Senior Design - Team 15
Spring 2025

Vision Based Camera Motion Tracking

THE TEAM

Eric Wittrock

Motion Estimation Solutions

Isaac Kenyon

UI Interface + Automatic Error Detection

Andrew Gooding

Process Automation + Automation Error Detection

Will Ernatt

Camera Tracking Research/ UI Tools and Development

Project Statement

Current camera motion estimation approaches involve lots of manual steps, requiring the user to spend lots of time perfecting the VFX movement.

We want to create a VFX product where users are able to save time and labor, while still resulting in a superior output.

Background

- The VFX industry was $10.60 billion in 2024, expected to be $11.19 billion in 2025.
- Timely, costly, technical, and artistic

Why Camera Tracking?

- Integral part of VFX
- Needed if you want the camera to be moving during shots
- Camera tracking is known to be tedious in the VFX community, good automatic options would be adopted quickly.

https://www.precedenceresearch.com/visual-effects-market

Project Plan

Management

For our project we decided to use a waterfall management style. Waterfall management is a linear, sequential project management style. This force each phase must be complete before the next phases begins. We felt waterfall management was good for our project because:

- ➔ Consists of a concrete pipeline
- ➔ Better suits our project design
- ➔ Helps assess the functionality of each component and address any flaws.

Project Milestones

Reading Frames From Videos

⟶ Should be able to read in and have the ability to process each individual frame

Finding Viable Tracking Points

⟶ The program will be able to look across all of the frames and detect high contrast points to be used as tracking points. The user should be able to change the amount of points with the blender UI.

Persistence of Points Over Time

⟶ Make the point tracker does not have large jumps over points that leave the screen, from camera jitter, or other color contrast problems.

Estimation of Camera Motion

⟶ Calculated output to be applied to object being put with the video. Also is directly affected by the accuracy of persistence of points over time.

Filtering Out Errors

⟶ Available within the blender UI plugin that gives the user an interface to delete certain points out that cause the final render to be jittery.

Improving Readability

⟶ Hopefully displaying the calculated data that our project outputs is displayed such that all of our users can understand. The ability to export to common file formats(CSV, JSON) can let users display that data however they want.

UI Interface

⟶ Once again within blender it will allow new users to use our project without having a lot of background knowledge or being an expert in the field.

Requirements

⟶Solve motion of physical camera when given footage and assign that motion to a virtual blender camera with little to none user intervention. The output of this should not have any jitter between 3d and real scenes

Function

Resource

⟶Users should be able to run software on local hardware with computational and memory requirements similar to a standard pc. The software should easily work as a blender plugin.

