COLLABORA

Introducing Monado’s Hand Tracking

Moses Turner

Open First

COLLABORA

Moses Turner

- Freedesktop
- GitHub
- Blog
- @mosesturnervevo
- moses@collabora.com

Open First

COLLABORA

Talk to me about

- Transhumanism
- Radical life extension
- Life as a virtual being
- XR, VR & AR
- Machine learning, AI and computer vision!
- Math!
- FOSS!
- Joining Collabora!

COLLABORA

- **Why do we care so much about optical hand tracking?**
- What have we done so far?
- Current status
- What's next?
- Wrapping up

Why optical hand tracking?

- Can’t use controllers at the same time as a keyboard, which makes controller-based apps largely unusable for productivity
- Markerless - you don’t need anything besides the headset on your head and/or an external camera in the room
- Very accessible
  - Your hands are always there, and optical hand-tracking can be an “always-on” input that applications can rely on easily
  - If you can’t use hand tracking, it’s a lot easier to emulate hand gestures using controllers than it is to emulate controllers using hand gestures
  - Markerless optical hand tracking is generally already robust to missing fingers/etc
- Largely proven out, see Mediapipe, Oculus Quest, Ultraleap, HTC
- Feels like *your hands!* Can be a lot more transformative than controllers.
- Note: Controllers *are* great for games (Beat Saber, Pavlov etc.) and art (Tilt brush etc.) But we want to be able to do boring stuff in VR too!

Why are we working on optical hand tracking?

- No good FOSS solution, besides *maybe* MediaPipe.
  - I have some controversial negative opinions about MP's feature engineering
  - Dataset is unpublished; you have to rely on them to retrain and publish weights
- Some good open-source datasets, but licensing is tricky if you aren't a non-commercial research institution. Can be difficult to get commercial licenses.
- Chicken-and-egg problems like:
  - You could do sensor fusion with myoelectric wristbands and optical hand tracking, but to do so you'd have to convince somebody that this'll work so they will share the sensor-fusion component of their software.
- It sucks to be doing commercial R&D in this space. Huge companies can sometimes do this profitably by vertically integrating, but it's risky.
- We want to help make it possible to try new things on a budget, using our software as a base to start from.
- Push the SoTA forward while also lowering the barrier to entry.

Explicit goals at 30,000 feet

- Publish a completely FOSS optical hand-tracking software that competes with or exceeds SoTA, runs in realtime, and has a C api. ✅
- Publish all real datasets we've collected, with open licenses that allow anybody to use and contribute to them. 🔜
- Publish all of the data and code used to generate our synthetic dataset, with the same set of licenses and contribution pathways. 🔜
- Keep improving what we've got, and make it easy for other people (individuals, research institutions, companies; literally anybody) to help! ✅
- Contract work for people who want a specific R&D area explored, or for people who want tight integration with a product they're shipping. 🔜

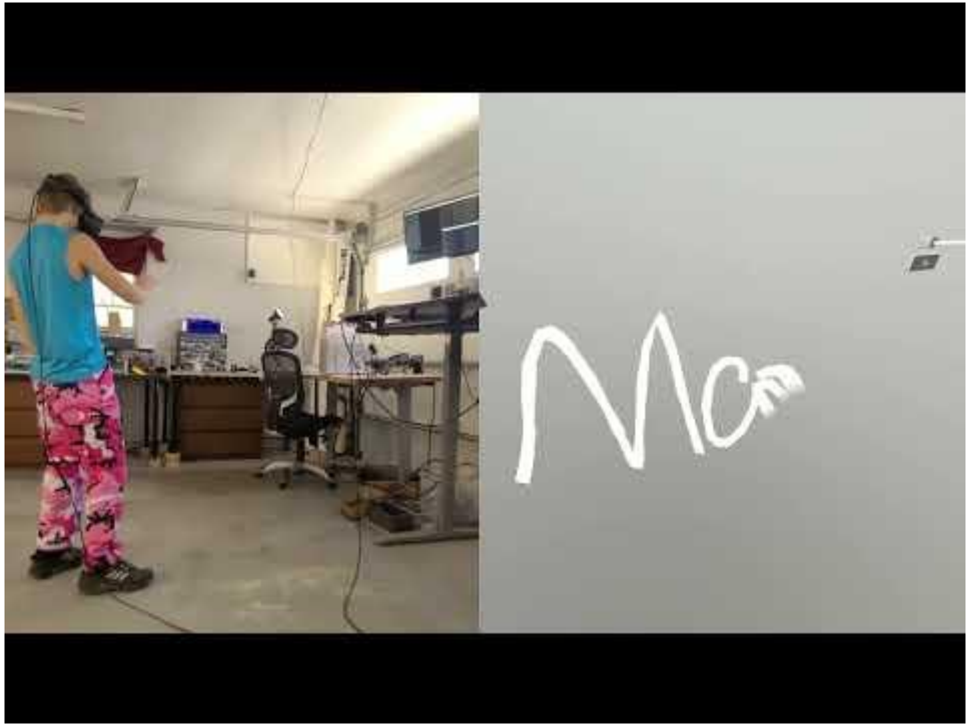
COLLABORA

- Why do we care so much about optical hand tracking?
- **What have we done so far?**
- Current status
- What’s next?
- Wrapping up

