Real-time Calibration for Stereo Cameras

LUCID

Sheldon Fernandes

May 23, 2018

Index

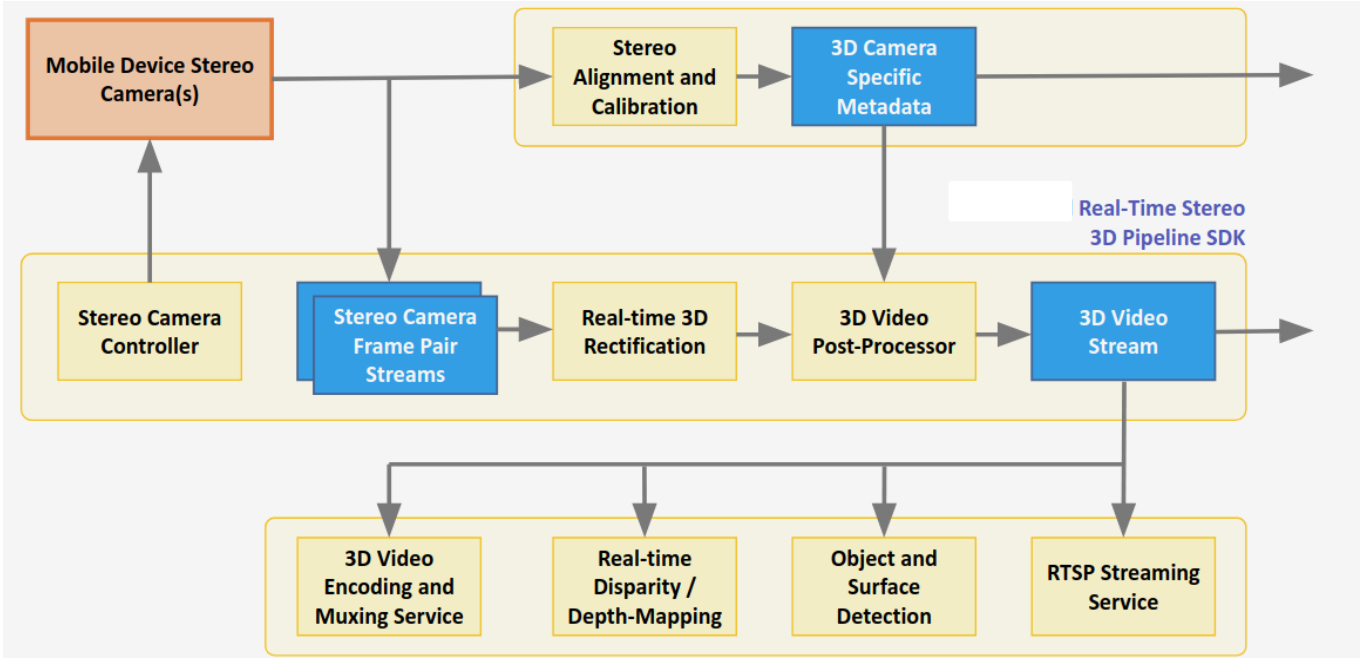
- Importance of Calibration
- Overall flowchart
- Parameters – Extrinsic and Intrinsic
- Factory Calibration – Fixing Errors
- ML Model
- Realtime advanced processing and rendering