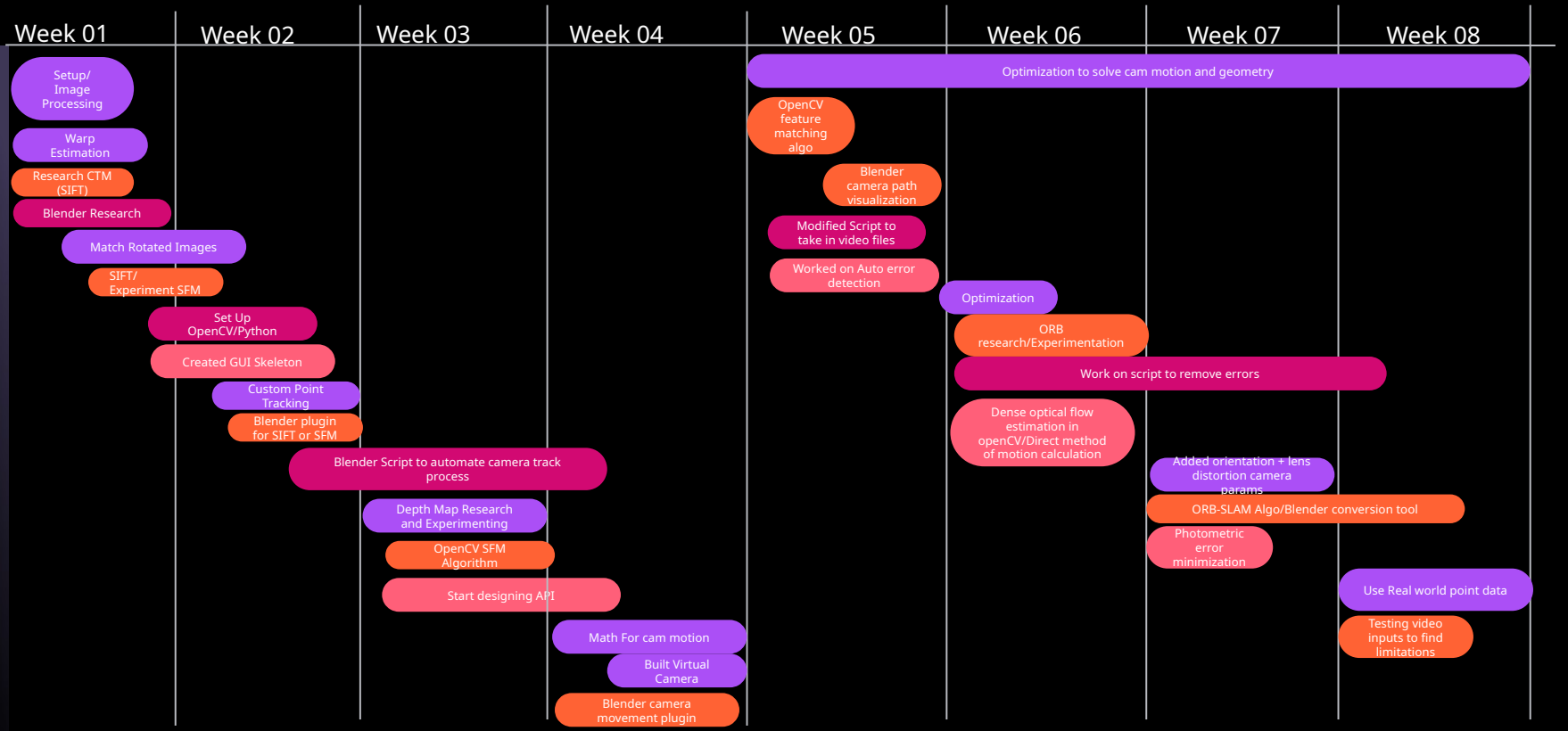
Aesthetic

User Experiential

Minimalistic visual appealing design will be easy for users to learn to use.

⟶Must have simple UI that is easy for users to navigate. Users shouldn’t have any trouble getting past the learning curve.

GANTT CHART


Week 01
Week 02
Week 03
Week 04
Week 05
Week 06
Week 07
Week 08
Setup/ Image Processing
Warp Estimation
Research CTM (SIFT)
Blender Research
Match Rotated Images
SIFT/ Experiment SFM
Set Up OpenCV/Python
Created GUI Skeleton
Custom Point Tracking
Blender plugin for SIFT or SFM
Blender Script to automate camera track process
Depth Map Research and Experimenting
OpenCV SFM Algorithm
Start designing API
Math For cam motion
Built Virtual Camera
Blender camera movement plugin
Optimization to solve cam motion and geometry
OpenCV feature matching algo
Blender camera path visualization
Modified Script to take in video files
Worked on Auto error detection
Optimization
ORB research/Experimentation
Work on script to remove errors
Dense optical flow estimation in openCV/Direct method of motion calculation
Added orientation + lens distortion camera params
ORB-SLAM Algo/Blender conversion tool
Photometric error minimization
Use Real world point data
Testing video inputs to find limitations

Design Process

Question:

Which features to track?

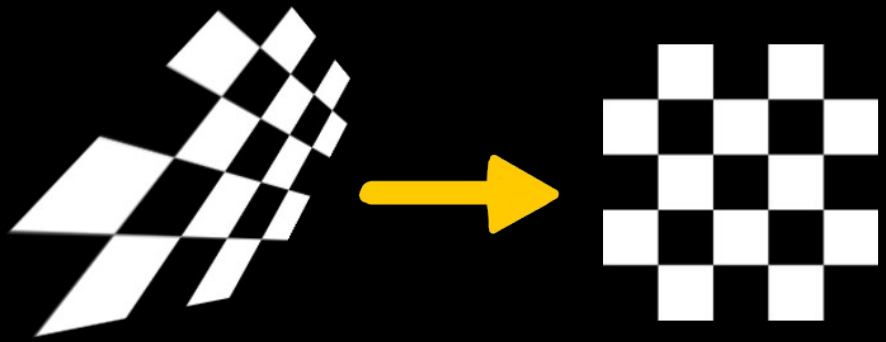
Point

Plane

Edge

Question:

What can and can’t ML do?

