Proprietary + Confidential

Google

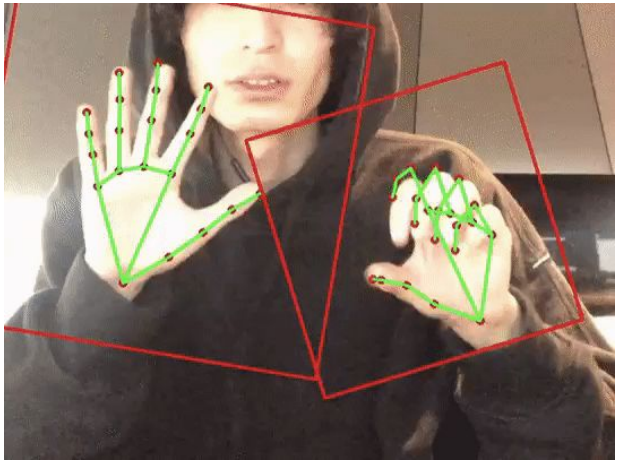
Empowering Live Perception with MediaPipe

Ming Yong (mgyong@google.com)
Google Research
mediapipe.dev

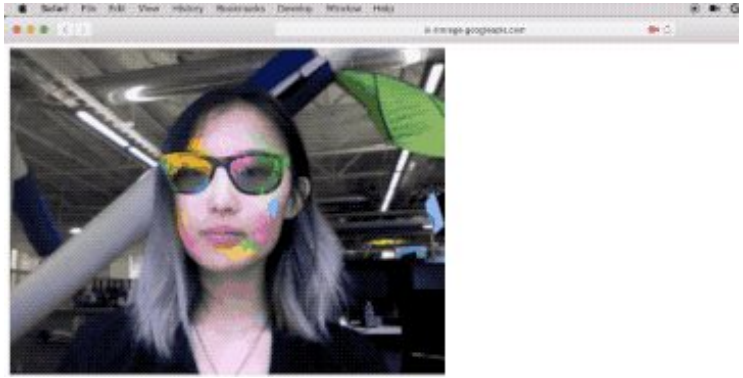
What do these things have in common?

Proprietary + Confidential

Android



iPhone XR



Web

Nest Cam

YouTube

"OK Google"

Rapberry Pi, Coral EdgeTPU

Google

Proprietary + Confidential

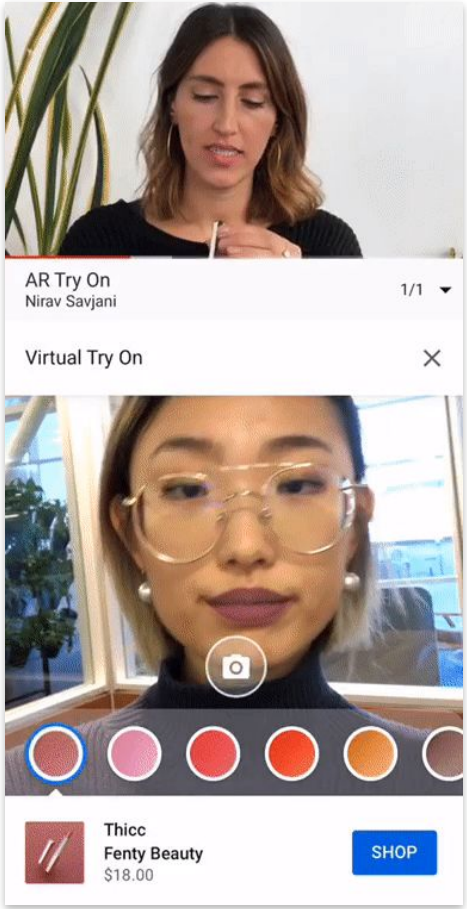
Introduction

MediaPipe is Google's **open source cross-platform** framework for building **perception pipelines**

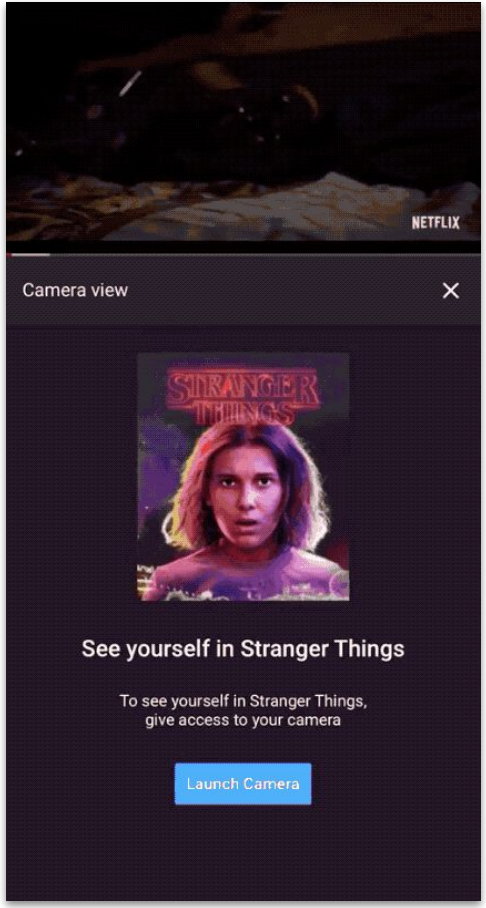
Widely used at Google in **research & products** to process and analyze video, audio and sensor data:

- Dataset preparation pipelines for ML training
- ML inference pipelines

Confidential



AR Lipstick Try-on in YouTube



AR movie trailer in YouTube

Lens Living Surfaces

Lens Translate

Augmented Faces at Google I/O 2019 demo

Google

Public to date

Proprietary + Confidential

On-device & **server** use cases across Google Since 2012

Many more not public

On-device/Edge

YouTube
YouTube Stories

Lens
AR Ads, Playground

Hardware
Nest Cam Perception
Nest Hub Max

Server/Cloud

Video Preview

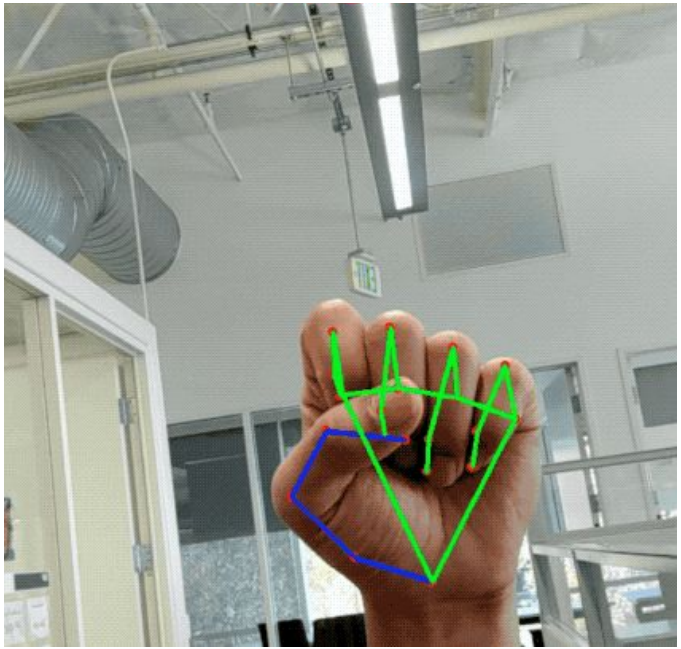
Cloud
Cloud Vision API
Cloud Video Intelligence API
Cloud AI video

