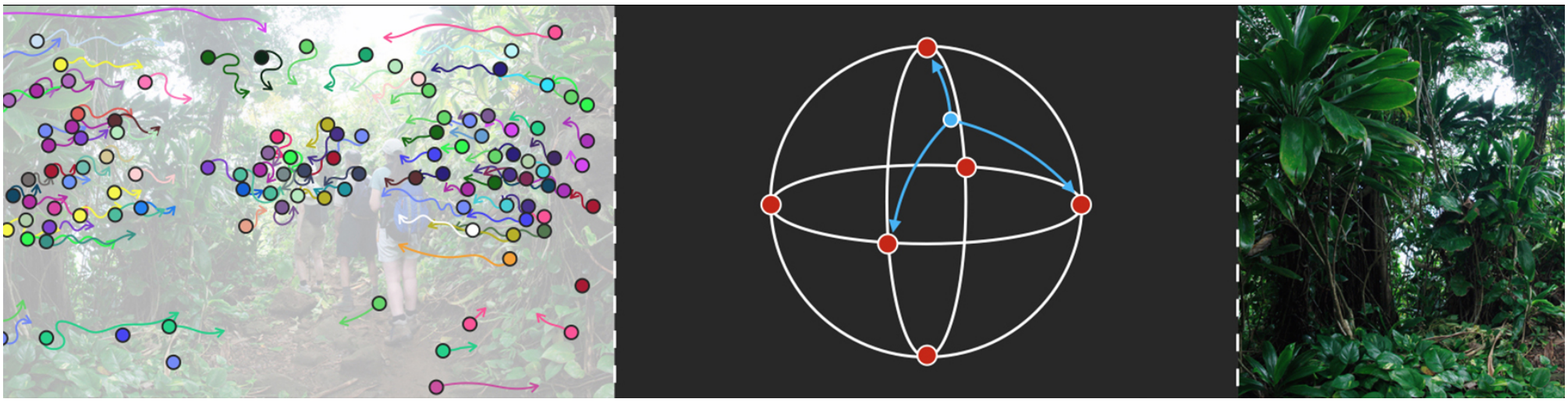
POSTED ON AUG 31, 2016 TO AI RESEARCH, VIDEO ENGINEERING

360 video stabilization: A new algorithm for smoother 360 video viewing



By Johannes Kopf

360 video is rapidly becoming more widespread. There are dozens of devices for capturing 360 video, from professional rigs to consumer har cameras, all with different specs and quality outputs. As these types of cameras become more prevalent, the range and volume of 360 conter expanding, with people shooting 360 video in a variety of situations and environments. It's not always easy to keep the camera steady and avo particularly when filming motion (like a mountain bike ride or a walking tour) with a handheld camera. Until now, most video stabilization tech been designed for videos with a narrow field-of-view — like traditional videos filmed using a smartphone — and it uses techniques that don't to 360 video.

As more people are shooting more immersive videos in real-life scenarios and sharing these moments with their friends, we decided to develo stabilization technology custom-designed for 360 video. We're beginning to test it now and will eventually make it available to everyone on Fa the Oculus platform. This approach uses a new "deformed-rotation" motion model that we designed specifically for 360 video, and a new hyb technique for optimizing the model parameters in order to make shaky 360 videos incredibly smooth. It improves efficiency, with a 10 percen percent reduction in bit rate for the same video quality. And it's fast — it can stabilize 360 video in less than 22 milliseconds per frame on a sta machine. In fact, videos can be stabilized using this technology in less time than it takes to play the video at normal speed.

Another advantage of our new stabilization method is that you can speed up a stabilized 360 video more easily, turning a lengthy video (e.g., a ride) into a fast-paced and fun action experience. To enable this, we've developed a 360-hyperlapse algorithm as an extension of the main stab algorithm. It changes the timing of the video frame timestamps to balance out the camera velocity over time. We're testing this now and hope available to people in future versions.

Let's dig into the technology behind the new algorithm and how we built it.

Creating a novel algorithm architecture: Hybrid 3D-2D + deformed-rotation mo

Most existing video stabilization algorithms use the same architecture: They track motion in the video, then fit a motion model, smooth the m produce the stabilized output frames. The main difference between algorithms is how they model the motion in the video.