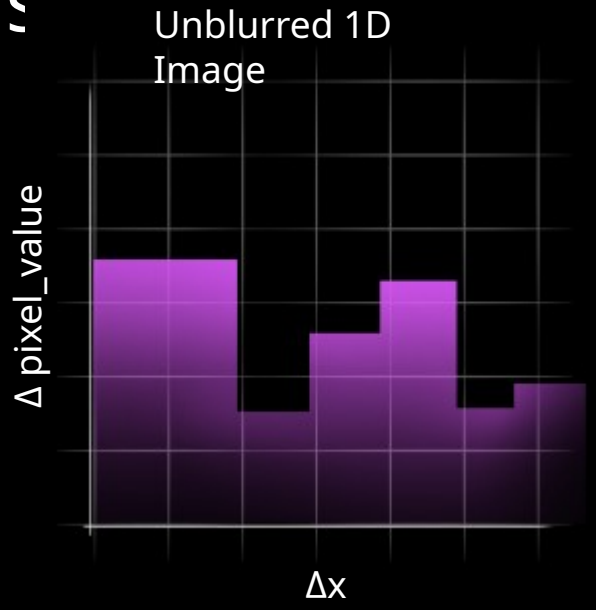


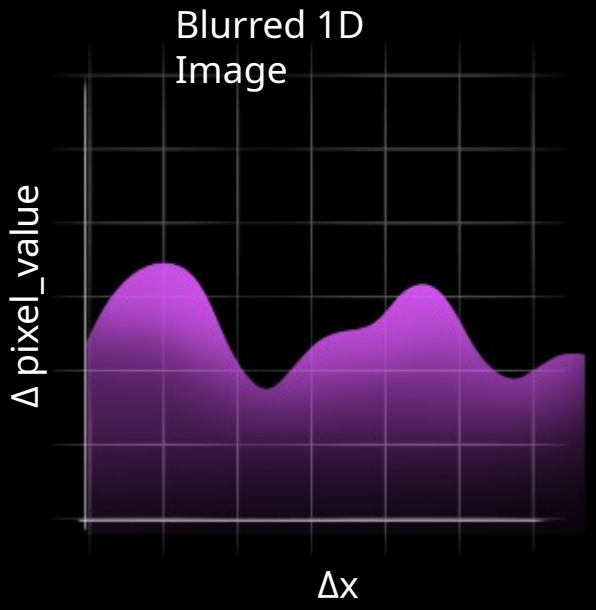
Can it unwarp perspective distortion?

If we know how the perspective changed, we know how the camera moved

Hypothesis:

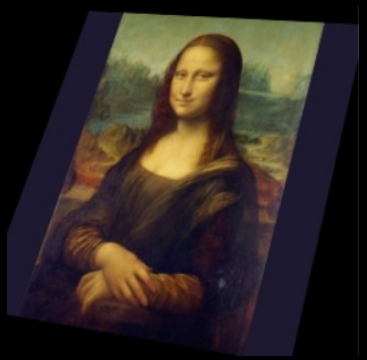
If we blur an image, it will be easy to undistort





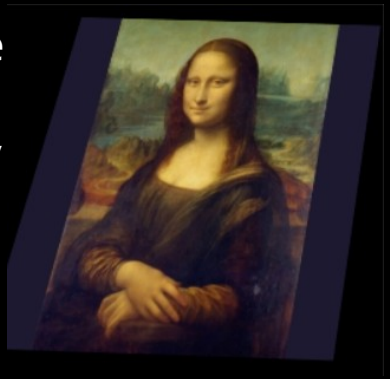
Conclusion: Blurring has no effect and estimated doesn’t match synthesized