2021: Mediapipe derivative

- Before I joined, Jakob Bornecrantz, Marcus Edel and Aaron Boxer had created some fine-tuned Mediapipe models.
- I joined in May 2021, was really excited to work on hand tracking, and got myself the (daunting, but exciting!) task of integrating those models into a real-time tracking pipeline in Monado!
- Mostly finished in October. Didn't do that much more work on this due to unexpectedly high university workload.

2021: Mediapipe derivative (Demo)



2021: Mediapipe derivative (Cons)

- It took me tens of takes to get a video where the tracking didn’t fail horribly. It was bad, really bad, and everybody should know that it was bad. It was my first try at this and I’m really proud that it worked at all, but this was a dead-end and we needed to rearchitect.
- Didn’t have a nonlinear optimizer so
  - Relied on triangulation and required the hands to be visible in both views
  - Couldn’t keep the hand constrained within the set of possible hand poses
  - Couldn’t keep the hand size/joint lengths constant
  - Had to write some really hacky code to estimate joint orientations
  - If one of the NNs estimated an incorrect hand pose, the depth would be super wrong
- Didn’t use NNs to classify left/right hands, and as such got this wrong all the time
- Super jittery, had to add a ton of euro filtering, which added a ton of latency
- (This could have been fixed but) Ran the hand detection model every single frame, which was really slow
- Generally not robust at all.

2021: Mediapipe derivative (Pros)

- ONNX is an awesome file format, and is a solid solution to a real-world problem.
- ONNX Runtime (a ML inference runtime that uses ONNX as its native file format) is also awesome, and is fast enough on CPU
- In general, doing everything on CPU is fast enough and means that you can leave the GPU alone and get more consistent frametimes

December 2021-now: Mercury hand tracking

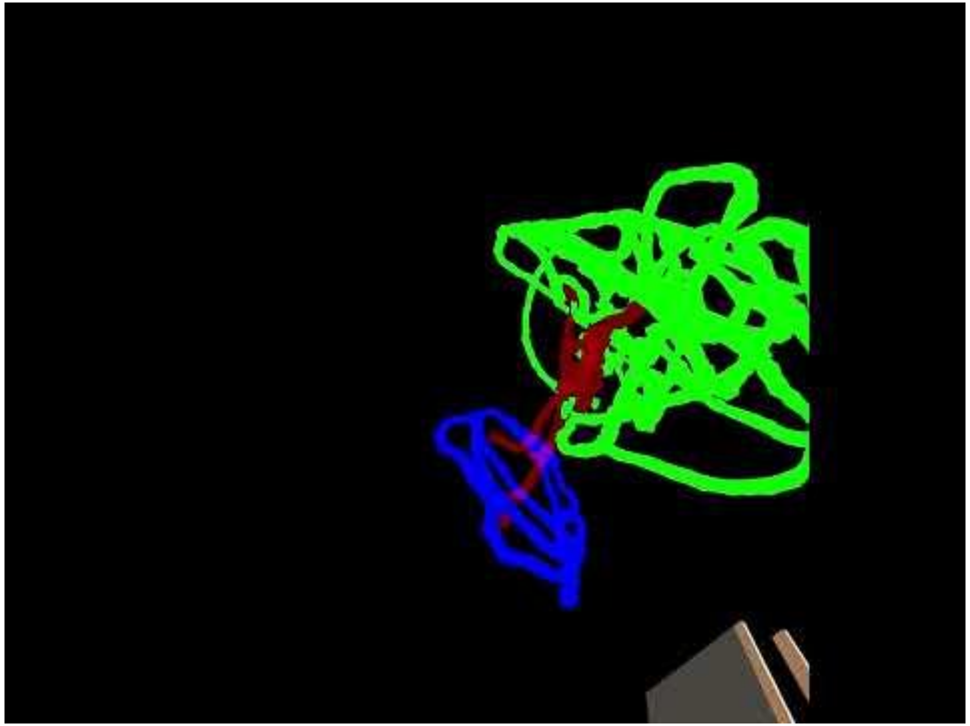
- Mercury doesn’t mean anything - it’s a codename that popped into my head and nothing more. We may rebrand at some point.
- Sees in grayscale
- Better hand detector that jointly classifies left+right, using global image info (ie. we’re on somebody’s head; we can use the forearm direction to guess a lot about which hand this must be)
- Better feature engineering for keypoint estimator: don't rotate it so the fingers are on top, but *do* flip the right-hand images so it only ever sees left hands during inference
- Has some way of constraining the hand so that
  - The hand size and joint lengths stay constant over time, once we've correctly estimated the overall shape of the hand
  - Overall jitter is lower, without adding latency like One Euro filters do
  - Tracked hand doesn't assume impossible poses
  - Stretch goal: Use gaussian priors everywhere so that we always predict the statistically most likely pose given current information and previous uncertainty values
  - I started without knowing a lot of this. I started by using a totally crazy optimizer that used Inverse Kinematics, still depended on stereo depth estimation and was not good at lowering jitter.
  - Levenberg-Marquardt is the clear winner out of established, non-ML solutions. I'll get deeper into this later.
- Uses pose-prediction to predict hand ROIs so that we don’t waste compute running the detection model every frame

Tick...