Importance of Calibration

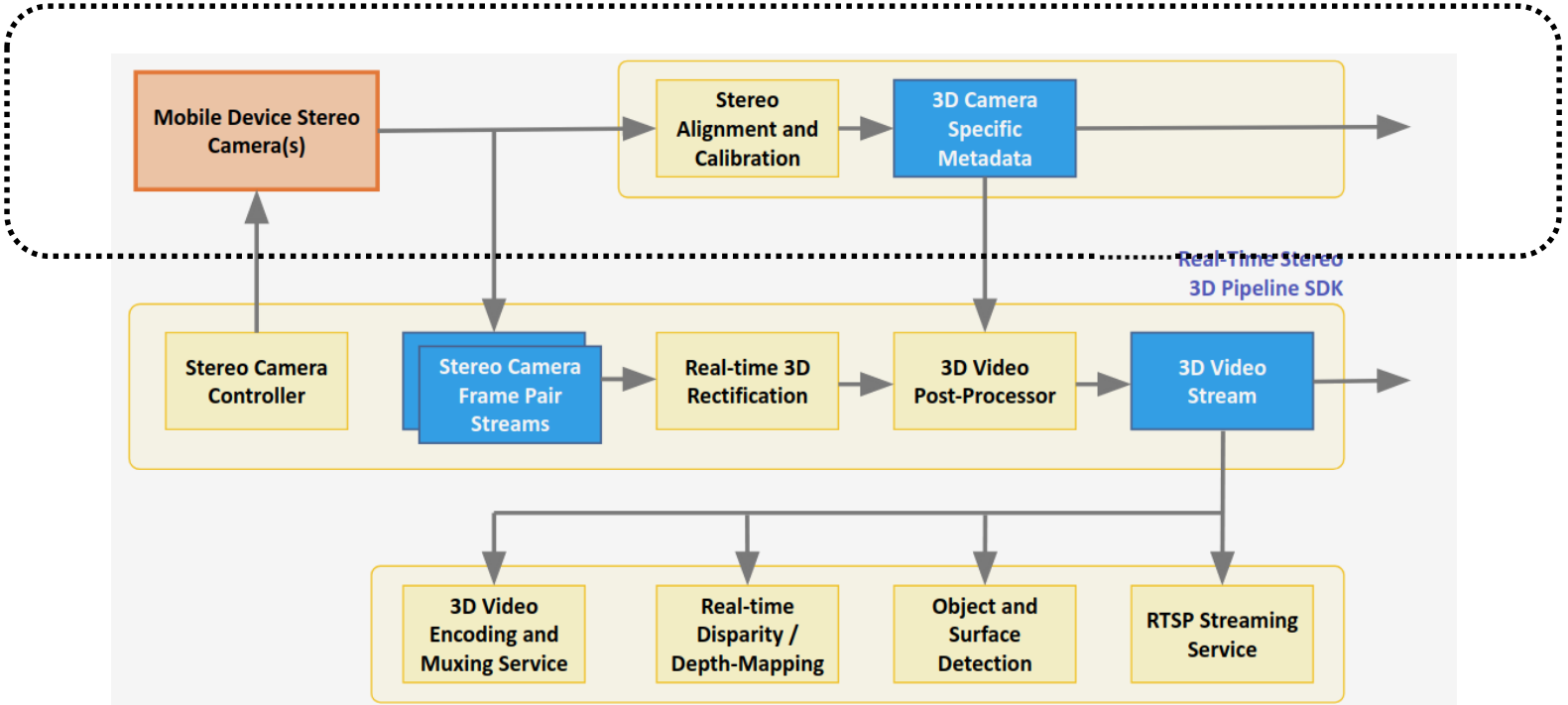
Importance of Calibration

- Calibration is an important process in making sure a camera's output is as close as possible to what it sees.
- Calibration for a stereo pair of cameras is even more critical because the process also involves getting data relative to each camera's position.

Complete SDK Flowchart



Flowchart - Calibration



Intrinsic and Extrinsic Calibration Parameters

- Intrinsic
  - Focal Lengths, Camera Centers, distortion coefficients

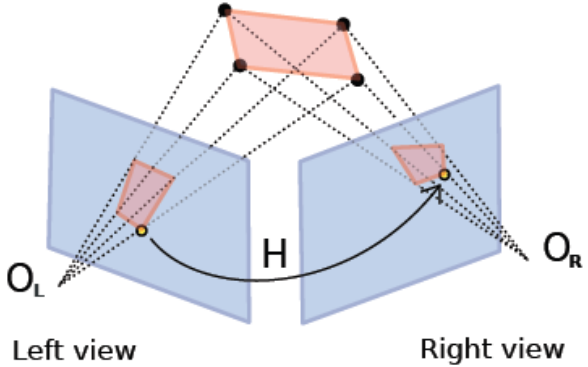
Uncalibrated Video Frame

Calibrated Frame

  - Ensures that each frame is undistorted

Intrinsic and Extrinsic Parameters

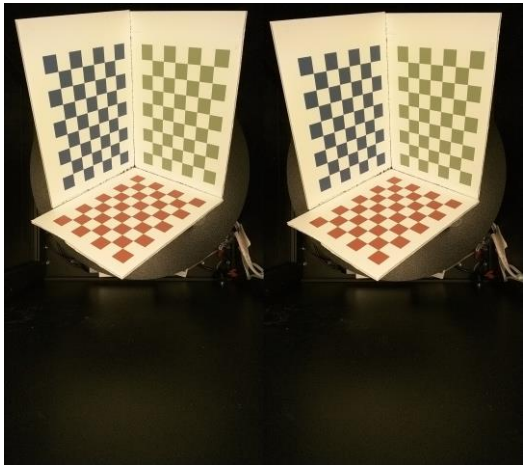
- Extrinsic
  - 3D Rotation matrices, Translation vectors