Proprietary + Confidential

Live Perception Example

Proprietary + Confidential

Goal: Hand Tracking

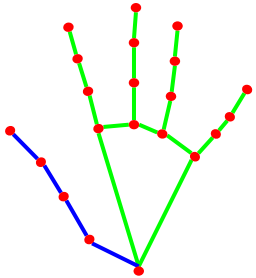


Proprietary + Confidential

ML Model to Localize Hand Landmarks

Image/Video

Model



Landmarks

There's more to it

Google

Proprietary + Confidential

Simple Perception Pipeline

Video input, e.g., from live camera viewfinder

Geometric transformation to resize and rotate etc

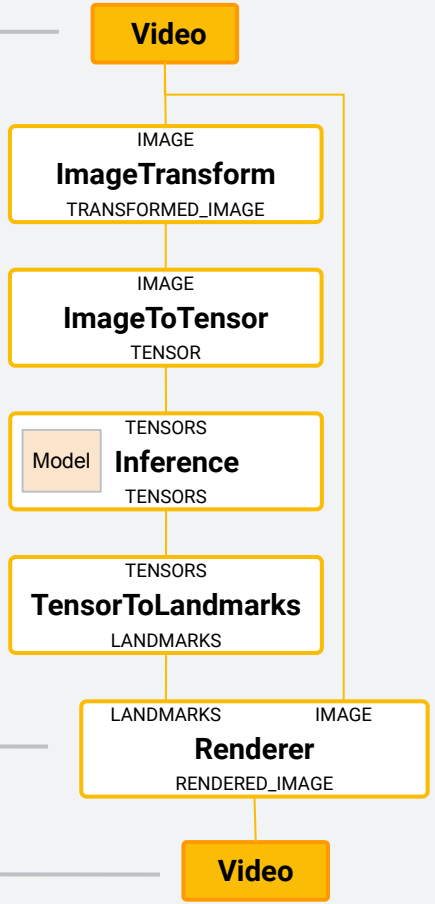
Converts images to tensors

For a given ML model, runs ML inference

Decodes tensors into high-level metadata

Renders landmarks onto associated images

Video output, e.g., display



Google

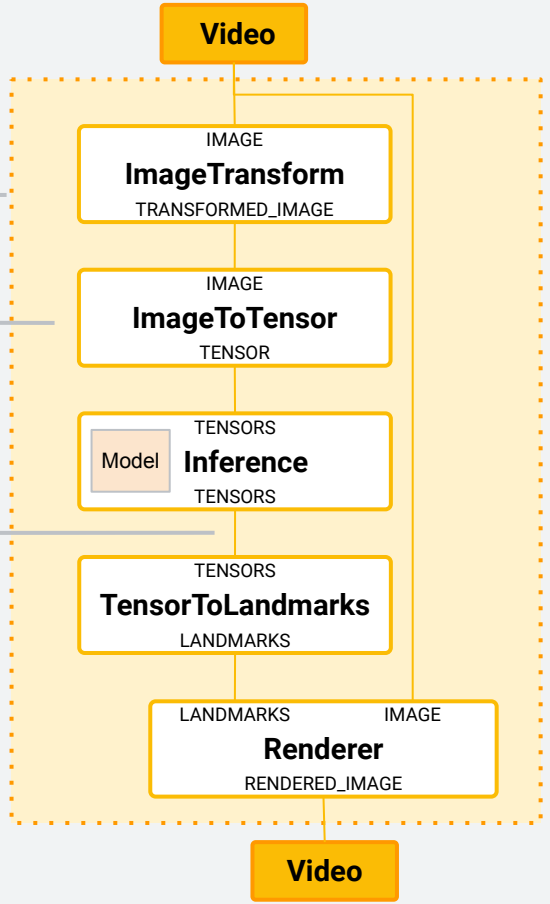
Proprietary + Confidential

Pipeline via MediaPipe

A MediaPipe **Graph** represents a **perception pipeline**

Each node in the pipeline is a MediaPipe **Calculator**

Two nodes can be connected by a **Stream**, which carries a sequence of **Packets** with ascending timestamps

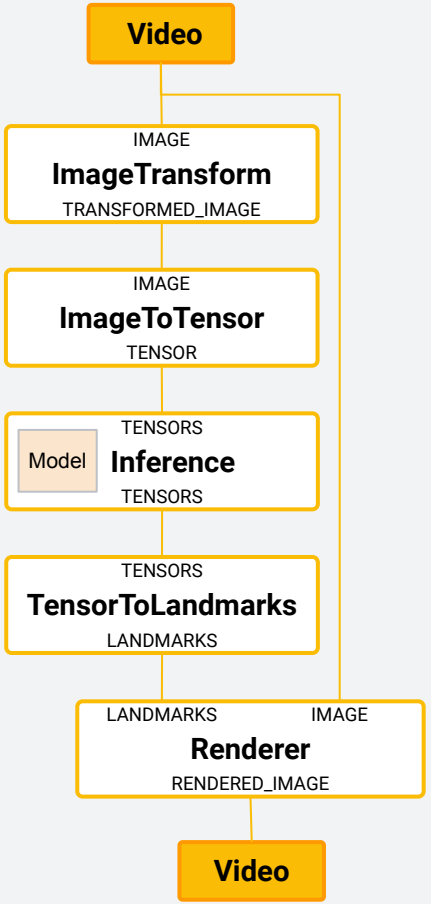


Google

Calculator

- Written in C++
- Declares input/output ports
  - Type of packet payload through the port
  - Ports can be declared as optional
- Implements **Open/Process/Close** methods, called by framework
  - **Open** - Before a full graph run
  - **Process** - Repeatedly with incoming packet(s) ready to be processed
  - **Close** - After a full graph run

Proprietary + Confidential



Google

Proprietary + Confidential

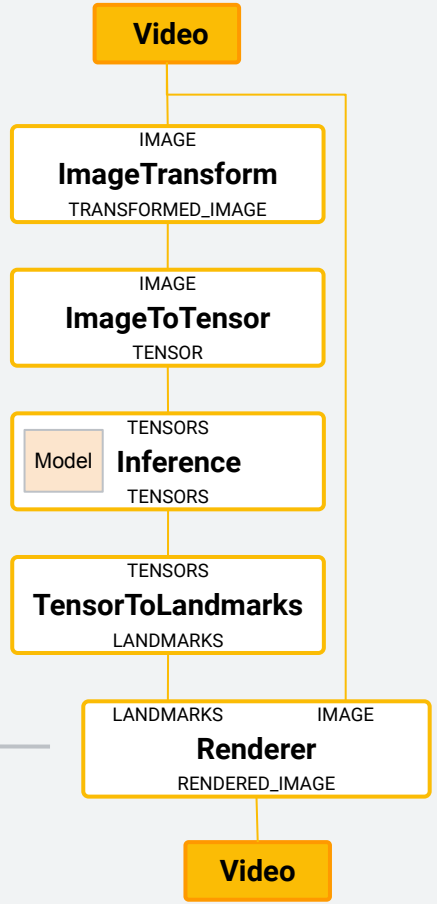
Built-in Calculators

Family of **image and media processing** calculators

Native integration with TensorFlow and TF Lite for **ML inference**

Family of **ML post-processing** calculators for common ML tasks, e.g., detection, segmentation and classification