(March 2022)
Uses IK optimization: note the unusably high jitter.

It's crazy that I got IK to work at all here, but this is not the right way to do it!

Tick...



(June 2022)

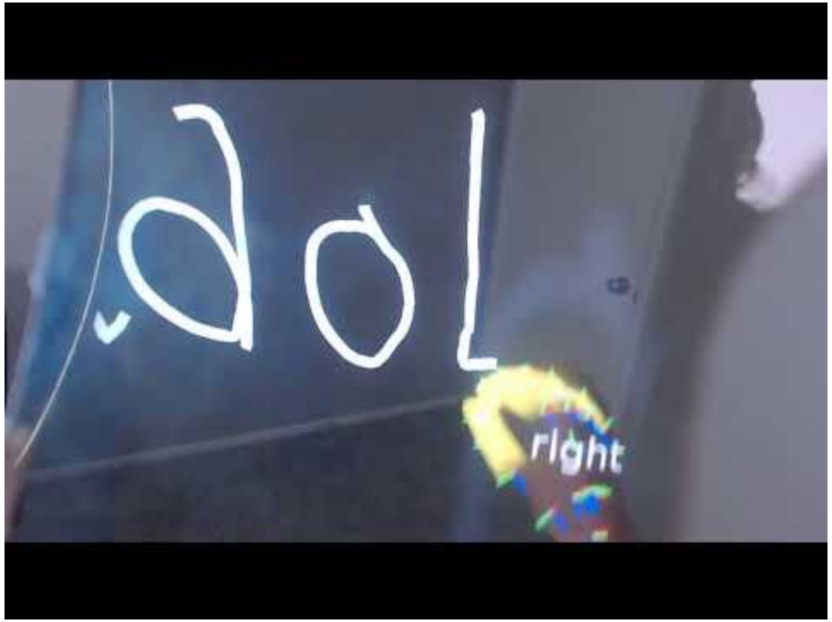
Uses Ceres's autodiff + Levenberg-Marquardt nonlinear optimization.

We can control jitter<->latency now!

The keypoint estimator model is still too jittery, but it's getting there!

Boom!

(Prerecorded video just in case)