Most algorithms are designed specifically for narrow field-of-view video and use simple parametric 2D-motion models, such as full-frame pers (homography) warps. While these methods are fast and robust, the motion models are too simple to handle complex motion, such as parallax different shake on foreground and background. More advanced methods, mostly described in academic publications, use more flexible motio such as non-parametric mesh warps. While these can handle more complex motion, it is challenging to constrain the extra flexibility to avoid p visible geometric deformations.

Another category of algorithms operate in 3D. They reconstruct a geometric model of the camera trajectory and the scene, and reason abou stabilized video in terms of 3D quantities. These algorithms have higher smoothing ability, since they use a more accurate model, but in practi reconstruction methods are often slow and less robust than 2D methods.

We developed a new hybrid 3D-2D stabilization architecture that combines the advantages of both types of algorithms. We use 3D analysis on key frames spaced a few seconds apart in time, and instead of performing a full reconstruction, we only estimate relative rotation, which is an problem that can be robustly solved. Using a 3D algorithm is helpful because it can distinguish whether the observed motion comes from rota translational camera motion.

For the inner frames (the remaining frames between the key frames), we turn to 2D optimization and directly maximize the smoothness of m video. We developed a new "deformed-rotation" model, which is similar to a global rotation but allows slight local deformations. We optimize parameters of this model so it can adapt to handle and undo some degree of translational shake (such as bobbing up and down while walking camera), rolling shutter artifacts, lens deformations, and stitching artifacts. In this way, we minimize the effects of these phenomena.

The hybrid 3D-2D architecture has several advantages over pure 2D or 3D methods:

- **Accuracy:** It's more accurate than pure 2D methods because we use a more powerful 3D analysis to stabilize the key frames.
- **Robustness:** It's more robust than pure 3D methods because we don't perform full reconstruction — we use 3D analysis to estimate re rotations, which is an easier problem. We also apply the 3D analysis only sparsely to the key frames, which avoids error accumulation.
- **Regularization:** The fixed key frames provide a regularizing backbone for the 2D optimization of the inner frames. This constrains the d rotation motion model and prevents wobble artifacts.
- **Speed:** The mixed architecture performs faster than performing purely 3D analysis or 2D optimization. Our algorithm stabilizes a video it would take to play the video at normal speed.