Family of **utility** calculators for, e.g., image annotation, flow control



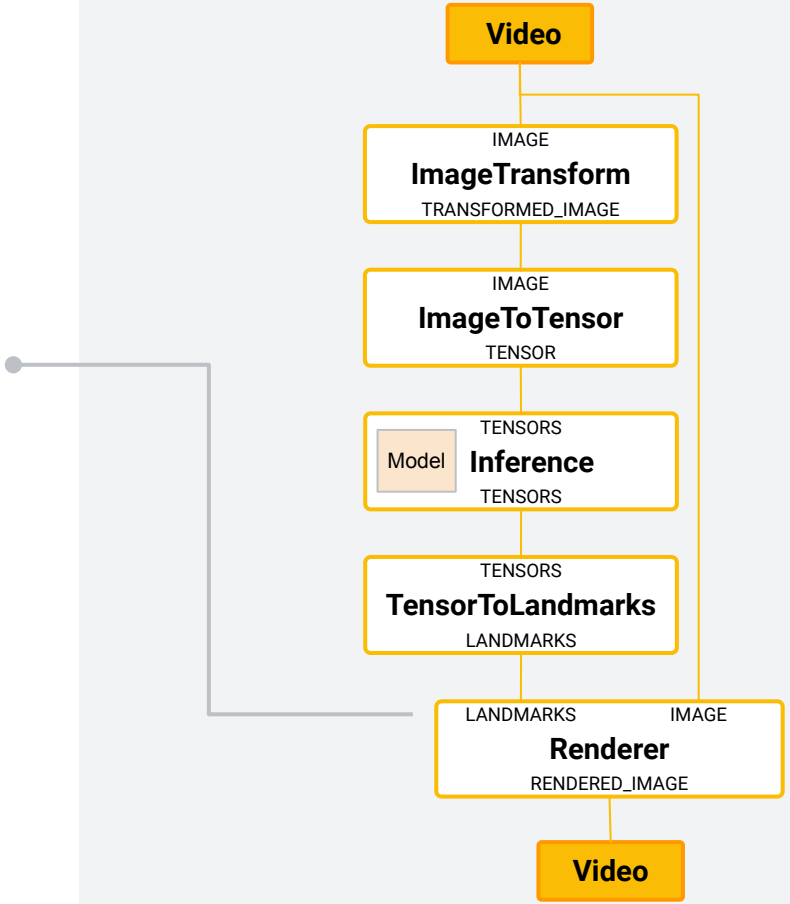
Google

Synchronization

Proprietary + Confidential

Synchronization handled by MediaPipe framework to align time-series data

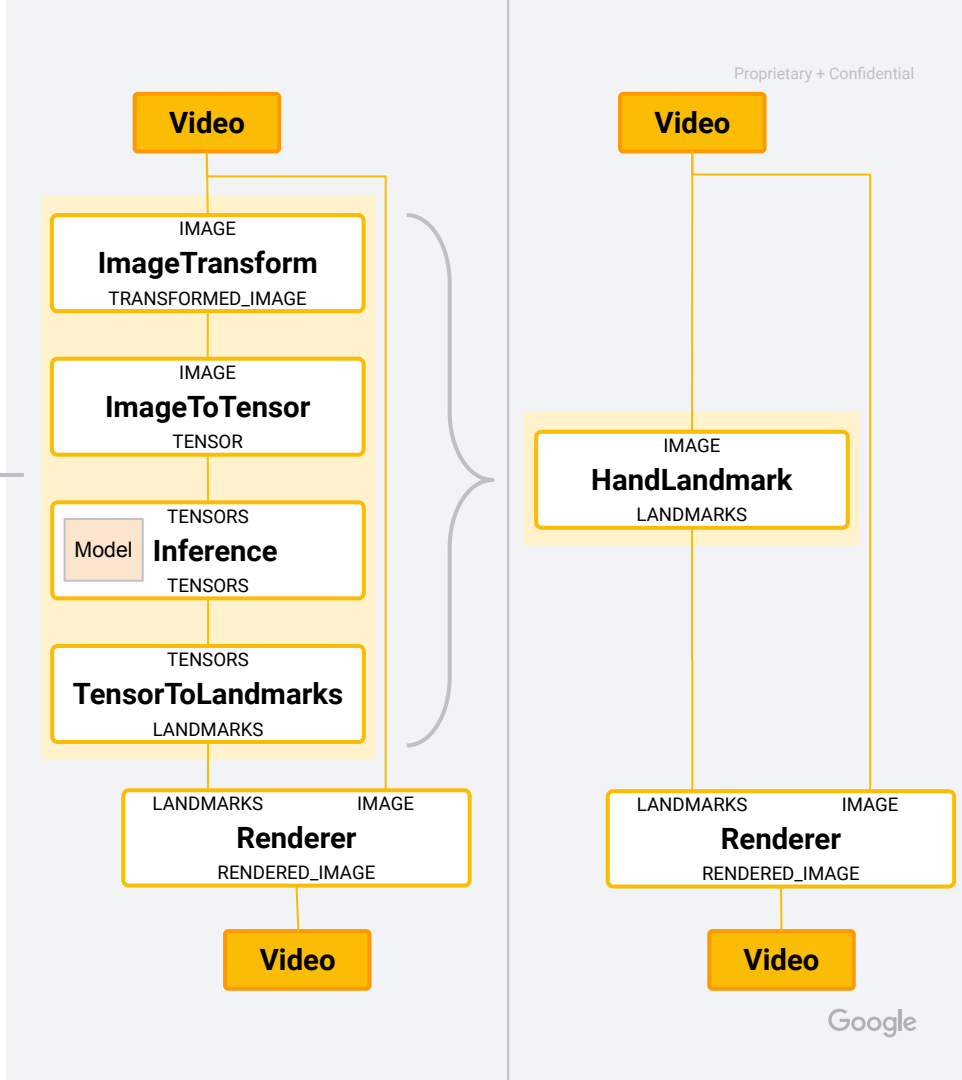
- By default all inputs to a calculator are synchronized
- **Process** method in a calculator is invoked with packets with the same timestamp across inputs



Google

Subgraph

A (partial) graph can be declared as a **subgraph** and used as a regular calculator in other graphs



Proprietary + Confidential

Simple Perception Pipeline

Remaining issues:

- In practice, a hand can appear anywhere in an image, at very different scales
- Takes a lot of model capacity to deal with the large variation in location & scale
- The model is either large/slow or inaccurate

Google

Proprietary + Confidential

Localize with Palm Detection

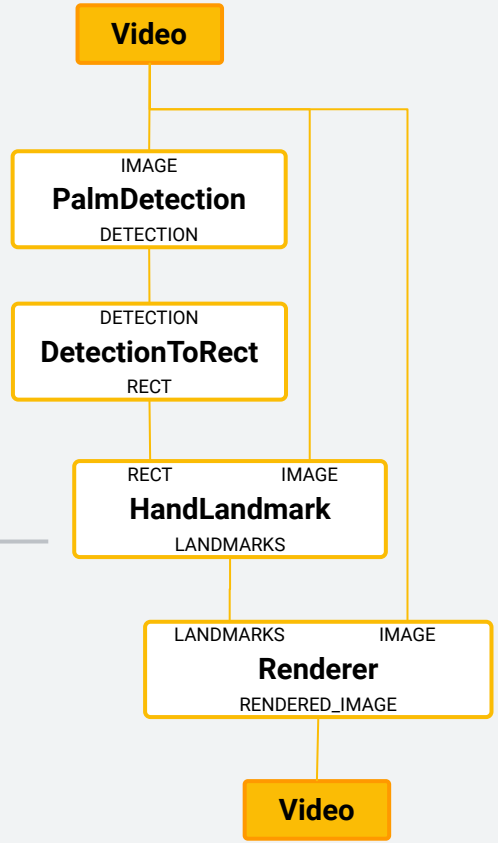
PalmDetection subgraph with another ML model