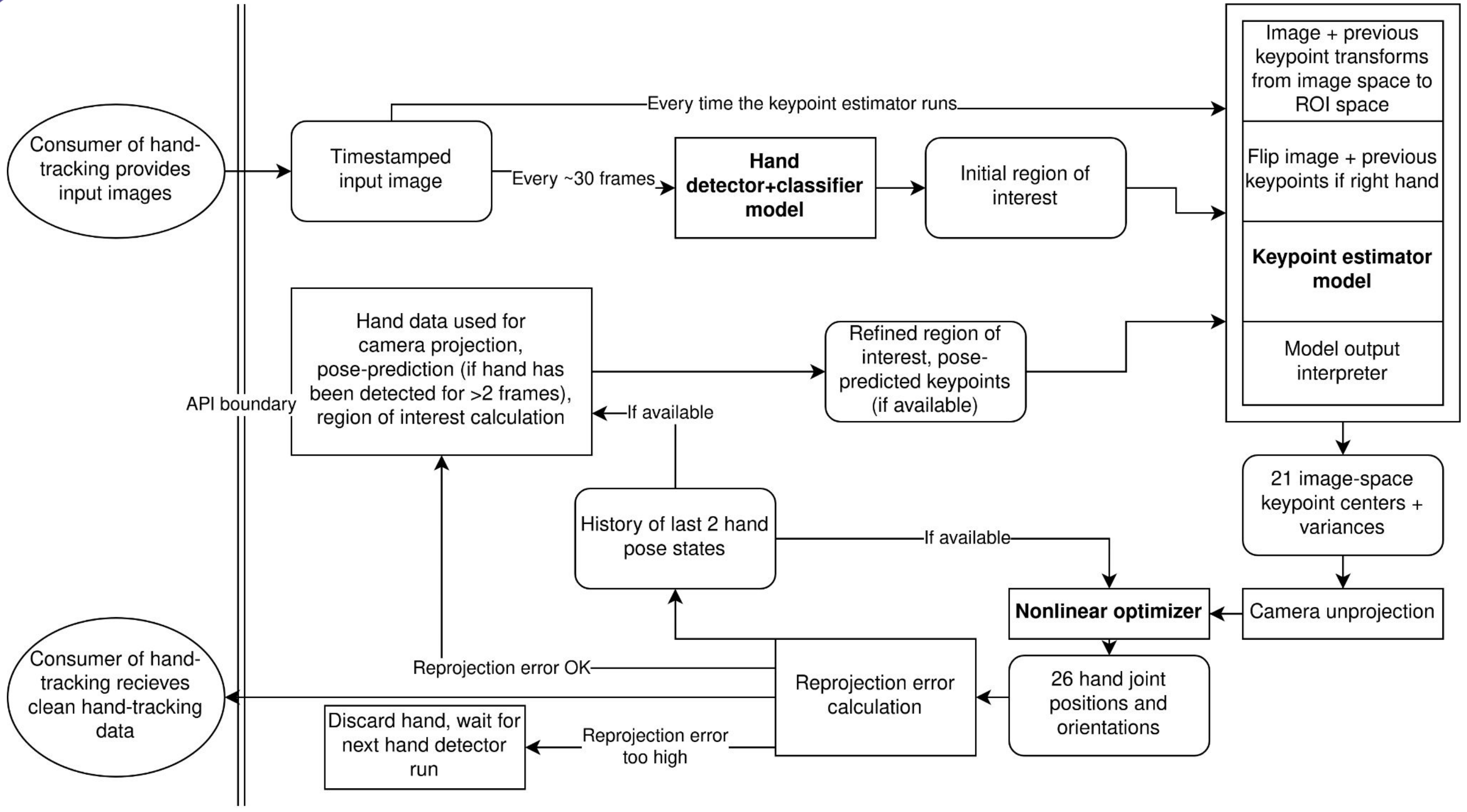


(About a week ago)
Same optimizer; better keypoint estimator trained with artificial data! Jitter is mostly gone now.

COLLABORA

- Why do we care so much about optical hand tracking?
- What have we done so far?
- **Current status**
- What's next?
- Wrapping up

COLLABORA

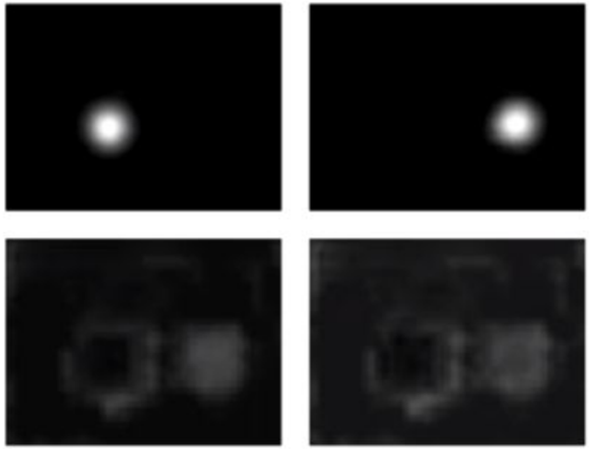


Consumer of hand-tracking provides input images
Timestamped input image
Every time the keypoint estimator runs
Every ~30 frames
Hand detector+classifier model
Initial region of interest
Image + previous keypoint transforms from image space to ROI space
Flip image + previous keypoints if right hand
Keypoint estimator model
Model output interpreter
Hand data used for camera projection, pose-prediction (if hand has been detected for >2 frames), region of interest calculation
API boundary
If available
Refined region of interest, pose-predicted keypoints (if available)
21 image-space keypoint centers + variances
History of last 2 hand pose states
If available
Nonlinear optimizer
Camera unprojection
Reprojection error OK
Consumer of hand-tracking recieves clean hand-tracking data
Reprojection error calculation
26 hand joint positions and orientations
Discard hand, wait for next hand detector run
Reprojection error too high

Open First

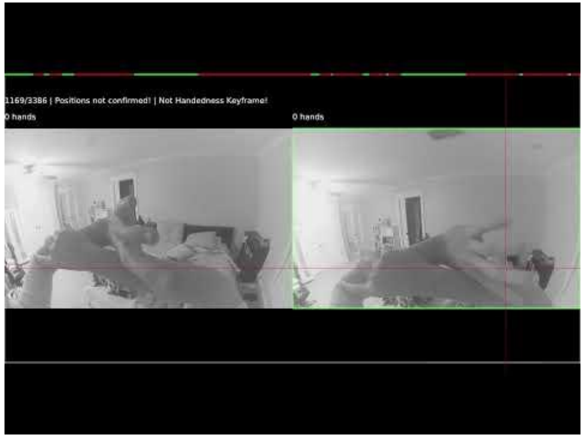
Mercury hand tracking: New hand detector

- Trained a custom hand detector + classifier
- Sees in grayscale, 320x240. (Something to revisit: we could train RGB variants and different aspect ratios)
- SimDR model architecture (Something to revisit: I'm pretty sure we can easily speed this up at least 5x)
- Outputs heatmaps for hand centers and radii
- Datasets:
  - EPIC-KITCHENS
  - EgoHands
  - TV-Hand and COCO-Hand
  - An in-house dataset collected with a North Star and annotated semi-automatically by yours truly
- Data pipeline uses a bunch of PyTorch Datasets and a ConcatDataset