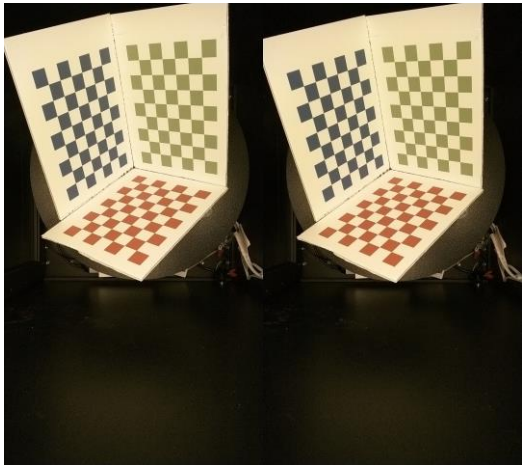
- Ensures 3d data has no rotations, and can be perfectly rectified for viewing, and further advanced processing

Factory Calibration

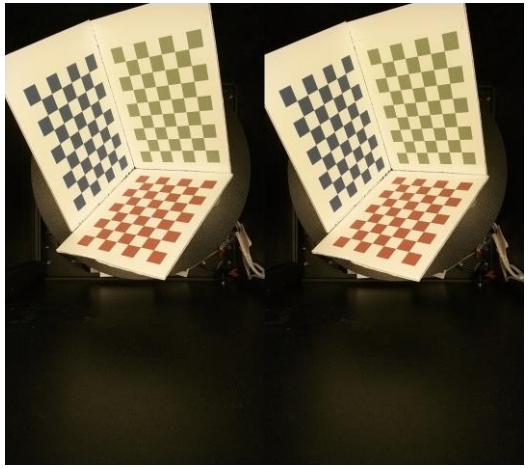
- Use calibration fixtures.
  - Extract pattern data