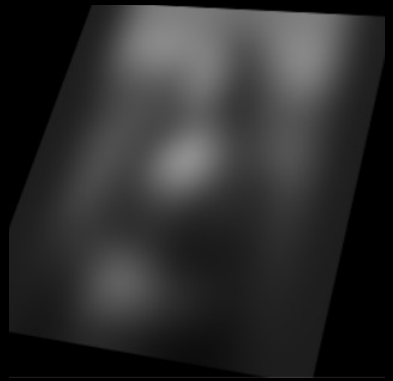
Original



Estimated perspective

Synthesized perspective



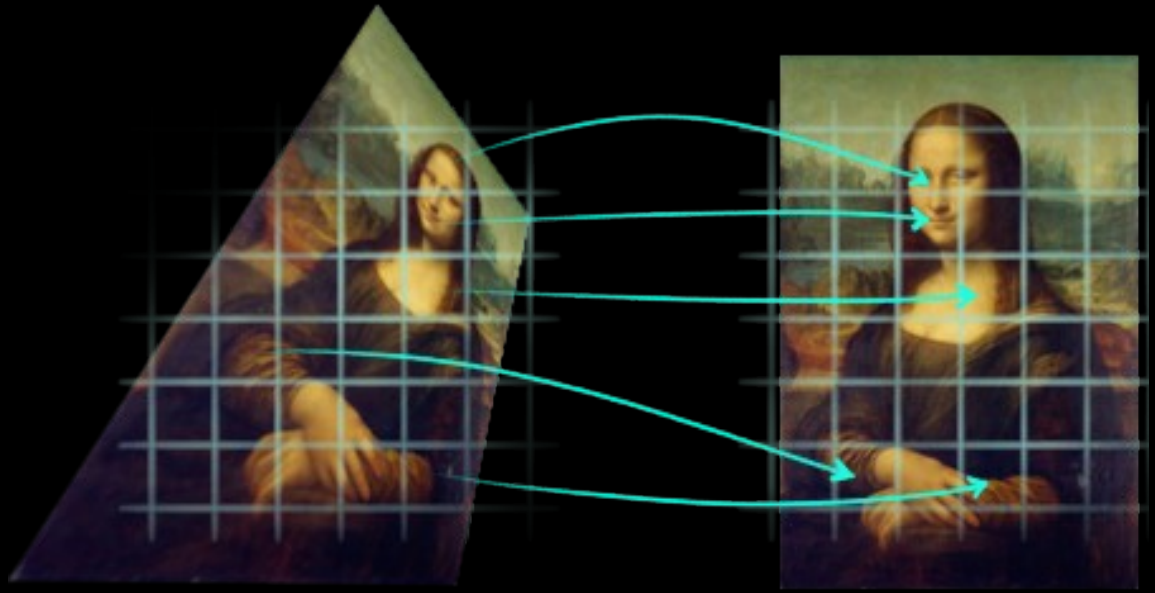


Blurred

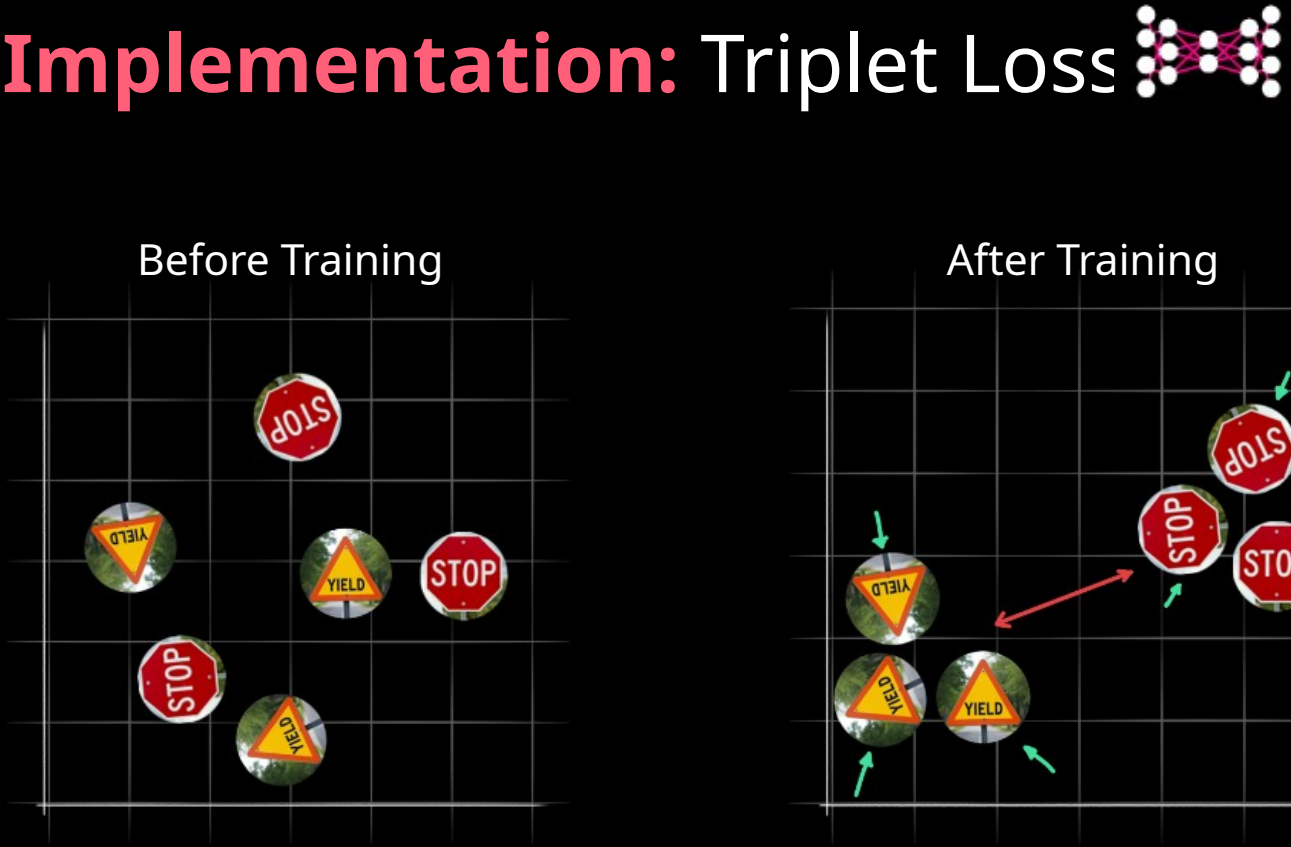
Hypothesis:

Only rotation must be estimated

As segment size approaches zero, the perspective transform becomes negligible

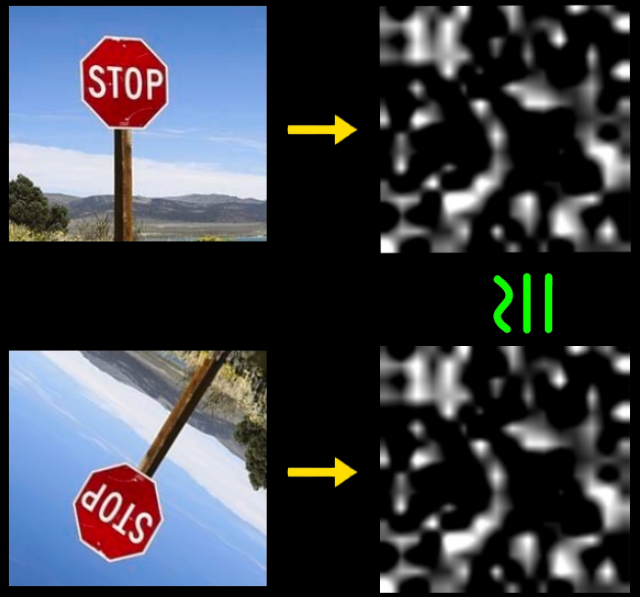


Implementation: Triplet Loss



Conclusion:

