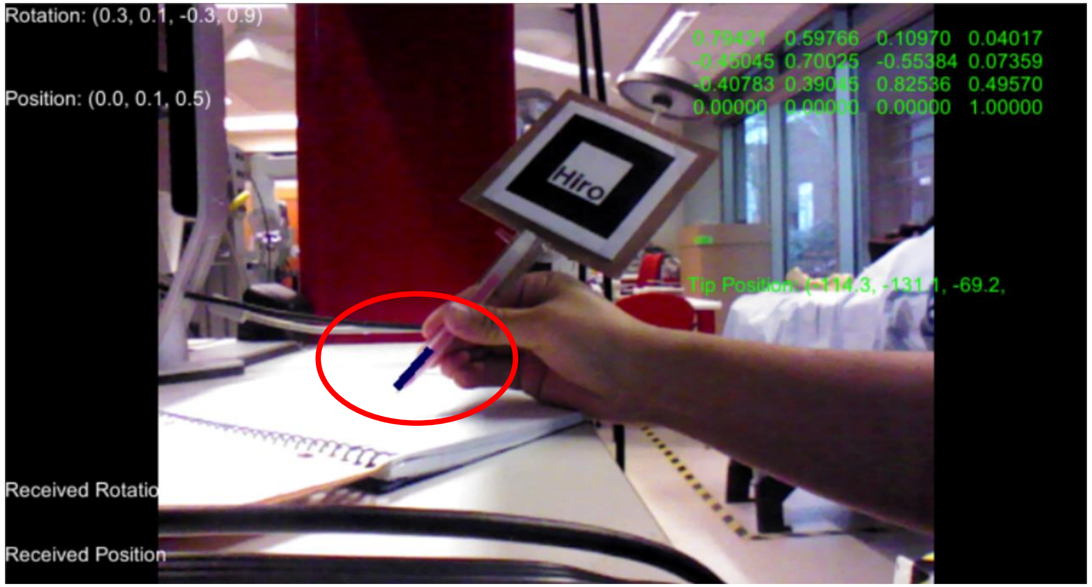
- Surface Point-Cloud Streaming

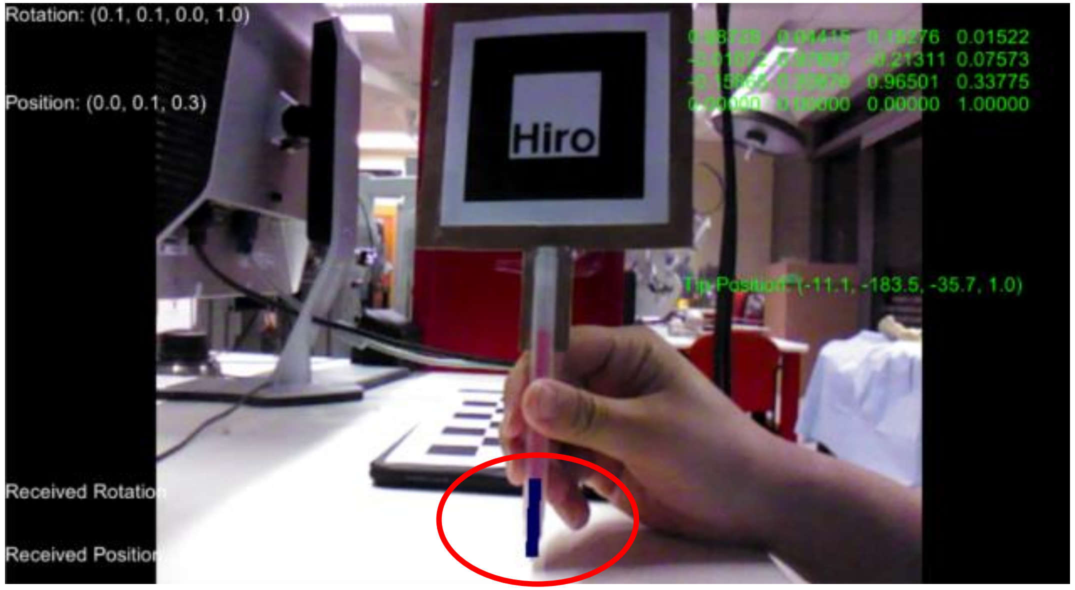


- Photo was taken in Mock OR, with Intel RealSense F200 located vertically above the green box

5. Virtual Needle Augmentation

- **Line Model**
  - Works for different viewpoints, align quite accurately in real-time