0 degrees

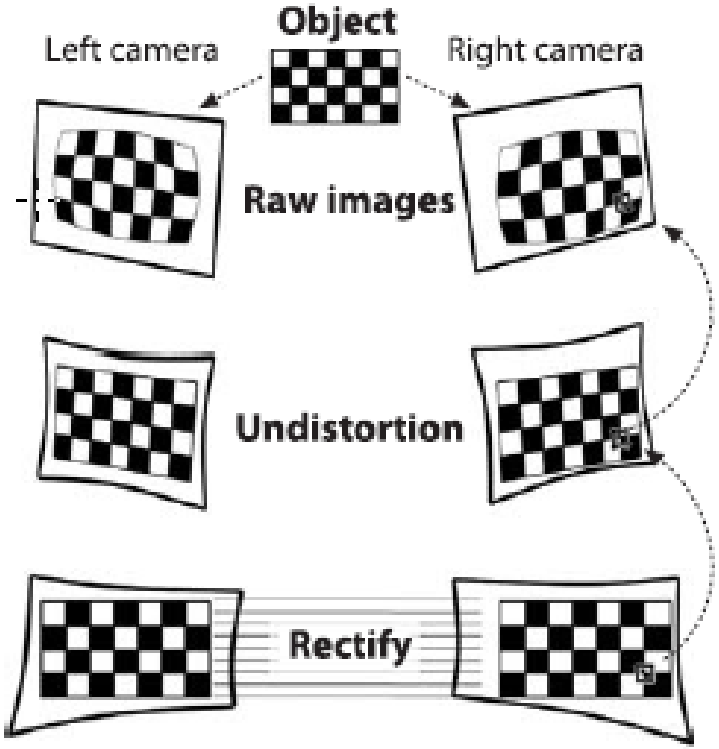


9 degrees



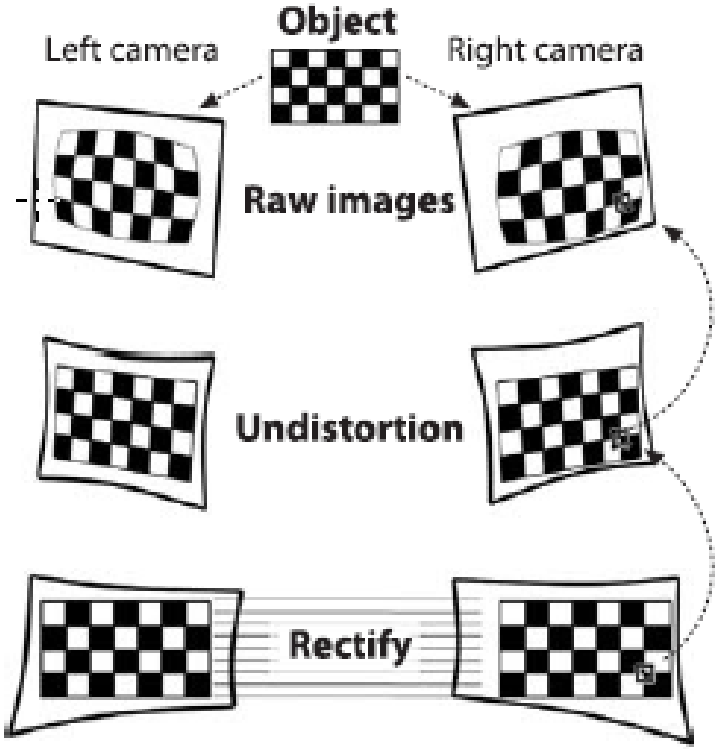
18 degrees

Factory Calibration



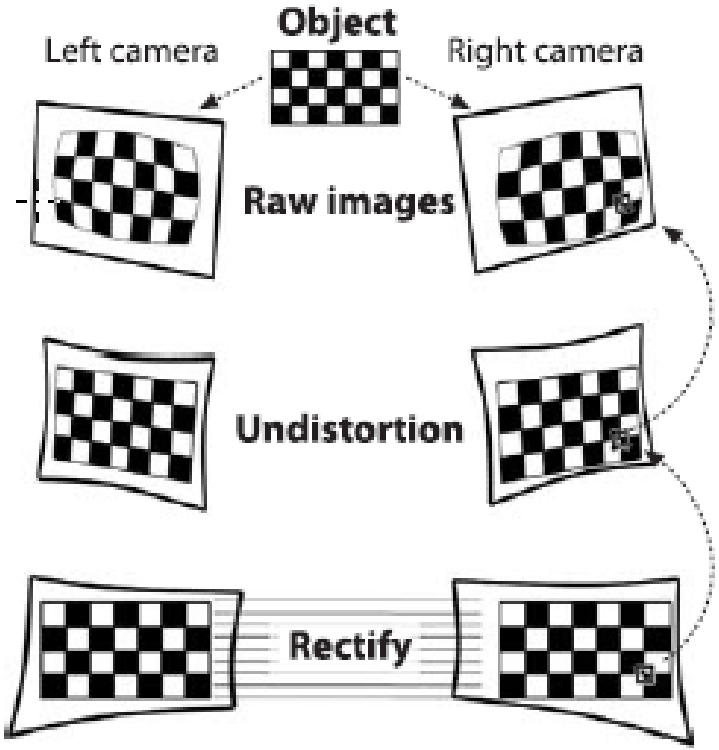
Capture raw input distorted frames.

Factory Calibration



Process data from the cameras to make an intrinsic parameter guess.

Algorithm (Factory Calibration)



Compute extrinsic parameters to get a rectified output, and calculate "reprojection error"

Algorithm (Factory Calibration)

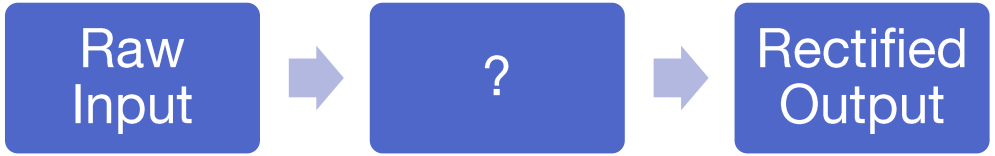
- Use all pattern data to compute the reprojection error

```
If(reprojection_error < 1 pixel) {
    Cout << "grab a beer" << endl;
}
```

- Encode this data for further rendering using GPUs.

Model using machine learning techniques.

- Use rectified data to extract feature points
- From all features, use only straight lines features.