- Palms more rigid than hands with articulated fingers
- Palms are squarish, can ignore other aspect ratios

HandLandmark subgraph modified to take a RECT

- Indicating local area with likely hand presence
- Dedicate model capacity to hand landmarks instead of hand localization

PalmDetection still slow, limiting pipeline throughput



Google

Proprietary + Confidential

Localize with Prev Landmarks

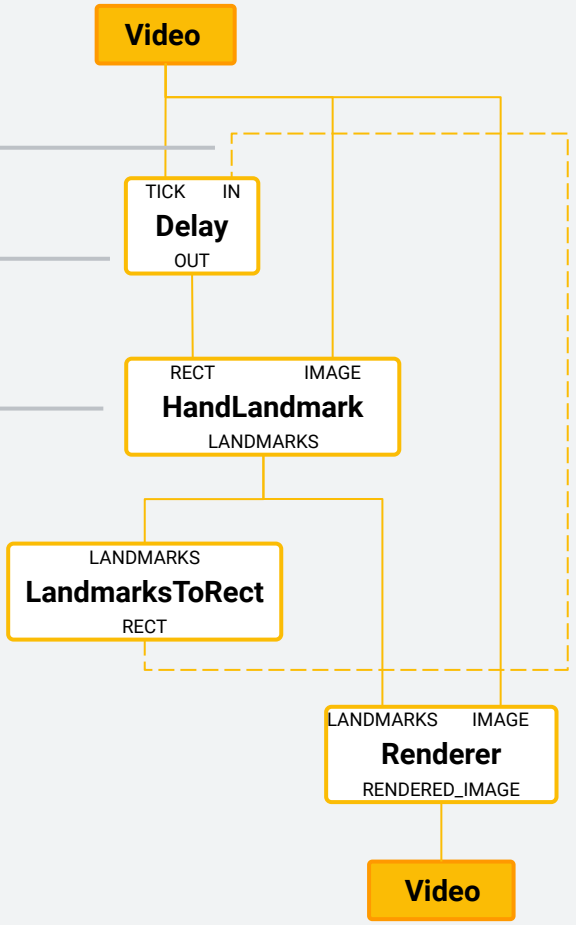
Loops allowed with back edges explicitly marked

Delays input by one frame

Runs on whole images until getting high-confidence landmarks

Derives RECT enclosing landmarks, indicating local area with likely hand presence in the next frame

Results inaccurate when running on whole image



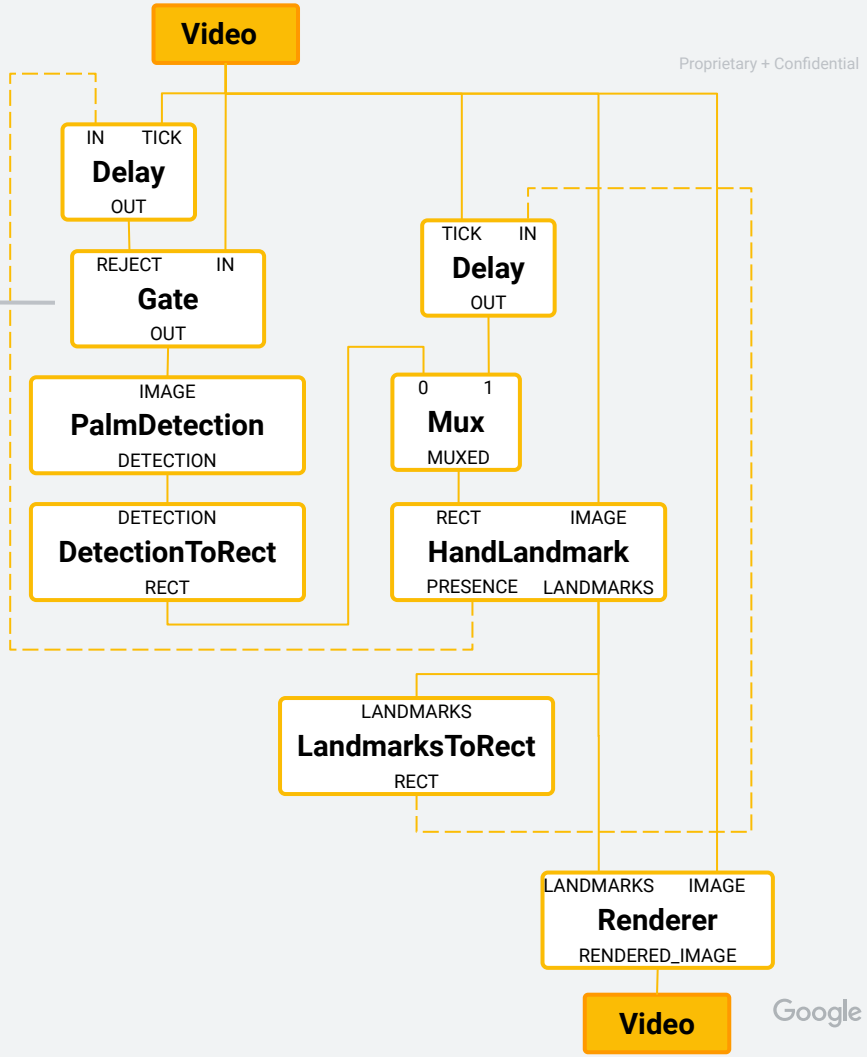
Google

Hybrid Hand Localization

Passes incoming video through only when HandLandmark has low confidence

High throughput and low resource consumption

- (Faster) HandLandmark runs every frame
- (Slower) PalmDetection runs only as needed