Step by step: How it works

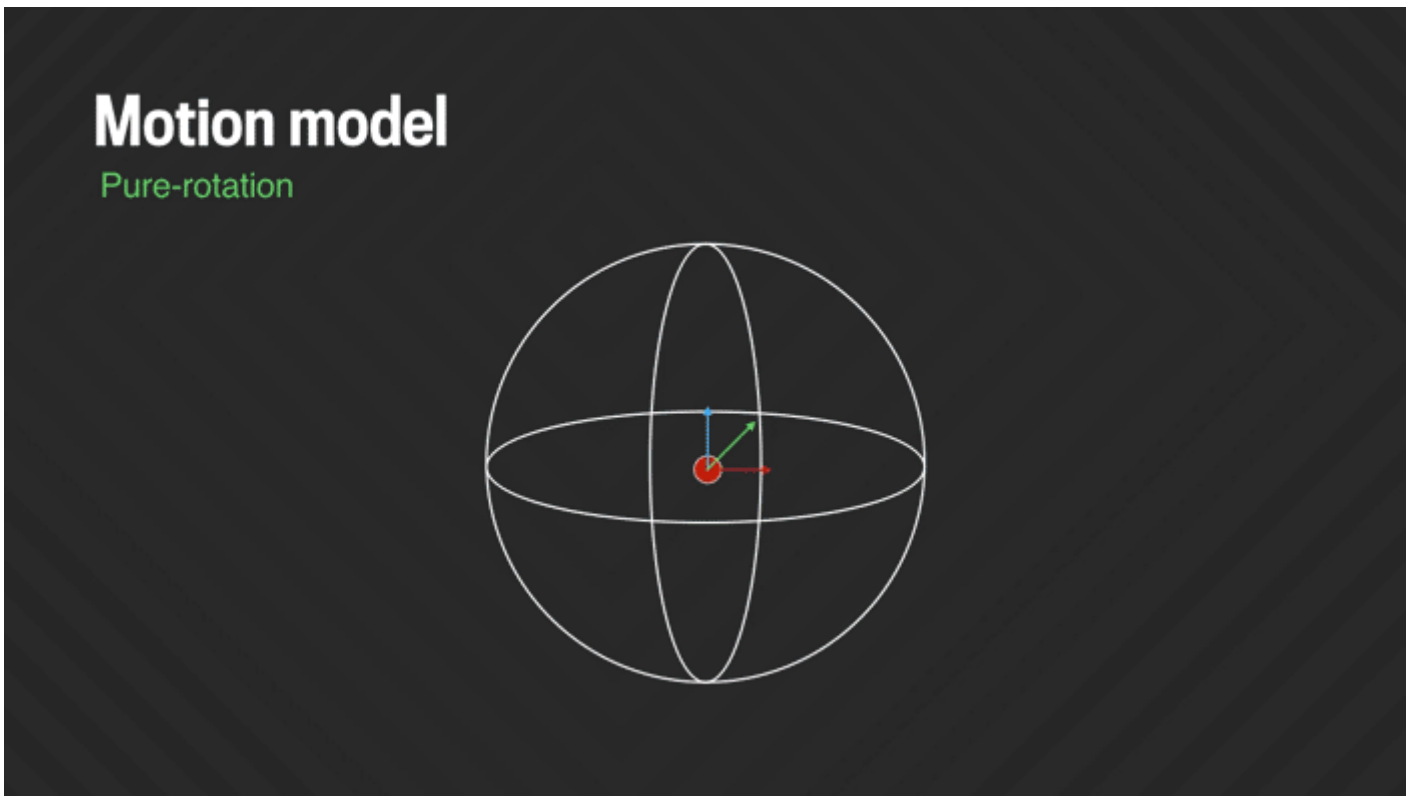
Like most existing stabilization algorithms, we start by tracking the motion of feature points in the video. (These are visually distinct patches, corners.) Since our input videos use equirectangular projection (well known from world maps), which is highly distorted near the poles, we co frames into a less distorted cube map representation for tracking. We always resample the cube map faces to 256×256 resolution. Since track well on grayscale images, we perform it directly on the video luma plane to avoid the slow YUV to RGB conversion.

We implemented a KLT tracker on top of the OpenCV library, which has routines for extracting feature points, and a highly efficient multi-sca Kanade implementation for tracking them. Our tracker has a notion of "key frames," which play a very important role, as we are estimating th relative rotations and they form a regularizing backbone for the subsequent 2D optimization. We use a heuristic to spread the key frames out packed more densely in parts of the video with more change.

We then estimate the relative rotation between successive key frames using a standard five-point computer vision algorithm that estimates th rotation and translation between two cameras given five-point matches (our algorithm does not use the recovered translations). We use the library's implementation of Nister's five-point algorithm in a RANSAC procedure for more robustness. The advantage of using a 3D-aware algo

estimate the rotation is that it has the ability to distinguish true rotation from translational motion. After having estimated all key frame rotati apply the inverse transformation to rotationally align all of them to the video's first frame.

Now that the key frames are stabilized, we lock their rotations for the rest of the procedure and shift our attention to the rotations of the in-l frames. As mentioned above, we stabilize them using 2D optimization instead of 3D analysis. The goal of this optimization is to find the best rc the non-key frames, such that the smoothness of the point tracks is maximized. During the optimization, we keep the previously computed ke rotations fixed; they provide anchors that regularize the optimization and improve convergence. We set the optimization up as a non-linear rc squares problem and use the Ceres library to solve it.



Solving for pure rotations removes most of the camera shake, but often some amount of residual jitter remains, due to a combination of sma translational motion (e.g., bobbing the camera up and down while walking), rolling shutter wobble, and suboptimal lens calibration and stitchi We address these problems by adding some flexibility to the motion model so it can adapt and undo slight image deformations. While it woul possible to model some of these effects directly for a specific camera model (e.g., the rolling shutter characteristics and lens deformation), w interested in a generic deformation model that does not require specific knowledge about the camera that was used. In our "deformed-rotati we replace the single global rotation with six local rotations spread across the sphere and interpolated across the rest of the sphere. This allo local low-frequency deviations from pure rotation to the data. It is critically important, however, that the model does not become too flexible is constrained properly, so it will not overfit the data and introduce wobble artifacts instead of removing them. We regularize the problem by key frames still fixed to pure (non-deformed) rotation, which effectively prevents the more flexible model for the inner frames from drifting t This new model is directly plugged into the previously described Ceres solver.