Model using machine learning techniques

Rotation Matrix

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Yaw

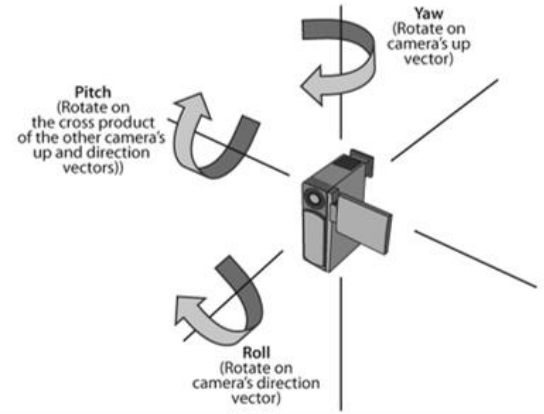
$$\alpha = \tan^{-1}(r_{21}/r_{11}),$$

Pitch

$$\beta = \tan^{-1}\left(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2}\right),$$

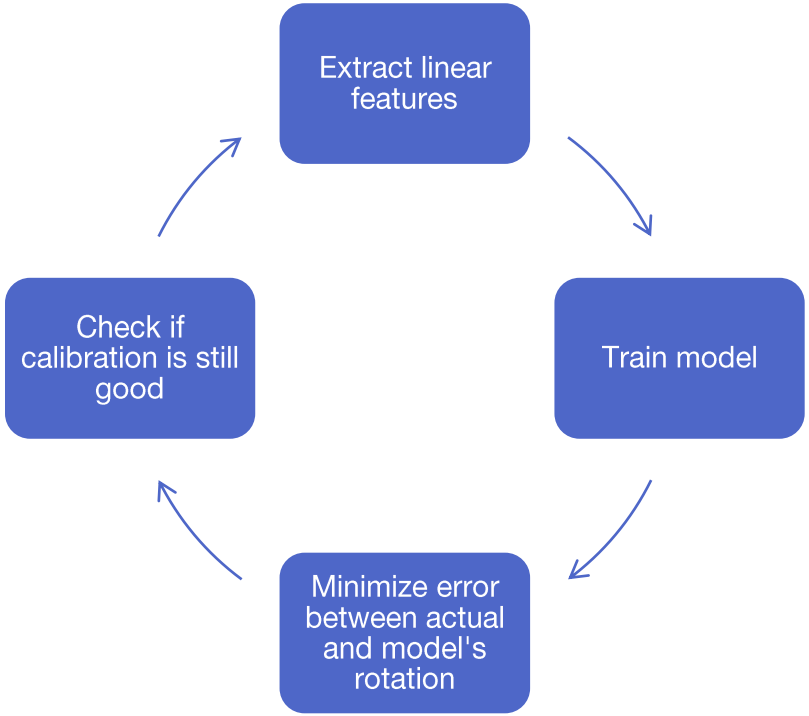