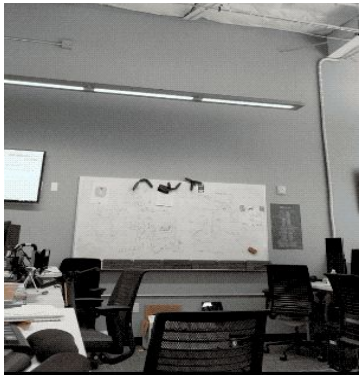


Proprietary + Confidential

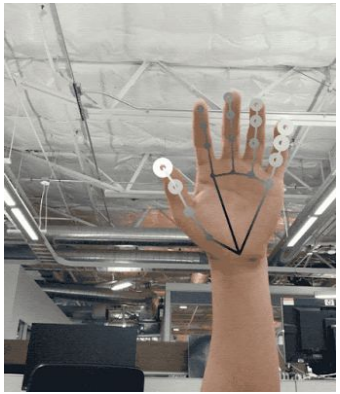
Google

Multi-models palm detection + landmark regression

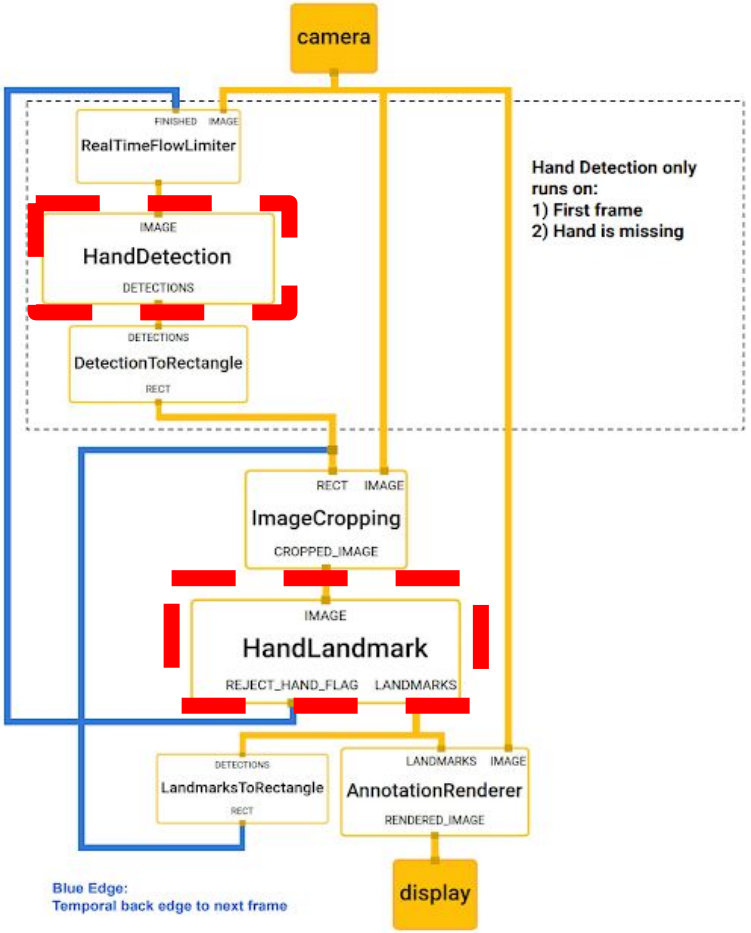
Proprietary + Confidential



Hand detection



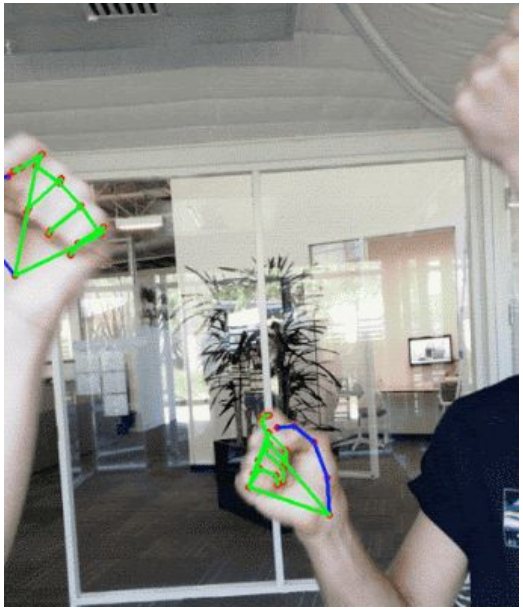
Hand landmarks



Proprietary + Confidential

Google AI Blog

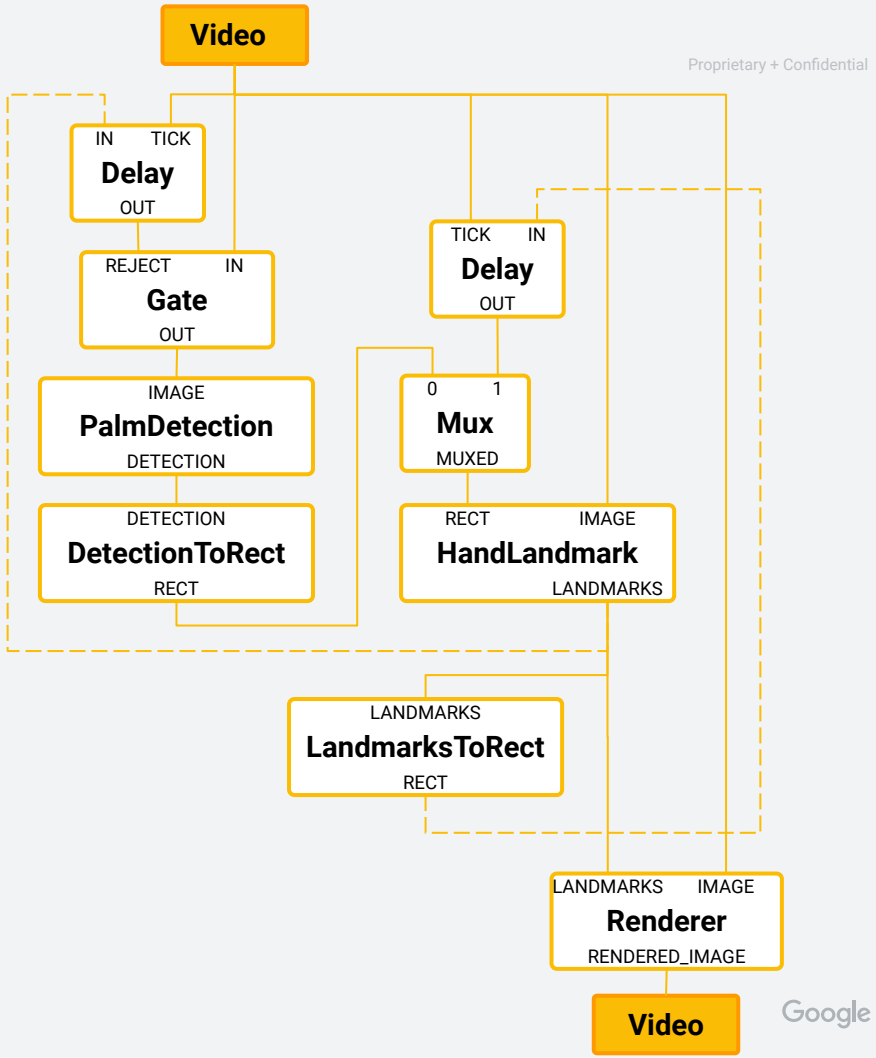
"*On-Device, Real-Time Hand Tracking with MediaPipe*", Aug '19



Performance Optimization

- GPU acceleration
  - Many built-in calculators come with options for GPU acceleration
  - Framework handles context sharing and synchronization, and interop with CPU calculators
- Executor
  - Built-in **ThreadPoolExecutor**, also accepts user-defined ones
  - In graph configuration, can assign calculators to run by different executors

Example: Run one branch of graph on GPU, and another branch on CPU (or a different GPU context) for improved resource utilization