Success, but slow performance



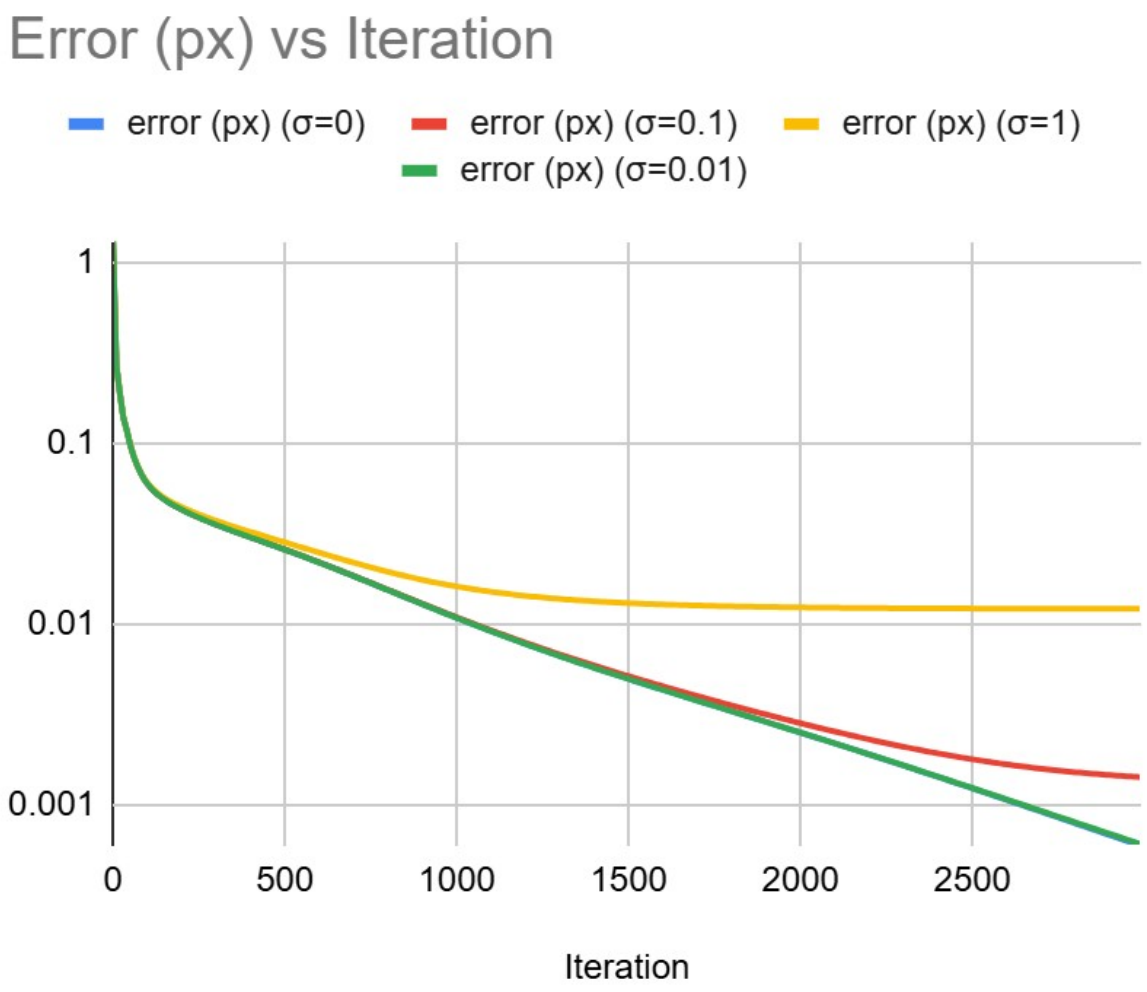
Hypothesis:

OpenCV’s sparse optical flow for point tracking will be sufficient

- With a large number of points, the error in the camera position will be low.
- If optical flow does not work, it can be used as a temporary solution while working on the motion solving algorithm

Conclusion:

Optical flow is not precise enough



Hypothesis:

Geometry *and* motion can be solved simultaneously with an optimization algorithm