Roll

$$\gamma = \tan^{-1}(r_{32}/r_{33}).$$

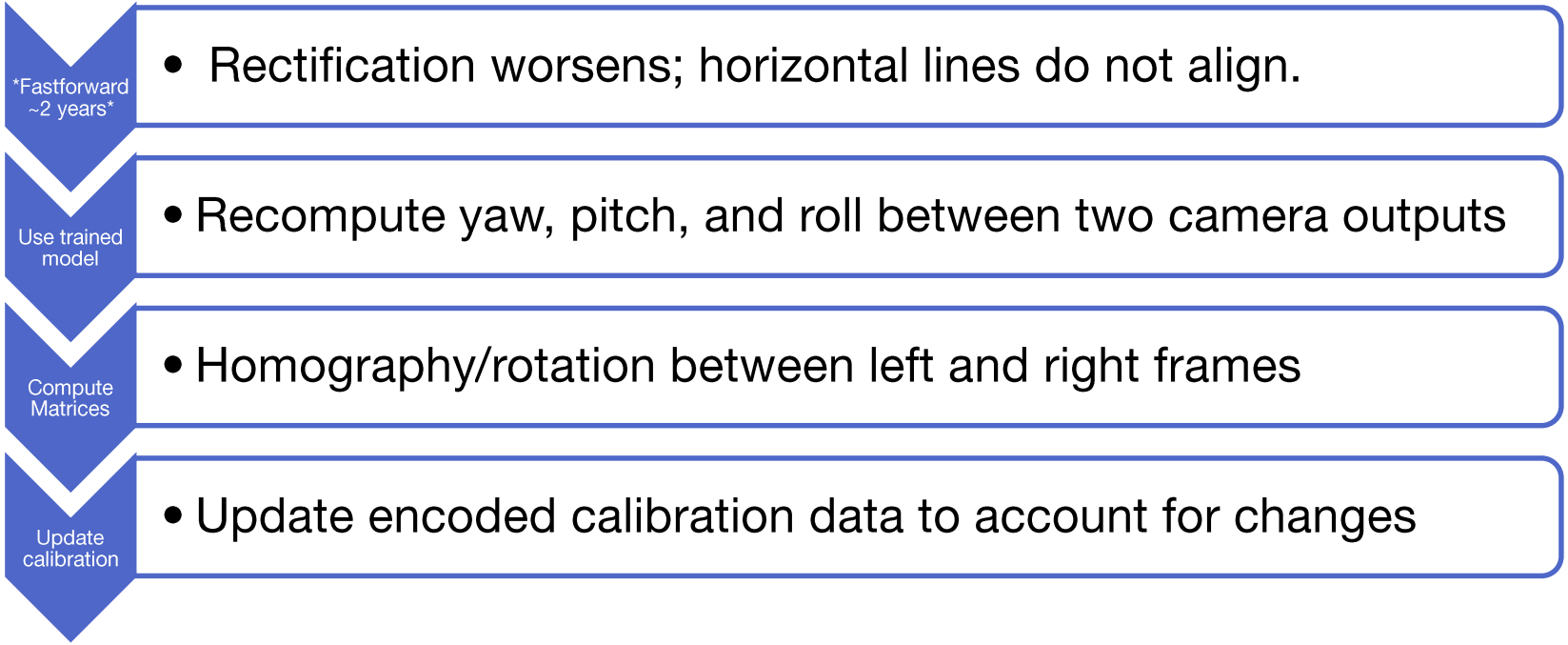


Train a model using good features to give the same yaw, pitch, roll as factory calibration?

Model using machine learning techniques (training)

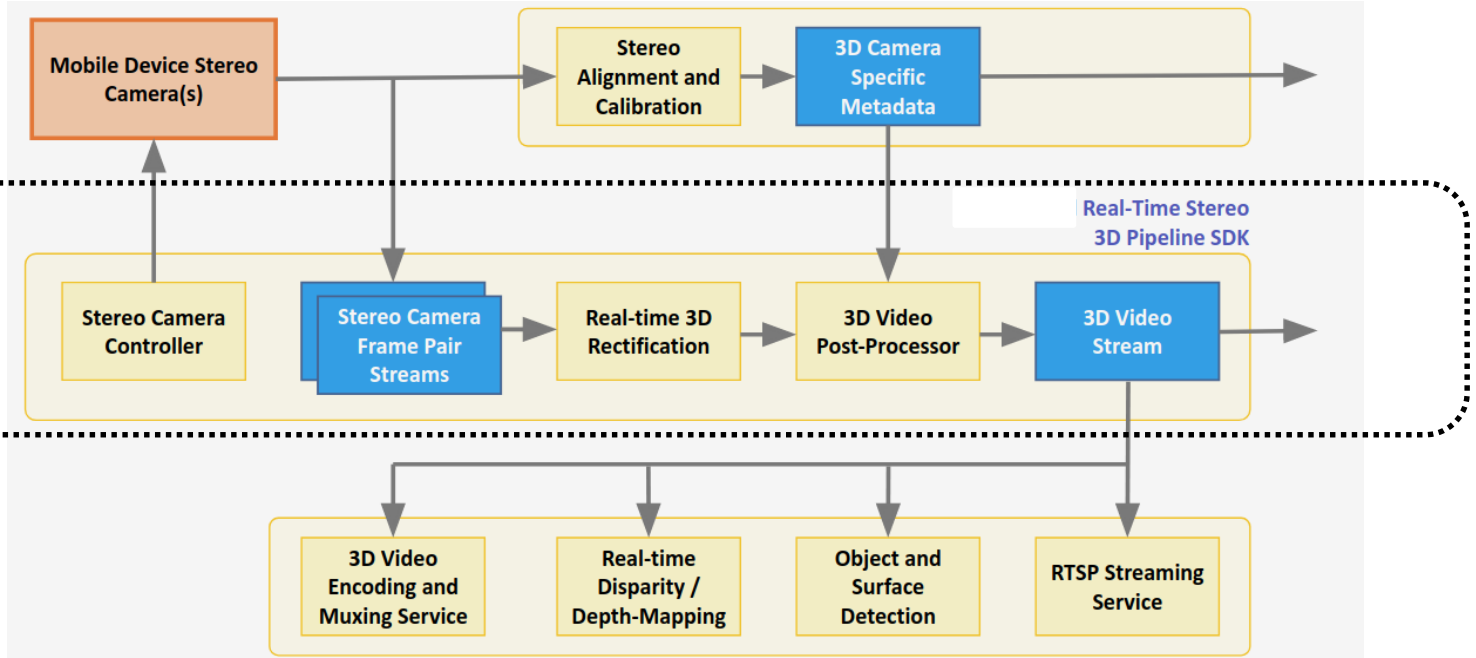


Model using machine learning techniques (usage)