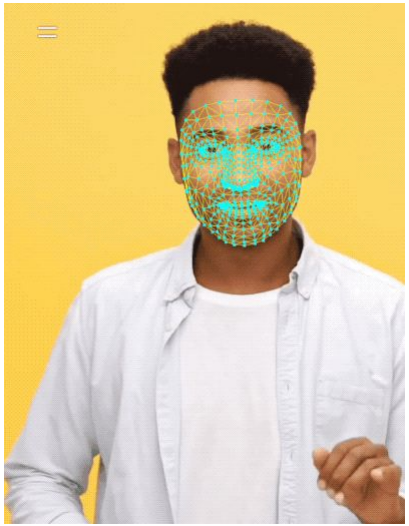


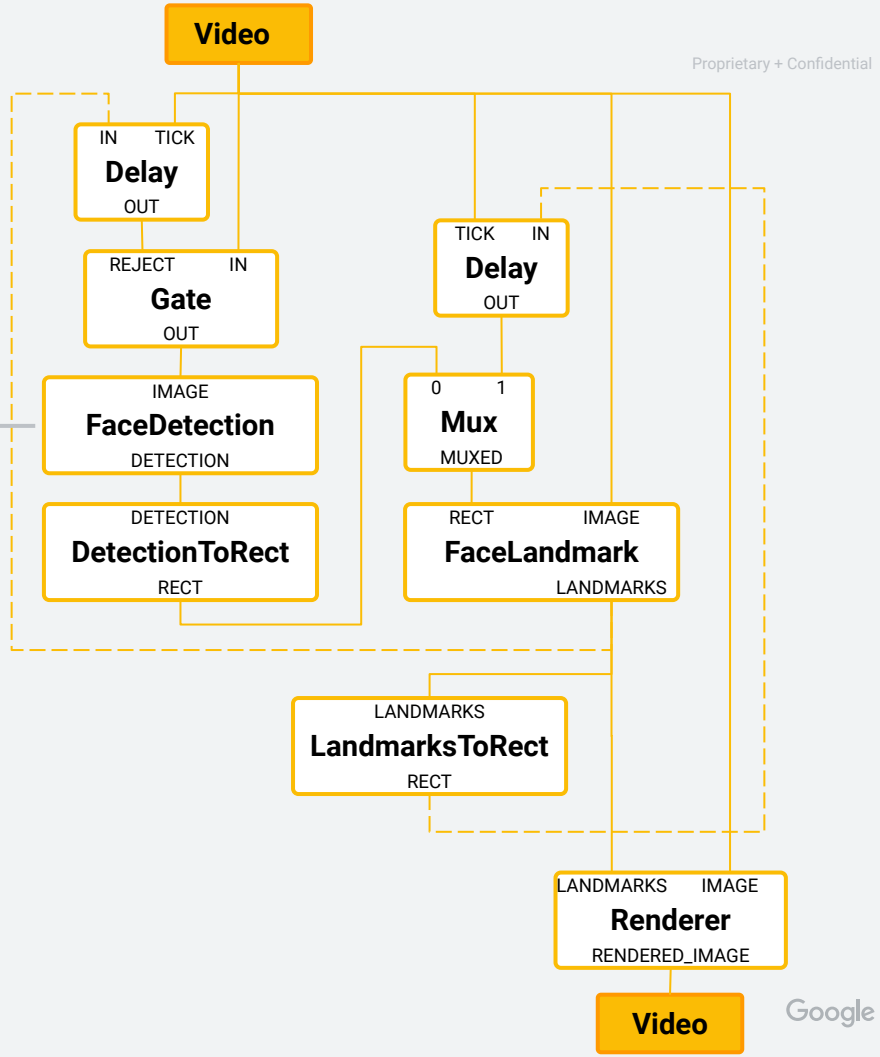
Proprietary + Confidential

Google

Customization & Extension

Replace PalmDetection with FaceDetection,
HandLandmark with FaceLandmark



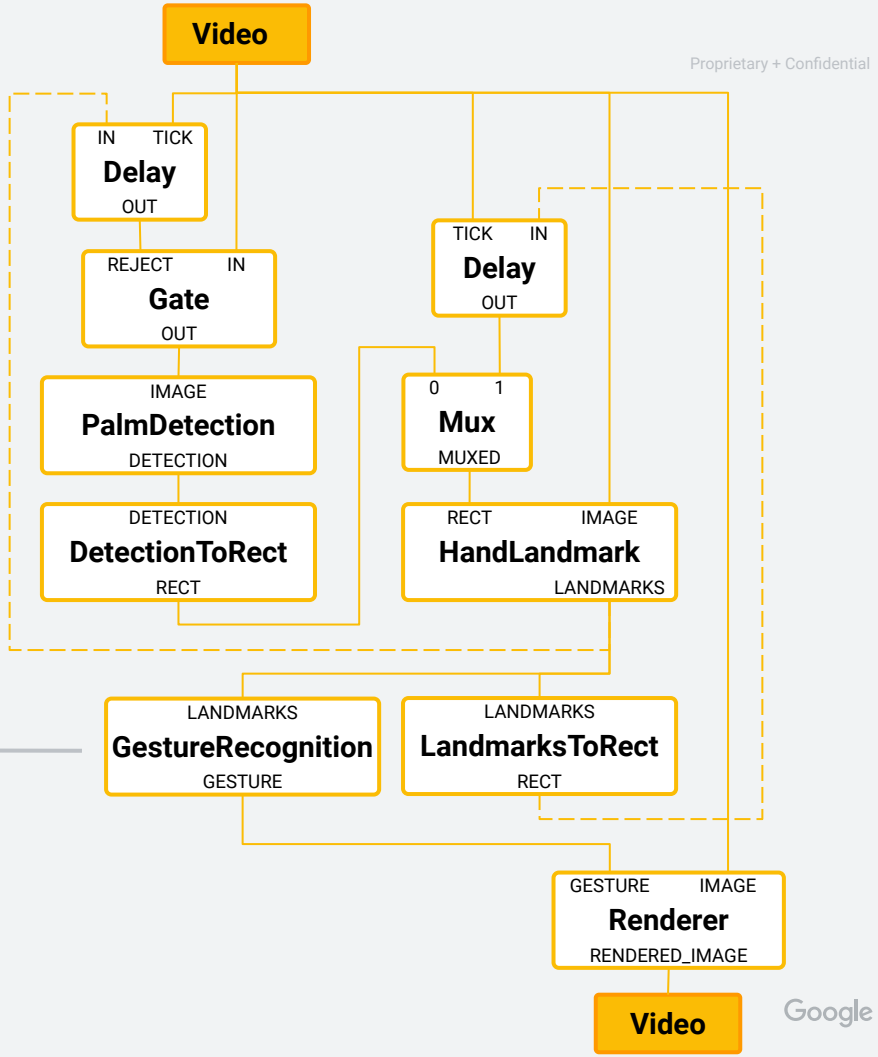


Proprietary + Confidential

Google

Customization & Extension

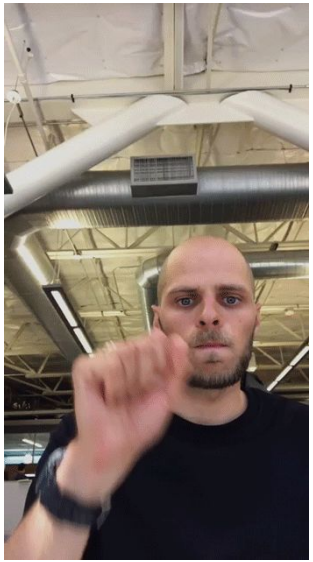
Add a calculator to infer gestures from landmarks