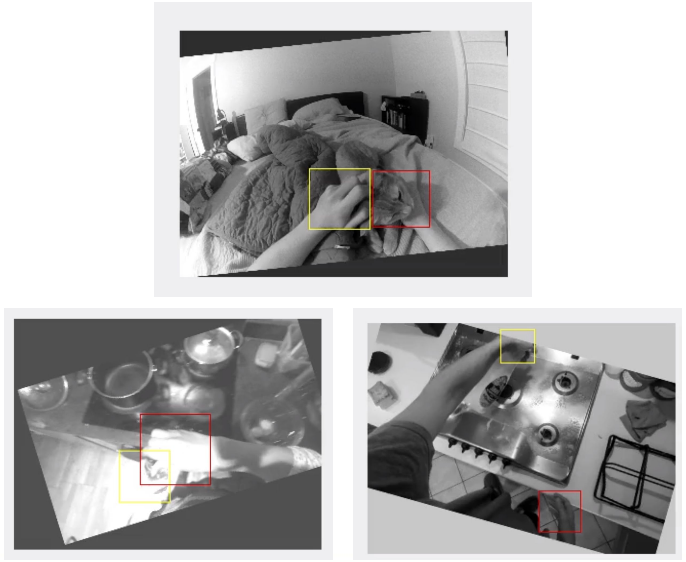


Training the hand detector

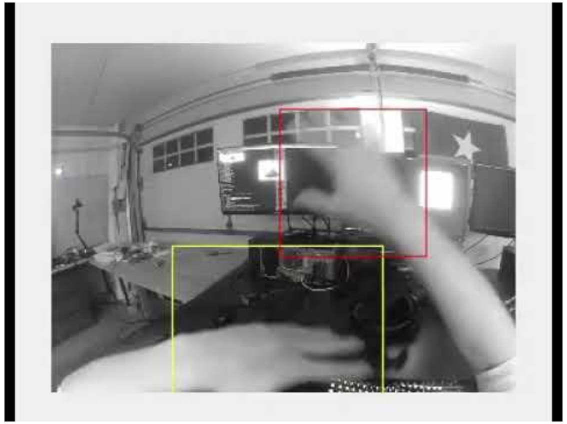
Annotation


```
1169/3386 | Positions not confirmed! | Not Handedness Keyframe!
0 hands                                0 hands
```