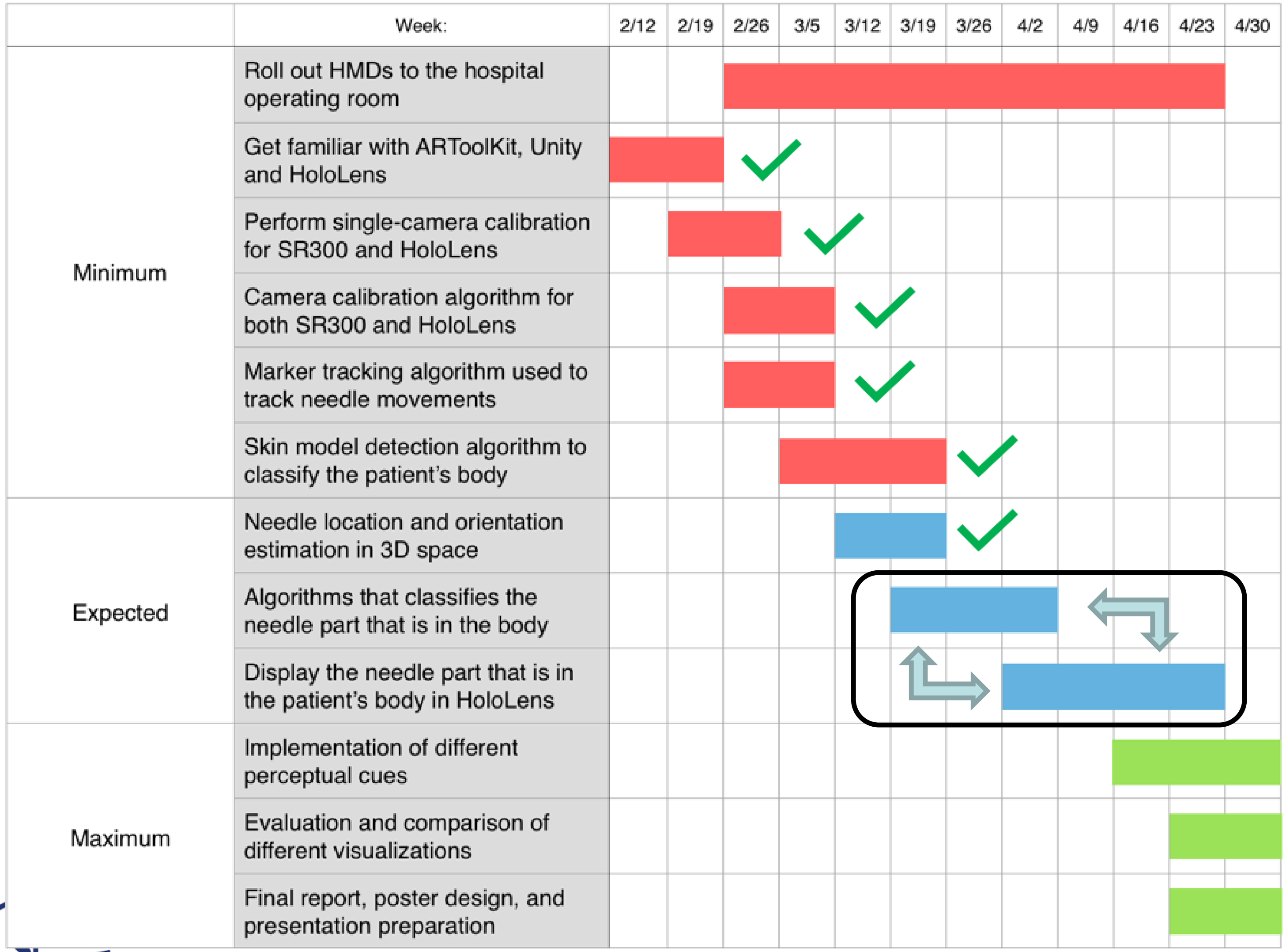
Future Work

- Algorithm to check if the needle is inserted
  - Current idea in mind is to use distance

- Explore using different perception cues to represent the virtual tool
  - Marked entry point
  - Different colors
  - Shadow

- Deploy the model from Unity to HoloLens

Order Change in Schedule



- I think it is better to build the needle model first, and then test the classification algorithm
- Swap the two tasks in the square

References

- Rusu, R. B. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz* **24,** 345–348 (2010).
- Q. Zhan, Y. Liang, Y. Xiao, Color-based segmentation of point clouds, in: Proceedings of ISPRS Laser Scanning Workshop, vol. 38, 248–252, 2009
- Brand, J., & Mason, J. (n.d.). A comparative assessment of three approaches to pixel-level human skin-detection. Proceedings 15th International Conference on Pattern Recognition. ICPR-2000. doi:10.1109/icpr.2000.905653
- Tan, W. R., Chan, C. S., Yogarajah, P., & Condell, J. (2012). A fusion approach for efficient human skin detection. IEEE Transactions on Industrial Informatics, 8(1), 138–147. doi:10.1109/tii.2011.2172451
- Yogarajah, P., Condell, J., Curran, K., McKevitt, P., & Cheddad, A. (2012). A dynamic threshold approach for skin tone detection in colour images. International Journal of Biometrics, 4(1), 38. doi:10.1504/ijbm.2012.044291

Thank you! Any questions?