Optimizing speed, performance, and efficiency

As we developed this algorithm, we focused on making stabilization fast in order to make it practical for the user — people don't want to end waits while uploading a video. During the final steps of our process, the final output frames are generated by applying the fitted motion mode pixel. While this could be done trivially on a GPU, we were interested in making our algorithm CPU-only so it could run on standard cloud harc Calculating the warp field coordinates on a CPU turned out to be very slow, but the warp field is very smooth. So we speed up the computatic computing the warp coordinates only for one in every 8×8 pixels, and interpolate the remaining coordinates bilinearly. This looks nearly identi but improves the warping speed dramatically. In this way we designed our algorithm so it runs fast on a single cloud machine without GPU ac The table below summarizes the performance of each algorithm stage, measured for a 1080p input/output resolution and timings reported fc frame.

Our stabilization performs faster than the time it takes to play the video normally: In a video player, each frame is shown for about 30 millisec we need only around 20 ms for stabilization. With some algorithm modifications, we could likely enable (delayed) real-time stabilization for liv video.

In addition to speed, we also optimized for efficiency. While it's not surprising that stabilization can have a beneficial effect on video bit rate consumption, there are also some non-obvious insights for 360 video. Because 360 videos are not cropped, we can trivially recover the origin sequence from the stabilized result, by applying the inverse rotation in the viewer. This does not have any performance implications because viewer already applies rotational view transformations.

We found that the bit rate savings from our stabilization are substantial. In the figure below, we analyze the bit rate consumption when encod video into H.264/MPEG-4 AVC format using the x264 library. We tried two sequences, fast-paced downhill skiing (orange) and a mostly static s (blue). As the figure shows, the bit rate savings are between 10 percent and 20 percent, depending on the encoder settings. To our surprise, t for the less shaky sequence (blue) were far more substantial, which is likely because after stabilization it becomes near-constant and very com while the skiing sequence remains fast-moving even with stabilization.

Hyperlapse algorithm

Since our stabilized 360 videos are so smooth, they provide excellent source material for creating sped-up hyperlapse videos. Creating a 360 l by simply dropping all but every nth frame in a sequence already looks great. However, a common element of hyperlapse videography is a sm balanced camera velocity. The varying camera velocity in the naively sped-up video clearly stands out, for example, when a skier stops repeate breaks while filming a skiing sequence. To simulate a constantly moving hyperlapse camera, we need to balance out the velocity temporally an breaks.

To do this, we use a simple heuristic to estimate the camera velocity at every frame. We use a simple 2D approximation, a robust average of th stabilization motion vector magnitudes. This estimate is very noisy, so we process it further by running through a temporal median followed b pass filter. Using the estimated camera velocity we can change the timestamps of the video frames in the stabilized video to stimulate a const velocity. This is a great way to create interesting action videos, and to summarize long sequences into short and fast-paced pieces.