Proprietary + Confidential

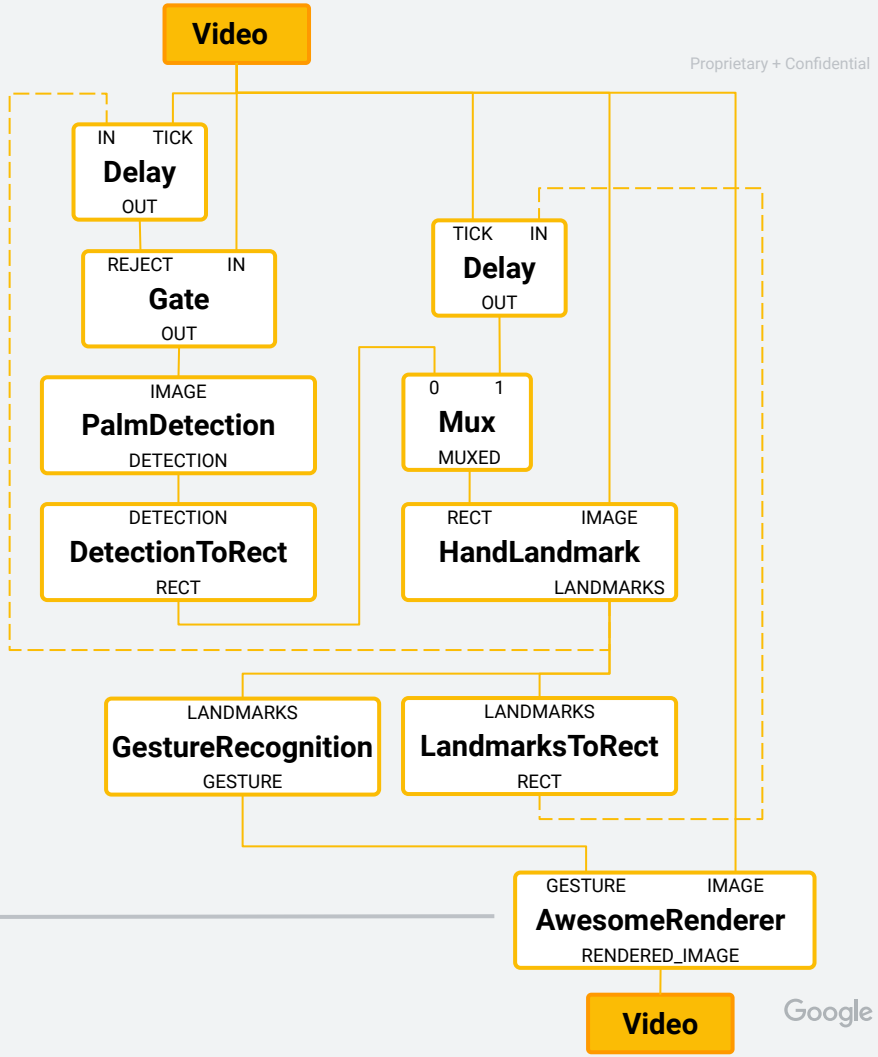
Google

Customization & Extension



* UX concept

Implement a custom renderer calculator for production-ready camera effects



Proprietary + Confidential

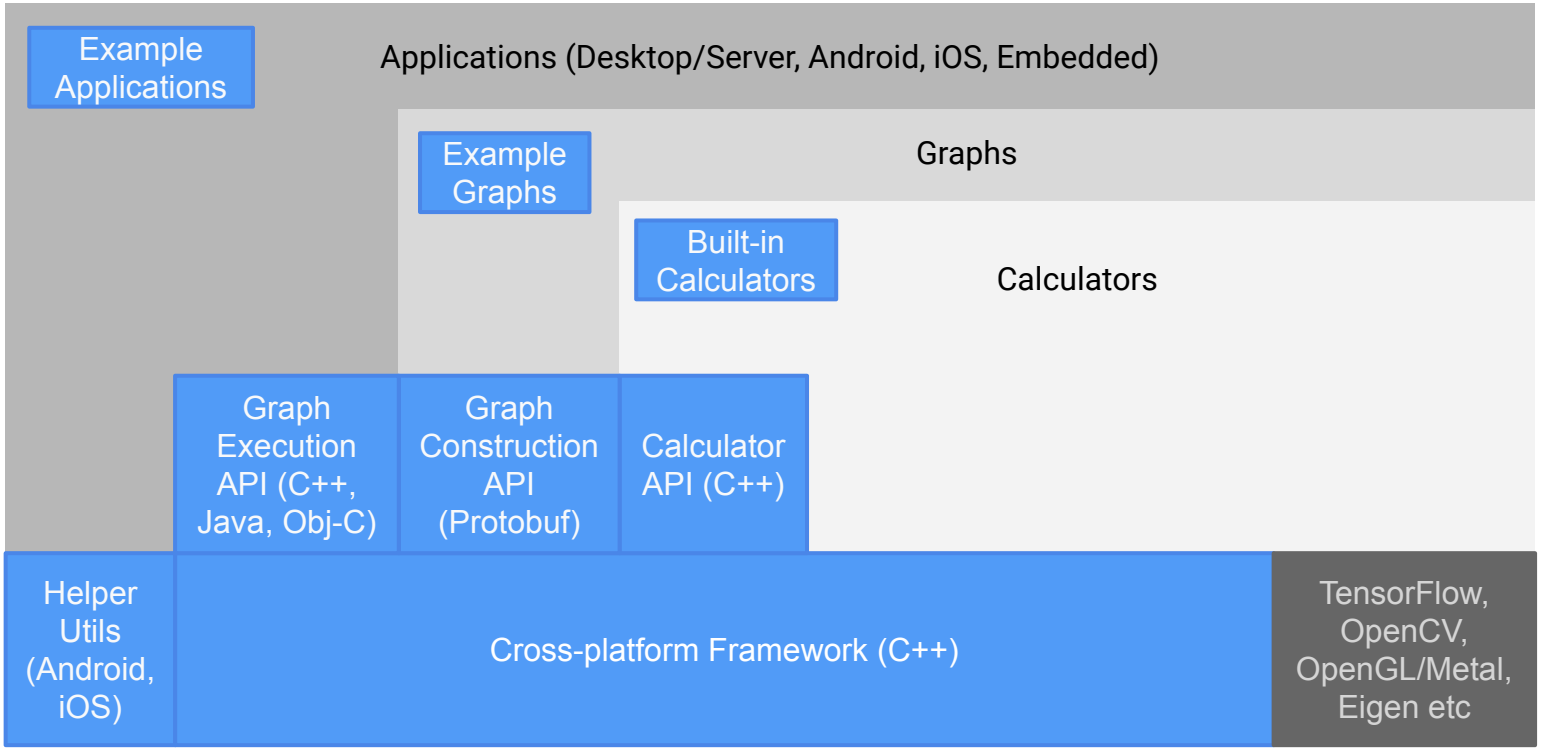
Google

Proprietary + Confidential

MediaPipe Toolkit

Proprietary + Confidential

MediaPipe Tech Stack



Google

Proprietary + Confidential

docs.mediapipe.dev

MediaPipe
latest

Search docs

- Installing MediaPipe
- MediaPipe Concepts
- Building MediaPipe Calculators
- Examples
- Visualizing MediaPipe Graphs
- Measuring Performance
- Questions and Answers
- Troubleshooting
- Getting help
- Framework Concepts
- Running on GPUs
- Framework Architecture
- License

DigitalOcean
New: DigitalOcean Marketplace Deploy your favorite dev tools with 1-Click Apps.
Sponsored · Ads served ethically

Read the Docs v: latest

Docs » MediaPipe

Edit on GitHub

MediaPipe

MediaPipe is a graph-based framework for building multimodal (video, audio, and sensor) applied machine learning pipelines. MediaPipe is cross-platform running on mobile devices, workstations and servers, and supports mobile GPU acceleration. With MediaPipe, an applied machine learning pipeline can be built as a graph of modular components, including, for instance, inference models and media processing functions. Sensory data such as audio and video streams enter the graph, and perceived descriptions such as object-localization and face-landmark streams exit the graph. An example graph that performs real-time hair segmentation on mobile GPU is shown below.

MediaPipe is designed for machine learning (ML) practitioners, including researchers, students, and software developers, who implement production-ready ML applications, publish code accompanying research work, and build technology prototypes. The main use case for MediaPipe is rapid prototyping of applied machine learning pipelines with inference models and other reusable components. MediaPipe also facilitates the deployment of machine learning technology into demos and applications on a wide variety of different hardware platforms (e.g., Android, iOS, workstations).