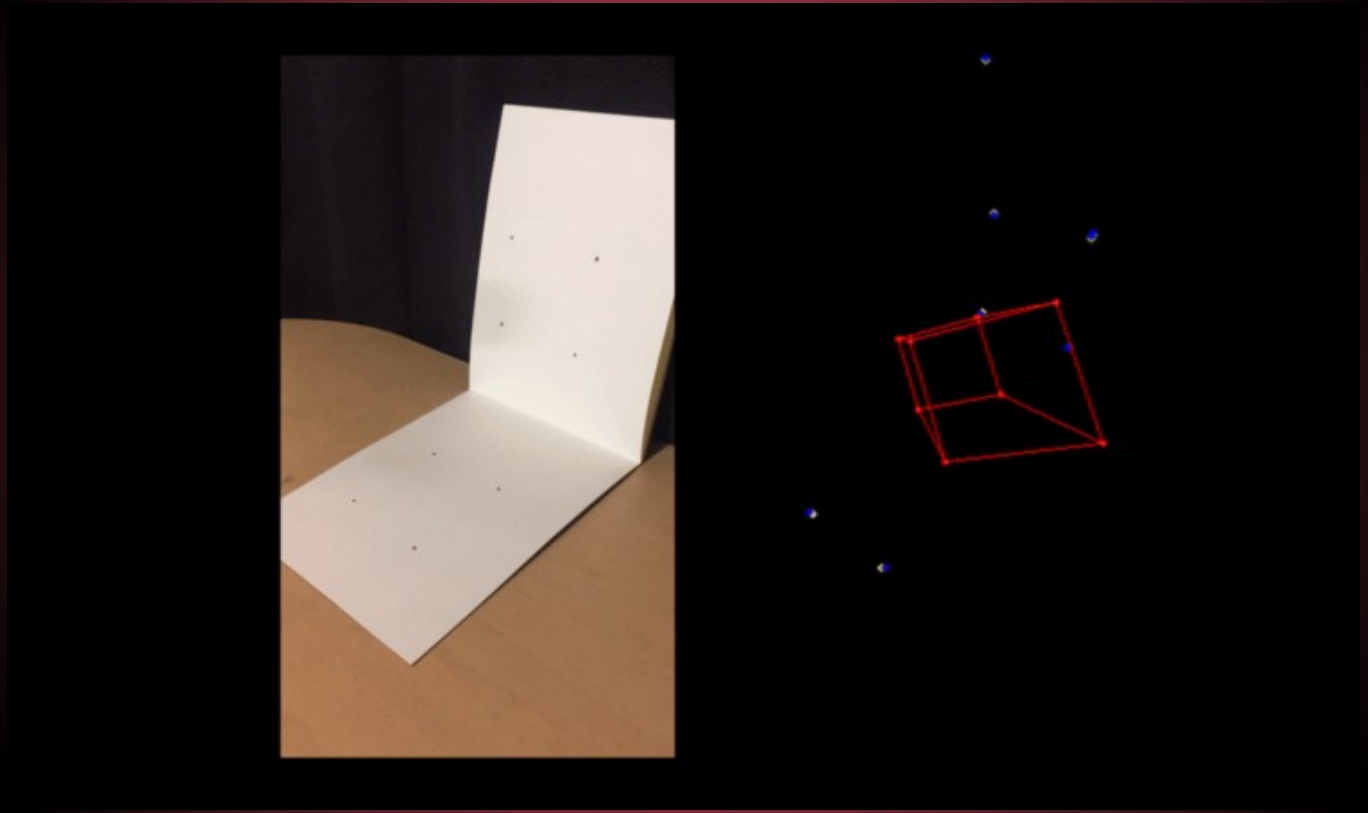
Conclusion:

The approach exceeded expectations. This will be the basis of the algorithm

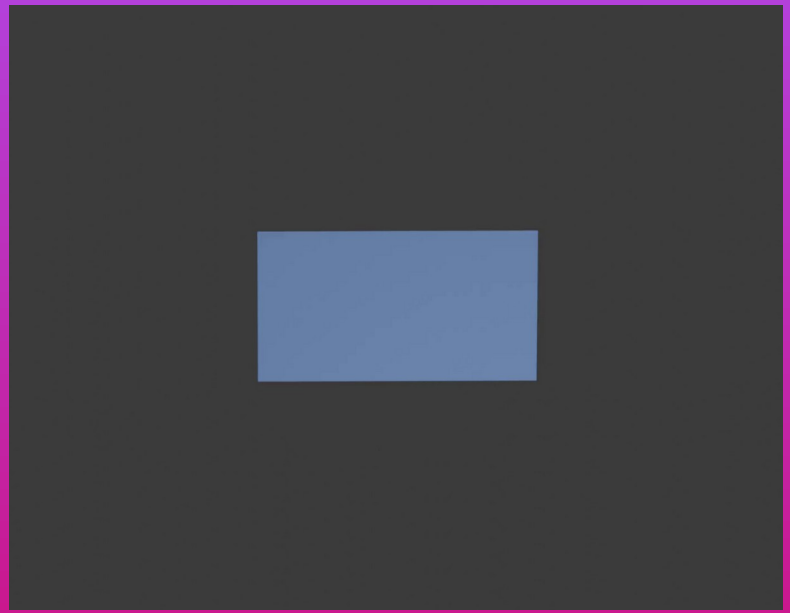
Optimizing Geometry

Optimizing Motion

Demo



Implementation



Blender

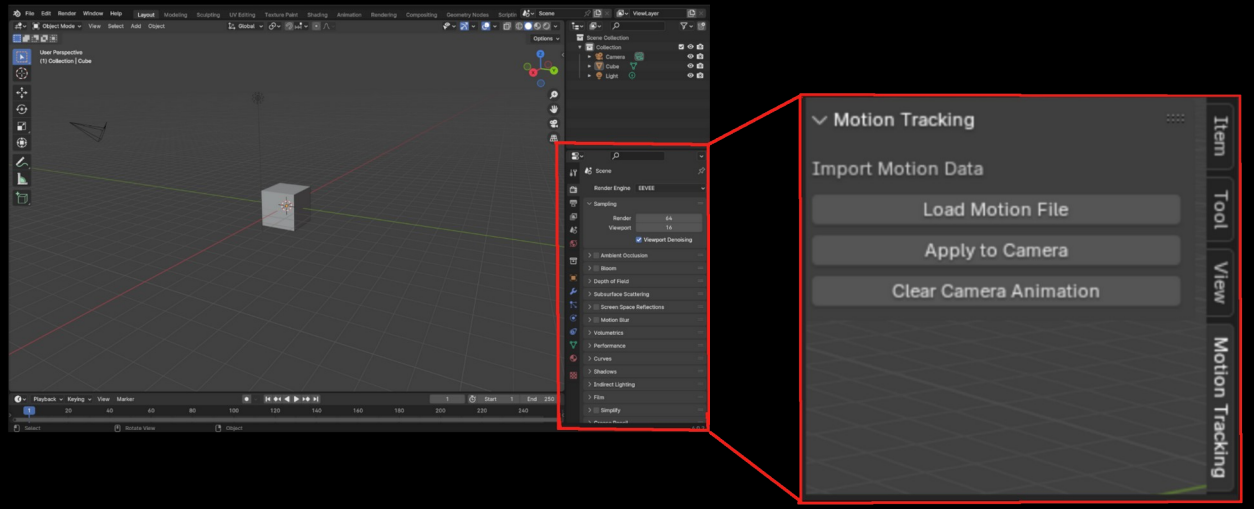
3D Computer Graphics Software

- The interface where users will interact and create with our product.
- Features plugin scripting so our product can be integrated into current and future developments.
- Will provide an easy to use interface for the camera tracking system.
- Less manual tweaks while providing parameter settings for users that want more control.
- Already very common among professionals and amature VFX artists.

Blender UI

↘

- The interface will show up on the right side along with blenders built in properties.
- Will feature import, export, apply, and reset buttons.
- A menu option for more advanced options will be available for users who wish to use them.