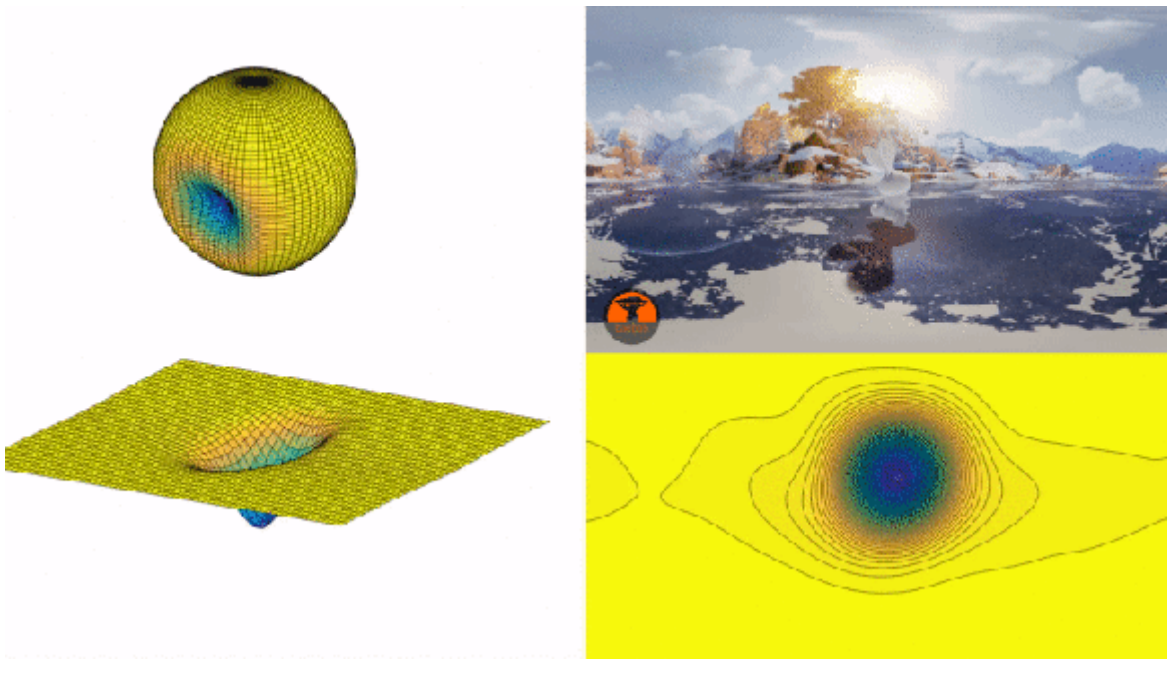
What's next

As we test this new technology, we're listening to feedback on how it works. We hope to make it available to everyone who uploads a 360 vide people can even more easily share high-quality and easy-to-watch immersive moments with their friends. We'll also work to incorporate the n hyperlapse adaptation into the functionality we offer, and looking further ahead, we're interested in exploring the possible applications of our stabilization tech for live 360 videos.

TAGS: OPTIMIZATION PERFORMANCE

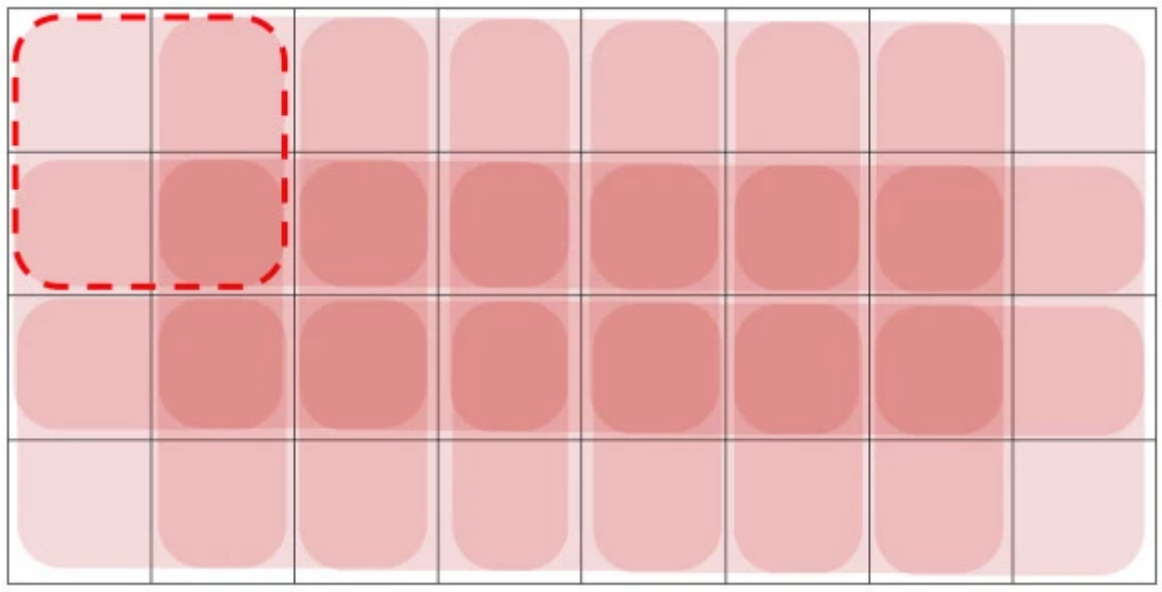
Thích Chia sẻ Hãy là người đầu tiên trong số bạn bè của bạn thích nội dung này.