Training

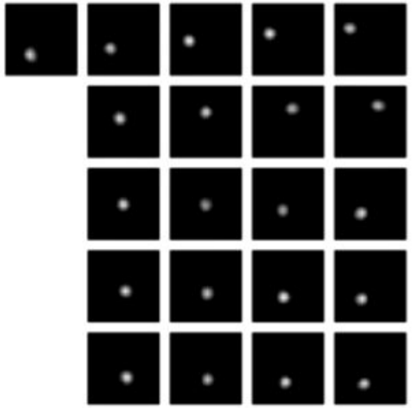


Inference



Mercury hand tracking: New keypoint estimator

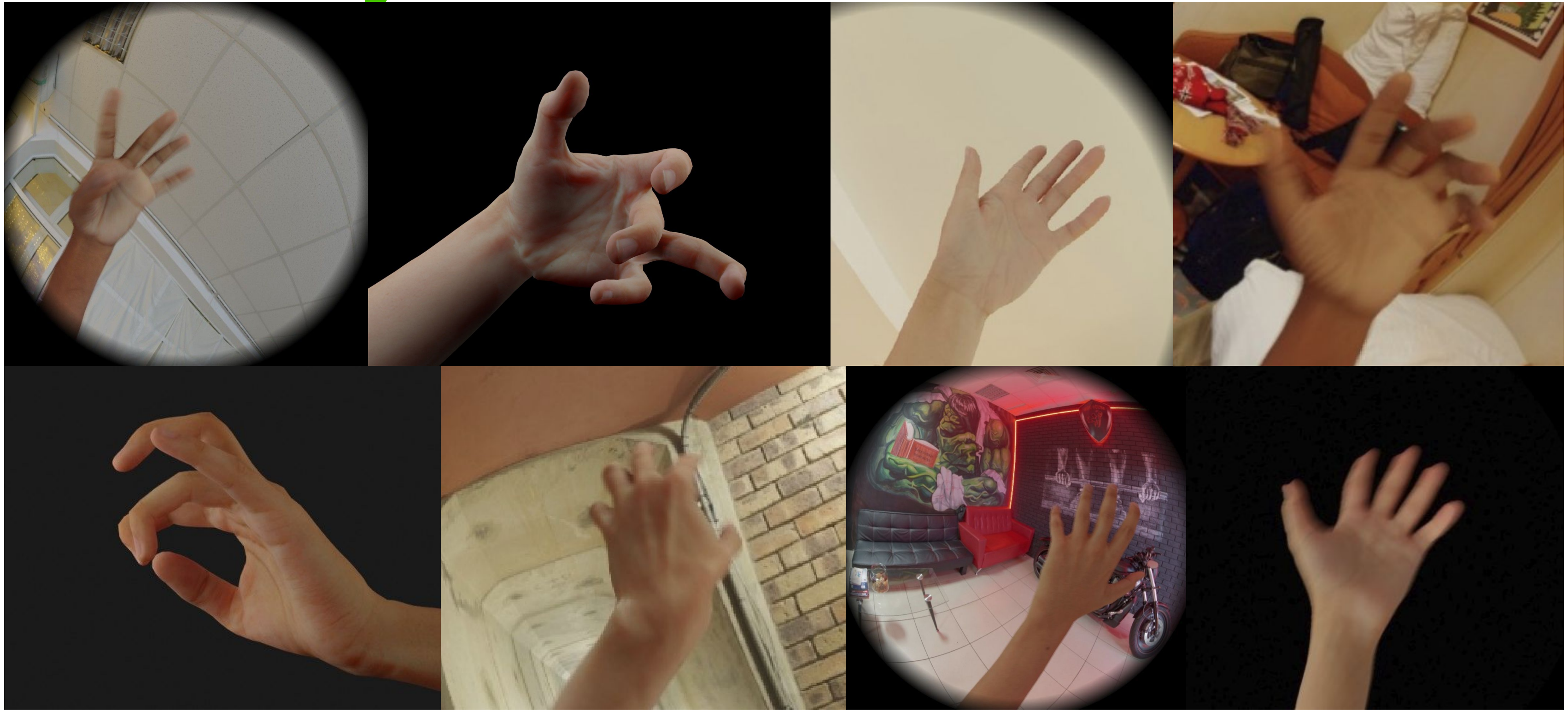
- Trained a custom hand keypoint estimator
- Sees in grayscale, 128x128. (Something to revisit: we could train RGB variants, and experiment with smaller/bigger input sizes)
- Outputs are
  - 21 2D heatmaps predicting the most likely keypoint position in pixel coords
  - 21 1D heatmaps predicting the most likely keypoint depth relative to the middle-proximal joint
- Same model architecture as Quest hand tracking
- Datasets:
  - Small greenscreen dataset I collected and annotated using Mediapipe
  - Dataset that my friend Tom Watts collected and annotated using OpenPose
  - CMU panoptic dataset
  - FreiHand
  - Artificial dataset I generated (talk about this later!)



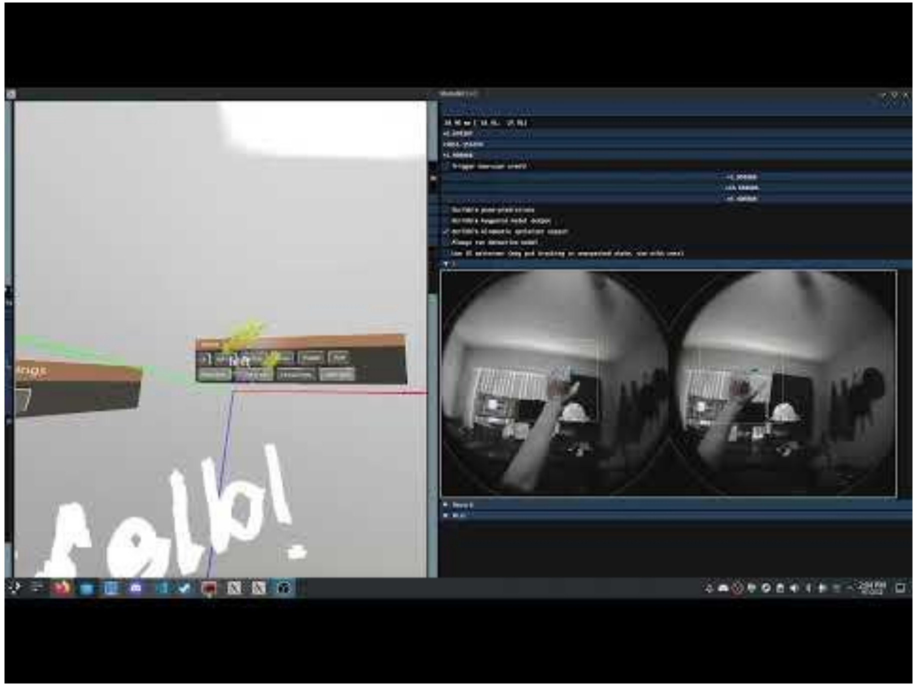
Problems with existing public hand landmark datasets

- Many don't have strictly sequential annotations: instead, you get a bunch of unrelated individual images. (Notably, InterHand2.6M does this exactly right)
- You need to calibrate your cameras and store joint locations in 3D relative to the hand! Pretty much all datasets just annotate in 2D image-space, making them unhelpful for more complicated model architectures. (Notably, InterHand2.6M does this exactly right)
- Basically all real-world datasets have some percentage of incorrect annotations that are hard to filter out. (InterHand was the worst here, so bad that I decided to give up on using it. Last I checked was April 2022; it may have been fixed since then.)
- Privacy and liability issues. You might be screwed if even one image has unwanted PII.
- If you collect a new real-world dataset, you have to worry about the above and:
    - Am I going to accidentally lose data?
    - How long will it take my annotators to annotate everything? How often are they making mistakes?
    - Am I using the right type of cameras for my application?
    - Panoptic studios are expensive

Solution: Synthetic data!



