Vision Framework

Building on Core ML

Session 506

Brett Keating, Apple Manager
Frank Doepke, He who wires things together
What Can Vision Do
Vision Concepts
The Code
What Can Vision Do
Vision Concepts
The Code
What Can Vision Do
Vision Concepts
The Code
What You Can Do with Vision
Face Detection
Face Detection: Small Faces
Face Detection: Strong Profiles
Face Detection: Partially Occluded
Face Detection: Hats and Glasses
Face Landmarks
Face Landmarks
Face Landmarks
Face Landmarks
Image Registration
Image Registration
Rectangle Detection
Barcode Detection
Global Accessibility Awareness Day
Object Tracking
For faces, rectangles, and general templates
Integration with Core ML

Significant advances in Computer Vision through Machine Learning

Core ML provides native acceleration for custom models

Vision provides the imaging pipeline to support Core ML models

Classification: Ballpoint, ballpoint pen, ballpen

Confidence: 0.86

Core ML and Natural Language Processing Lab | Technology Lab D | Thu 11:00AM-3:30PM
---|---|---
Core ML and Natural Language Processing Lab | Technology Lab D | Fri 1:50PM-4:00PM
High-level on-device solutions to Computer Vision problems through one simple API
High-Level Solutions
High-Level Solutions

You don’t have to be a Computer Vision expert
High-Level Solutions

You don’t have to be a Computer Vision expert

“I just want to know where the faces are”
High-Level Solutions

You don’t have to be a Computer Vision expert

“I just want to know where the faces are”

Handles the complexity for you
High-Level Solutions

You don’t have to be a Computer Vision expert

“I just want to know where the faces are”

Handles the complexity for you

Traditional and deep learning algorithms
On Device vs. Cloud
On Device vs. Cloud

Privacy

- Images and video stay on device
On Device vs. Cloud

Privacy
• Images and video stay on device

Cost
• No usage fees
• No data transfer
On Device vs. Cloud

Privacy
• Images and video stay on device

Cost
• No usage fees
• No data transfer

Real-time use cases
• No latency, fast execution
Analyzing an Image

- The Asks
- The Machinery
- The Results
Analyzing an Image

Requests

VNDetectBarcodesRequest
VNDetectFaceLandmarksRequest
VNDetectFaceRectanglesRequest
...

The Machinery

The Results
Analyzing an Image

Requests

VNDetectBarcodesRequest
VNDetectFaceLandmarksRequest
VNDetectFaceRectanglesRequest
...

RequestHandler

VNImageRequestHandler

The Results
Analyzing an Image

Requests
- VNDetectBarcodesRequest
- VNDetectFaceLandmarksRequest
- VNDetectFaceRectanglesRequest
- ...

RequestHandler
- VNImageRequestHandler

Observations
- VNClassificationObservation
- VNDetectedObjectObservation
- VNFaceObservation
- ...

VNDetectFaceLandmarksRequest
Tracking in a Sequence

- The Asks
- The Machinery
- The Results
Tracking in a Sequence

The Machinery

Requests
- VNTrackObjectRequest
- VNTrackRectangleRequest
- VNImageRegistrationRequest

The Results
Tracking in a Sequence

Requests
- VNTrackObjectRequest
- VNTrackRectangleRequest
- VNImageRegistrationRequest

RequestHandler
- VNSequenceRequestHandler

The Results
Tracking in a Sequence

Requests
- VNTrackObjectRequest
- VNTrackRectangleRequest
- VNImageRegistrationRequest

RequestHandler
- VNSequenceRequestHandler

Observations
- VNDetectedObjectObservation
- VNRectangleObservation
- VNImageAlignmentObservation
Image Request Handler

For interactive exploration of an image

Holds on to the image for its lifecycle

Allows optimization of various requests performed on an image
Sequence Request Handler