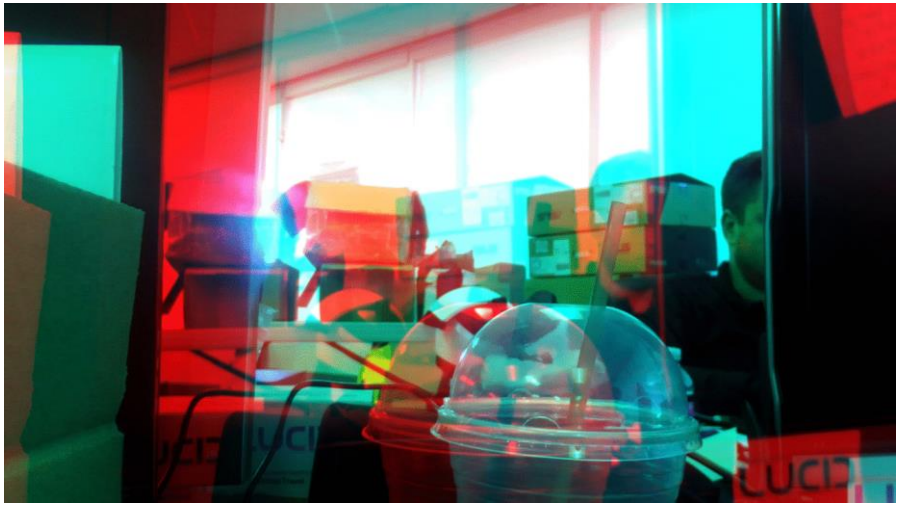
Goal is to finally make a stereo camera as close to the human eye that is self-calibrating.