Related Posts



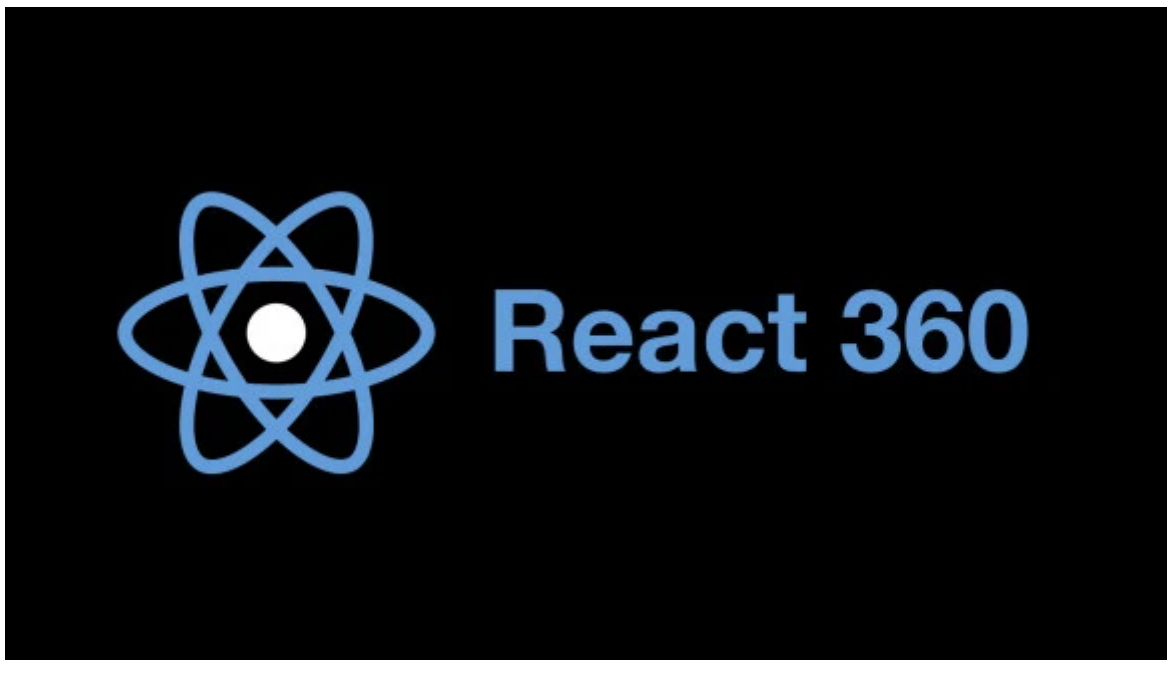
Dec 13, 2017

2017 Year in review: Building immersive experiences



Mar 09, 2018

Quality assessment of 360 video view sessions



May 02, 2018

React 360 replaces React VR for streamlined development focus

Related Positions

Research Scientist (AI)

PITTSBURGH, US

Applied Research Scientist, Multimodal Video Understanding (Deep Fake Detection)

NEW YORK, US

Applied Research Scientist, Core Machine Learning

MENLO PARK, US

Applied Research Scientist, Computer Vision (People AI)

MENLO PARK, US

Research Scientist, Systems and Infrastructure (PhD)

MENLO PARK, US

See All Jobs

Join Our Engineering Community

Available Positions

Research Scientist (AI)
PITTSBURGH, US
Applied Research Scientist, Multimodal Video Understanding (Deep Fake Detection)
NEW YORK, US
Applied Research Scientist, Core Machine Learning
MENLO PARK, US
Applied Research Scientist, Computer Vision (People AI)
MENLO PARK, US
Research Scientist, Systems and Infrastructure (PhD)
MENLO PARK, US

See All Jobs

Stay Connected

Facebook Engineering

Like

@fbOpenSource

Follow

facebook research

Facebook Research

Like

Facebook Developers

Like

Open Source

Facebook believes in building community through open source technology. Explore our latest projects in Artificial Intelligence, Data Infrastructure, Development Tools, Front End, Languages, Platforms, Security, Virtual Reality, and more.

ANDROID

iOS

WEB

BACKEND

HARDWARE

Learn More

RSS

Subscribe

Facebook © 2019

AboutCareersPrivacyCookiesTermsHelp