For anything that looks at images in a sequence like tracking

Does not optimize for multiple requests on an image
Putting It into Code
// Create request
let faceDetectionRequest = VNDetectFaceRectanglesRequest()
// Create request
let faceDetectionRequest = VNDetectFaceRectanglesRequest()

// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])
// Create request
let faceDetectionRequest = VNDetectFaceRectanglesRequest()

// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// send the requests to the request handler
myRequestHandler.perform([faceDetectionRequest])
let faceDetectionRequest = VNDetectFaceRectanglesRequest()

let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

myRequestHandler.perform([faceDetectionRequest])

for observation in faceDetectionRequest.results as! [VNFaceObservation] {
    /// do something
}

/// do something
// Create request
let faceDetectionRequest = VNDetectFaceRectanglesRequest()

// Create request handler
let myRequestHandler = VNImageRequestHandler(url: fileURL, options: [:])

// send the requests to the request handler
myRequestHandler.perform([faceDetectionRequest])

// Do we have a face
for observation in faceDetectionRequest.results as! [VNFaceObservation] {
    /// do something
}

/// do something
// Create a sequence request handler

let requestHandler = VNSequenceRequestHandler()
// Create a sequence request handler
let requestHandler = VNSequenceRequestHandler()

// Start the tracking with an observation
let observations = detectionRequest.results as! [VNDetectedObjectObservation]
let objectsToTrack = observations.map { VNTrackObjectRequest(detectedObjectObservation: $0) }
// Create a sequence request handler
let requestHandler = VNSequenceRequestHandler()

// Start the tracking with an observation
let observations = detectionRequest.results as! [VNDetectedObjectObservation]
let objectsToTrack = observations.map { VNTrackObjectRequest(detectedObjectObservation: $0) }

// Run the requests
requestHandler.perform(objectsToTrack, on: pixelBuffer)
// Create a sequence request handler
let requestHandler = VNSequenceRequestHandler()

// Start the tracking with an observation
let observations = detectionRequest.results as! [VNDetectedObjectObservation]
let objectsToTrack = observations.map { VNTrackObjectRequest(detectedObjectObservation: $0) }

// Run the requests
requestHandler.perform(objectsToTrack, on: pixelBuffer)

// Lets look at the results
for request in objectsToTrack
    for observation in request.results as! [VNDetectedObjectObservation]
Best Practices
Envisioning a Vision Task
Envisioning a Vision Task

Which image type is right for me?
Envisioning a Vision Task

Which image type is right for me?

What am I going to do with the image?
Envisioning a Vision Task

Which image type is right for me?

What am I going to do with the image?

What performance do I need or want?
Which Image Type Is Right for Me?
Which Image Type Is Right for Me?

Vision supports various image types:
- CVPixelBufferRef
- CGImageRef
- CIImage
- NSURL
- NSData
Which Image Type Is Right for Me?

Vision supports various image types:
- CVPixelBufferRef
- CGImageRef
- CIImage
- NSURL
- NSData

The image type to choose depends on where the image comes from.
Which Image Type Is Right for Me?

Vision supports various image types

CVPixelBufferRef
CGImageRef
CIImage
NSURL
NSData

The image type to choose depends on where the image comes from

You shouldn’t have to pre-scale the image
Which Image Type Is Right for Me?

Vision supports various image types

CVPixelBufferRef
CGImageRef
CIImage
NSURL
NSData

The image type to choose depends on where the image comes from

You shouldn’t have to pre-scale the image

Make sure to pass in the EXIF orientation of the image
Everything Streaming

CVPixelBuffer

Comes from a CMSampleBuffer in the VideoDataOut of a camera stream

Also a good low-level format to provide image data in memory
Files from Disk or Web

URL for image files on disk

NSData for images from the web

Least amount of memory footprint

Vision will do the scaling without reading the full image if possible

EXIF Orientation is derived from the file if possible but can be overwritten
Core Image

Already using Core Image

Preprocessing the image
Images Already Used in the UI

Use CGImage if the image was already used in the UI

UIImage and NSImage have accessors for CGImageRefs
What Am I Going to Do with the Image?
What Am I Going to Do with the Image?