APIs for MediaPipe

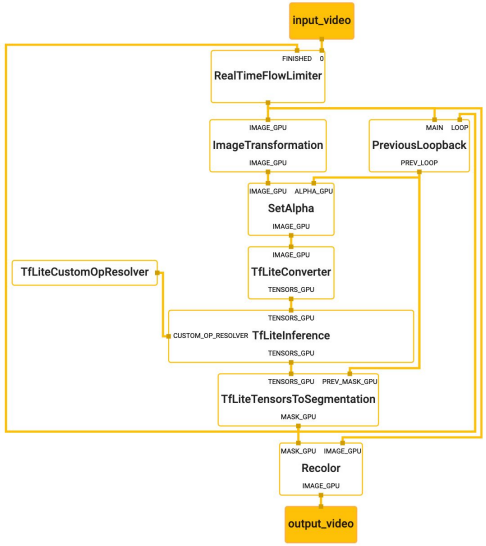
- Calculator API in C++
- Graph Construction API in ProtoBuf
- (Coming Soon) Graph Construction API in C++
- Graph Execution API in C++

viz.mediapipe.dev

MediaPipe Visit us at mediapipe.dev

New Upload

Graph (graph.pbtxt)



Editor (graph.pbtxt)

```
# "previous mask" share the same timestamp, and as a result
# and combined in the subsequent calculator. Note that upon
# very first input frame, an empty packet is sent out to jum
# loop.
node {
  calculator: "PreviousLoopbackCalculator"
  input_stream: "MAIN:throttled_input_video"
  input_stream: "LOOP:hair_mask"
  input_stream_info: {
    tag_index: "LOOP"
    back_edge: true
  }
  output_stream: "PREV_LOOP:previous_hair_mask"
}

# Embeds the hair mask generated from the previous round of
# as the alpha channel of the current input image.
node {
  calculator: "SetAlphaCalculator"
  input_stream: "IMAGE_GPU:transformed_input_video"
  input_stream: "ALPHA_GPU:previous_hair_mask"
  output_stream: "IMAGE_GPU:mask_embedded_input_video"
}

# Converts the transformed input image on GPU into an image
# tflite::gpu::GlBuffer. The zero_center option is set to fa
# pixel values to [0.f, 1.f] as opposed to [-1.f, 1.f]. The
# option is set to true to account for the descrepancy betwe
# representation of the input image (origin at the bottom-le
# OpenGL convention) and what the model used in this graph i
# at the top-left corner). With the max_num_channels option
# channels are contained in the image tensor.
node {
  calculator: "TfLiteConverterCalculator"
  input_stream: "IMAGE_GPU:mask_embedded_input_video"
  output_stream: "TENSORS_GPU:image_tensor"
  node_options: {
    [type.googleapis.com/drishti.TfLiteConverterCalculatorOp
      zero_center: false
      flip_vertically: true
      max_num_channels: 4
    }
```

Feedback (graph.pbtxt)

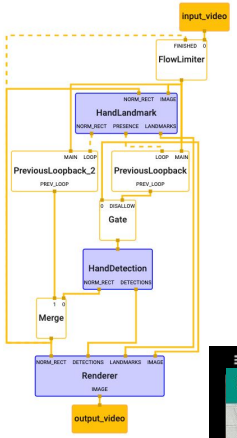
2 Editor Parser ran successfully.

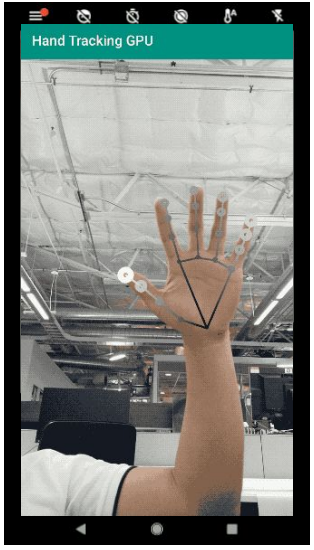
Google Terms of Service Google Privacy Notice

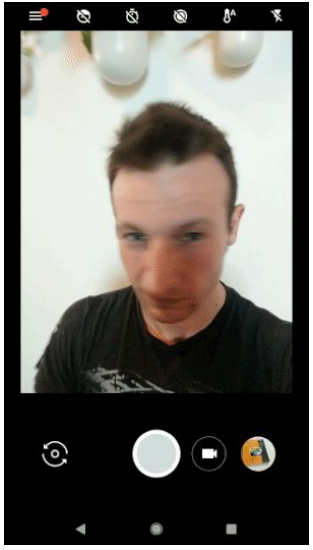
Google

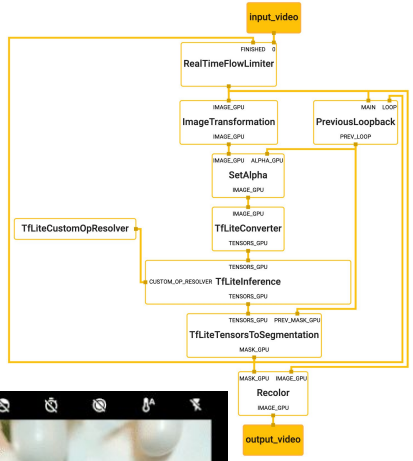
Proprietary + Confidential

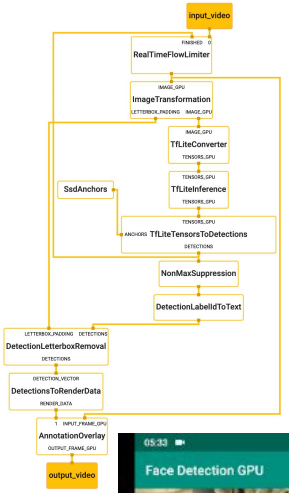
Mobile Examples

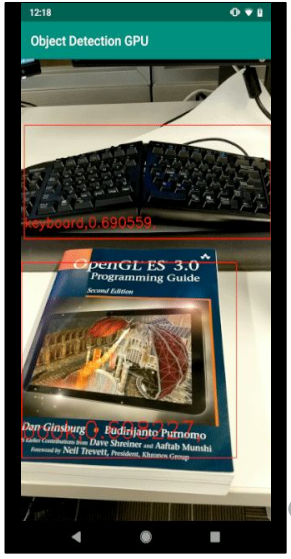


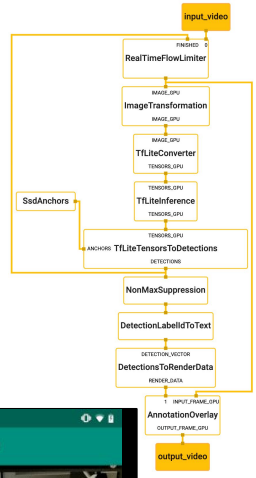












Google

Proprietary + Confidential

Keep track of MediaPipe

Google Developers

Blog of our latest news, updates, and stories for developers

https://mediapipe.dev

Object Detection and Tracking using MediaPipe

Tuesday, December 10, 2019

Posted by Ming Guang Yong, Product Manager for MediaPipe

MediaPipe in 2019

MediaPipe is a framework for building cross platform multimodal applied ML pipelines that consist of fast ML inference, classic computer vision, and media processing (e.g. video decoding). MediaPipe was open sourced at CVPR in June 2019 as v0.5.0. Since our first open source version, we have released various ML pipeline examples like

- Object Detection
- Face Detection
- Hand Tracking
- Multi-hand Tracking
- Hair Segmentation

https://mediapipe.page.link/devblog

Twitter: realmgyong@

Proprietary + Confidential

Thank you!

mediapipe.dev