Solution: Synthetic data!



What I showed you before was trained on a lot of synthetic data and a little bit of real data, but we can also track hands using _only_ synthetic data!
The quality isn't quite as good yet, but it's totally a fixable data problem: we need to add more diverse poses, clothing, rings, etc. to the dataset generator.

Solution: Synthetic data!

- ML mesh generation isn't quite good enough yet, so we're using "classical" techniques.
- Started with some very high-quality hand scans from an asset store
  - Having a small licensing issue here - we will publish existing scans or create new ones hopefully by Q1 2023
- Rigged them in Blender
- Wrote a C++ pipeline that
  - Generates pose data by permuting a wrist-pose dataset I collected using Lighthouse tracking with a finger-pose dataset I collected with our hand tracking under good lighting conditions
  - Runs blender with a custom Python script controlled by environment variables. Blender sets up lighting, HDR background, animates the hand and renders a bunch of cubemaps
  - Takes the cubemaps and a random camera calibration, and distorts them into final images that look like they were captured on a HMD's onboard camera
  - Writes out the finger poses as 3D coordinates relative to the left camera
  - Is published here (code quality has room for improvement)

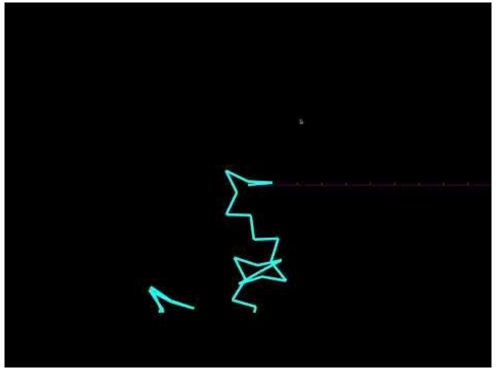
Training the keypoint estimator

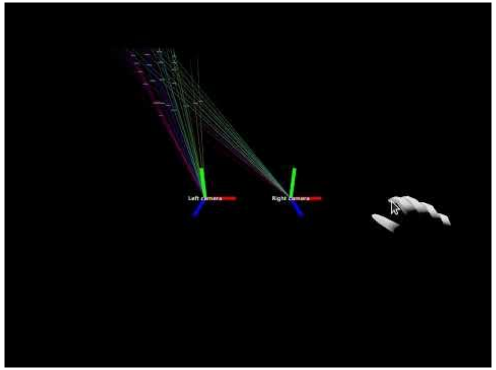
- Basically same idea as the hand detector. Many PyTorch Datasets, rolled into a ConcatDataset
- NoneChucks is really nice for when you've found some mistakes in the dataset you spent 30 hours generating
- Fancy loss function that only looks at model's depth prediction if we have ground truth depth
- Training code is published here (code quality has room for improvement)

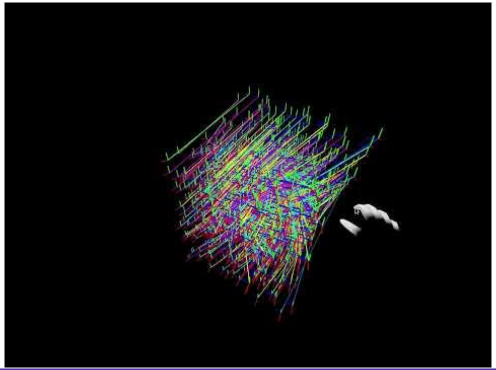
Mercury hand tracking: Realtime nonlinear optimization!

- We start with predictions of the keypoint locations in camera space, and want to end up with 6dof poses for each joint. You can very painfully and inaccurately do this by hand using camera triangulation and Inverse Kinematics, but it's way, way easier to…
- Write a function that takes a low-dimensional vector encoding hand pose (hand size, 6dof wrist pose, joint curl values), evaluates this to 6dof joint poses, and outputs a metric for "How close are these poses to the model predictions?"
  - Then, ✨ask your computer✨ to evaluate this function many times, working towards a good solution. This is easily accomplished using gradient descent - you could easily implement this in PyTorch, for example. Levenberg-Marquardt is typically faster, and it's really just a mashup of gradient descent and Gauss-Newton.
- The exact same technique is used to solve for
  - HMD/controller pose for lighthouse tracking (Vive/Index)
  - HMD/Controller pose for constellation tracking (Oculus/WMR)
  - Head pose for SLAM (Basically everyone)
- Very simple to add extra terms to system: temporal consistency, extra sensors, myoelectric tracking, wrist-mounted IMU, etc.
- Works for underconstrained, exactly-constrained and over-constrained systems.
- More flexible than, and often more efficient than Kalman filters. Typically does the same thing given Gaussian priors.