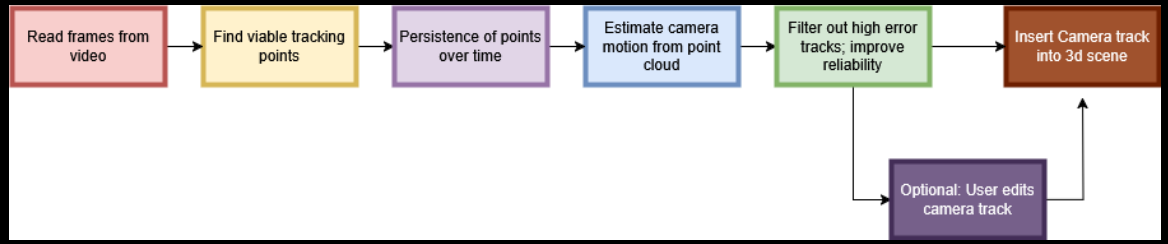


Testing

Testing Strategy

↘

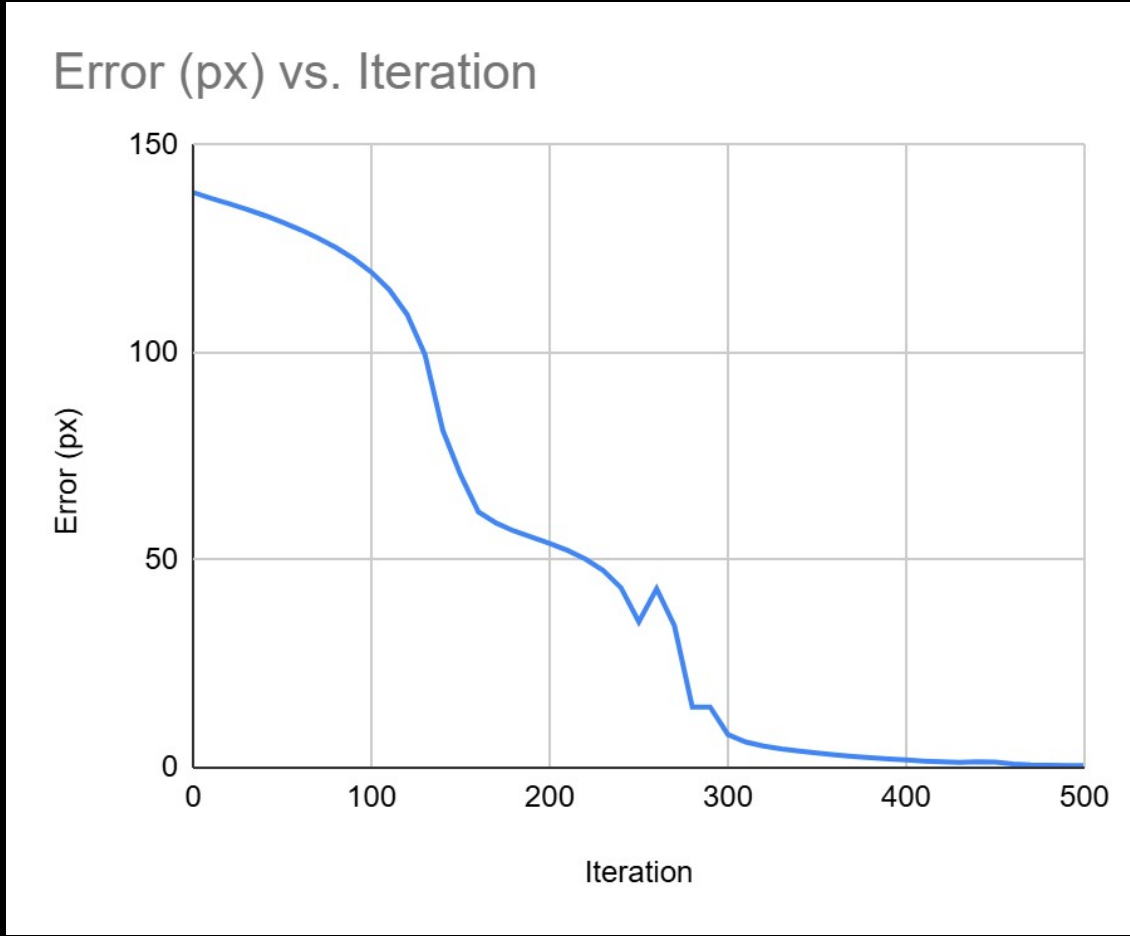
- We need to be confident that each stage of the pipeline is functional both on its own and as part of the whole system.
- Performance metrics must be met, tasks completed in timely manner.
- Test early and test often.



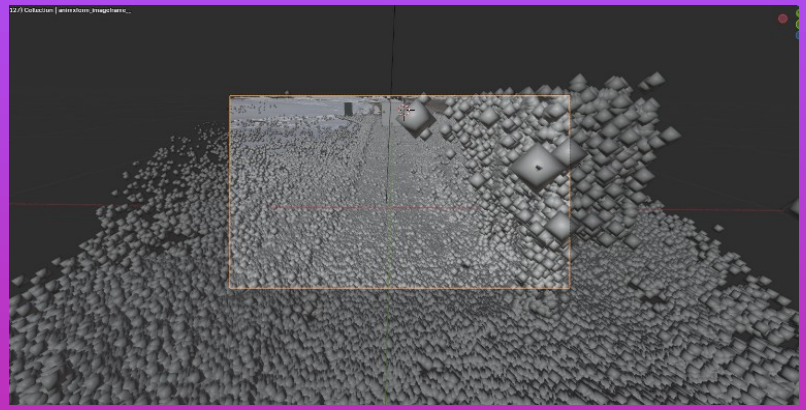
Tools/ Results

↘

- UI_simulate, PyTest, Real User Testing.
- Helped us realize the efficacy of the approach using Adam Optimizer Algorithm.



Conclusion



Our Work So Far

- **The Goal:** Reduce the tedium of the camera tracking process with an easy to use plugin, appealing to all users.

- We’ve shown this is achievable, with prototypes for most main steps of the pipeline.

- Many steps have yet to be integrated with each other.

What’s Next

- Meet performance metrics. Should not take longer than ~15 minutes.
- Explore more complex scenes.
- Package product and release for Blender Extensions

What’s Next

ORB-SLAM for initial first guess

⟶Used for understanding the environment in robotics

⟶Not accurate enough for VFX, but can be used as a starting guess for the optimizer

ML to assist the optimizer

⟶Camera motion is predictable

⟶It is possible to reconstruct geometry from a still image with ML

Add more constraints to prevent skewed geometry

⟶There are infinite solutions to the inverse of the camera equation. Some of these result in skewed geometry

⟶Adding more constraints to the equation can prevent this

⟶Depth estimation or better point sampling are two possible solutions

THANK YOU

Any questions?