Interactively explore the image
• Use VNImageRequestHandler and hold onto it
• Remember that the input image is immutable
What Am I Going to Do with the Image?

Interactively explore the image
• Use VNImageRequestHandler and hold onto it
• Remember that the input image is immutable

Tracking an observation
• Use VNSequenceRequestHandler
• Tracking state is kept in the VNSequenceRequestHandler
• Lifecycle of images is not tied to the life of the VNSequenceRequestHandler
What Performance Do I Need or Want?

Vision tasks can be time consuming and processing intensive
• Dispatch your work on a queue with appropriate QOS
• Use the completion handler to work with the results
• Completion handler is called on the same queue as the request
Yet Another Face Detector?

Vision uses deep learning for face detection

• Highest precision and recall
• Slower on older hardware in particular
## Face Detector Landscape

<table>
<thead>
<tr>
<th>Feature</th>
<th>Vision</th>
<th>Core Image</th>
<th>AV Capture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Best</td>
<td>Better</td>
<td>Good</td>
</tr>
<tr>
<td>Processing time</td>
<td>Fast</td>
<td>Faster</td>
<td>Fastest</td>
</tr>
<tr>
<td>Power usage</td>
<td>Good</td>
<td>Better</td>
<td>Best</td>
</tr>
<tr>
<td>Availability</td>
<td>iOS, macOS, tvOS</td>
<td>iOS, macOS, tvOS</td>
<td>iOS capture only</td>
</tr>
</tbody>
</table>
Core Image Vision
CIDetector vs. Vision

CIDetector will remain as they are in Core Image

New algorithms will be exposed through Vision

Algorithm improvements will be made available in Vision
Show and Tell
Part one
Show and Tell

Part two
MNISTVision

Concepts to be covered

• Spin off requests from other requests
• Use Core Image for processing
• Use Core ML for machine learning

MNIST Samples
The Flow Chart
The Flow Chart

Step 1
Find sticky note using
Rectangle Detector
The Flow Chart

Step 1
Find sticky note using Rectangle Detector

Step 2
Use CI for perspective correction and image processing
The Flow Chart

Step 1
Find sticky note using Rectangle Detector

Step 2
Use CI for perspective correction and image processing

Step 3
Run MNIST classifier on resulting CIImage

Confidence: 0.87
MNISTClassifier: 4
Demo
And that is Vision.framework
Summary

Vision is a new high-level framework for Computer Vision

Various detectors and tracking through one consistent interface

Integration with Core ML allows you to use custom models with ease
More Information

<table>
<thead>
<tr>
<th>Session</th>
<th>Venue</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing Core ML</td>
<td>WWDC 2017</td>
<td></td>
</tr>
<tr>
<td>Photography Get Together</td>
<td>Technology Lab J</td>
<td>Wednesday  6:30PM</td>
</tr>
<tr>
<td>Core ML in Depth</td>
<td>Hall 3</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Advances in Core Image: Filters, Metal, Vision, and More</td>
<td>Executive Ballroom</td>
<td>Thursday 1:50PM</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
<td>Days</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Core ML and Natural Language Processing Lab</td>
<td>Technology Lab D</td>
<td>Thur</td>
</tr>
<tr>
<td>AVFoundation Lab</td>
<td>Technology Lab F</td>
<td>Thur</td>
</tr>
<tr>
<td>Photos Editing and Core Image Lab</td>
<td>Technology Lab F</td>
<td>Thur</td>
</tr>
<tr>
<td>Vision Lab</td>
<td>Technology Lab A</td>
<td>Fri</td>
</tr>
<tr>
<td>Core ML and Natural Language Processing Lab</td>
<td>Technology Lab D</td>
<td>Fri</td>
</tr>
</tbody>
</table>