Flowchart - Rendering

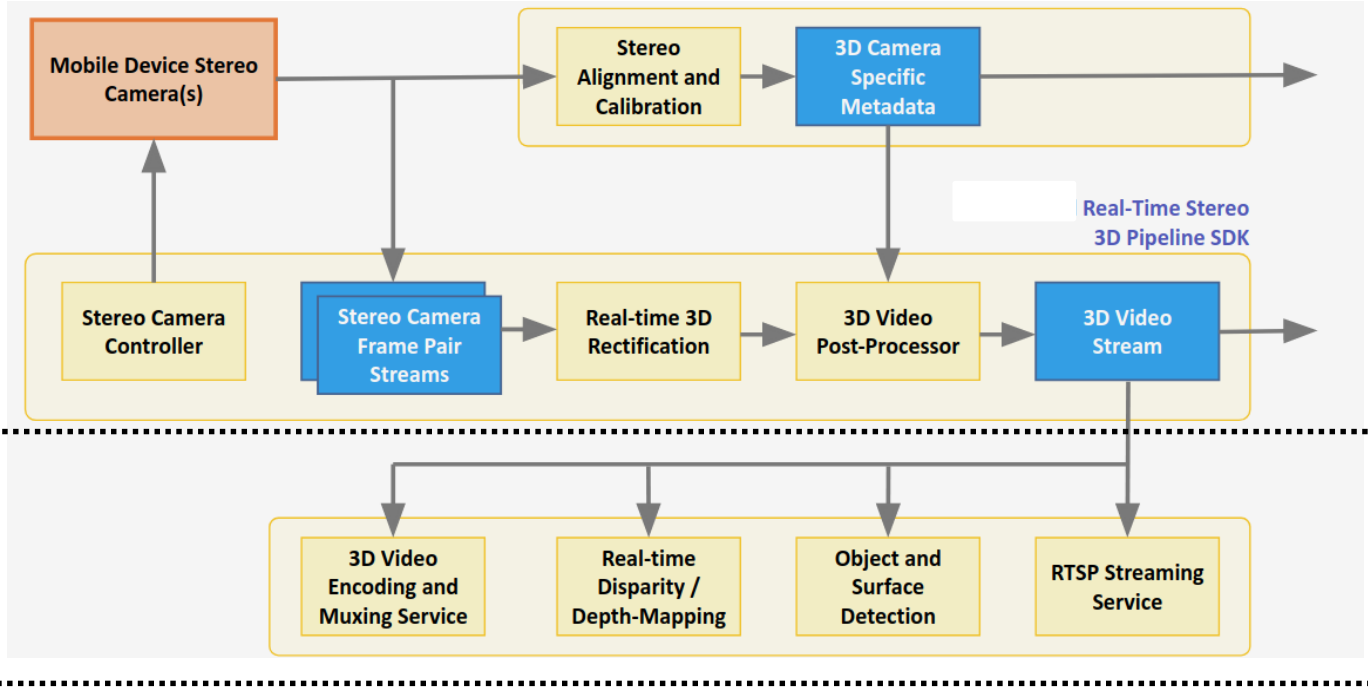


Rendering using calibration data

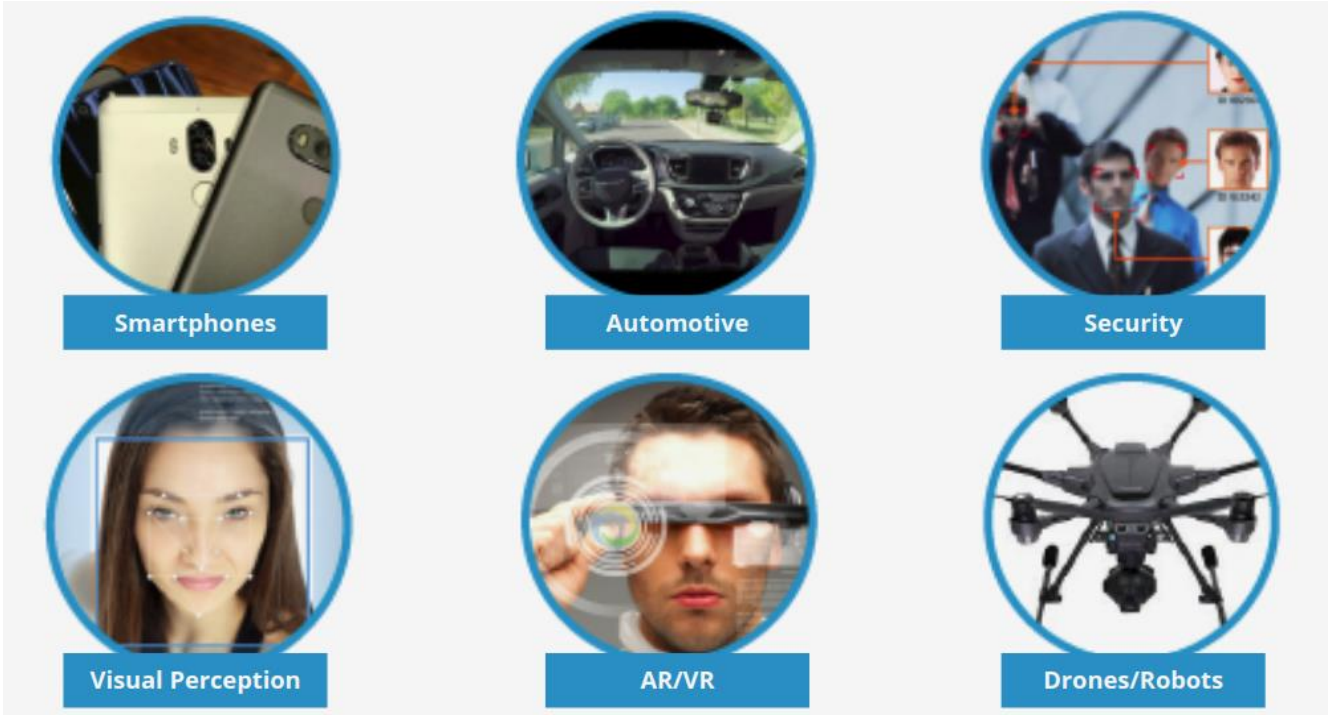
- Leverage use of GPU
  - Rectification
  - Zero Disparity Correction



Flowchart - Advanced processing



Advanced processing



Conclusion

- In order for advanced processing to work correctly, calibration data should not have any errors. The algorithms we work on ensure that the output is validated, and if we do have errors we fix them using on ML techniques, all in real time.

Resources

- Camera Calibration with OpenCV
- Research @ Lucid