Nonlinear optimization is awesome!







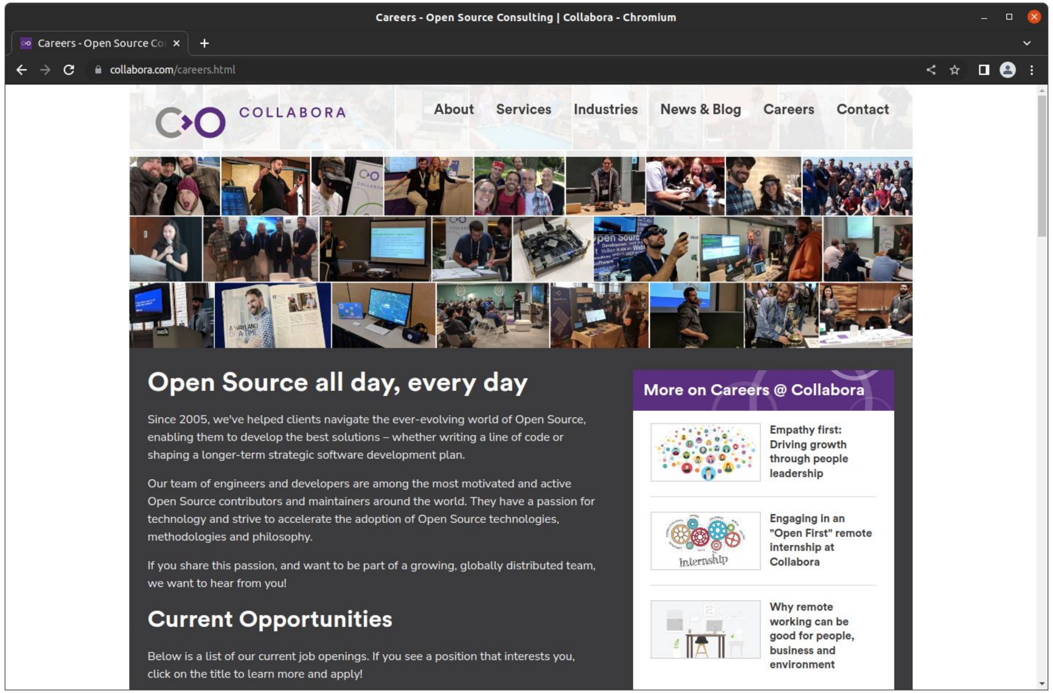
COLLABORA

- Why do we care so much about optical hand tracking?
- What have we done so far?
- Current status
- **What's next?**
- Wrapping up

What’s next?

- Improving our artificial data to the point where real data has no marginal value
-
    - Better pose remapping from mocap datasets to the specific model
    - Soft-body flesh/tendons/skin simulation? (Do I know anybody who knows anything here? Code-first approach pls)
    - More mocap data
    - Random walks over plausible hand pose space
    - Intertwining fingers
    - Try simulating depth cameras and event cameras
- Train very slow but very accurate models to explore the limit of how accurate optical egocentric hand tracking can get
    - Diffusion in pose-space?
- Try a bunch of ideas for improving accuracy in our real-time models
    - Train a keypoint estimator that sees in stereo?
    - “Refinement” model that fixes up the region of interest, so that we can track quick movements with less jitter?
    - Lots of stuff to try here. “idk try it i guess” is by far the most effective way to do novel computer vision research.
    - Create better techniques to assess accuracy/performance
- Elbow tracking!
- Publish every last bit of our dataset, dataset generation and tracking code.

We are hiring!


Careers - Open Source Consulting | Collabora - Chromium
collabora.com/careers.html
COLLABORA
About
Services
Industries
News & Blog
Careers
Contact

Open Source all day, every day

Since 2005, we've helped clients navigate the ever-evolving world of Open Source, enabling them to develop the best solutions – whether writing a line of code or shaping a longer-term strategic software development plan.

Our team of engineers and developers are among the most motivated and active Open Source contributors and maintainers around the world. They have a passion for technology and strive to accelerate the adoption of Open Source technologies, methodologies and philosophy.

If you share this passion, and want to be part of a growing, globally distributed team, we want to hear from you!

Current Opportunities

Below is a list of our current job openings. If you see a position that interests you, click on the title to learn more and apply!

More on Careers @ Collabora

Empathy first: Driving growth through people leadership

Engaging in an "Open First" remote internship at Collabora

Why remote working can be good for people, business and environment

Questions?

Thank you!