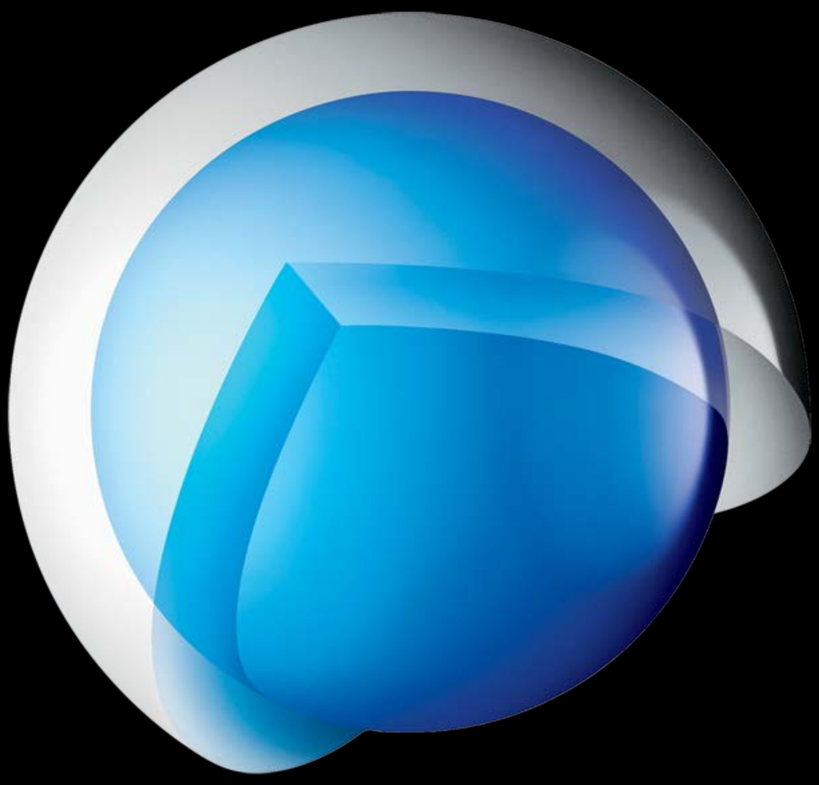
Core Image Effects and Techniques

Session 509

David Hayward
Advanced Imaging Team



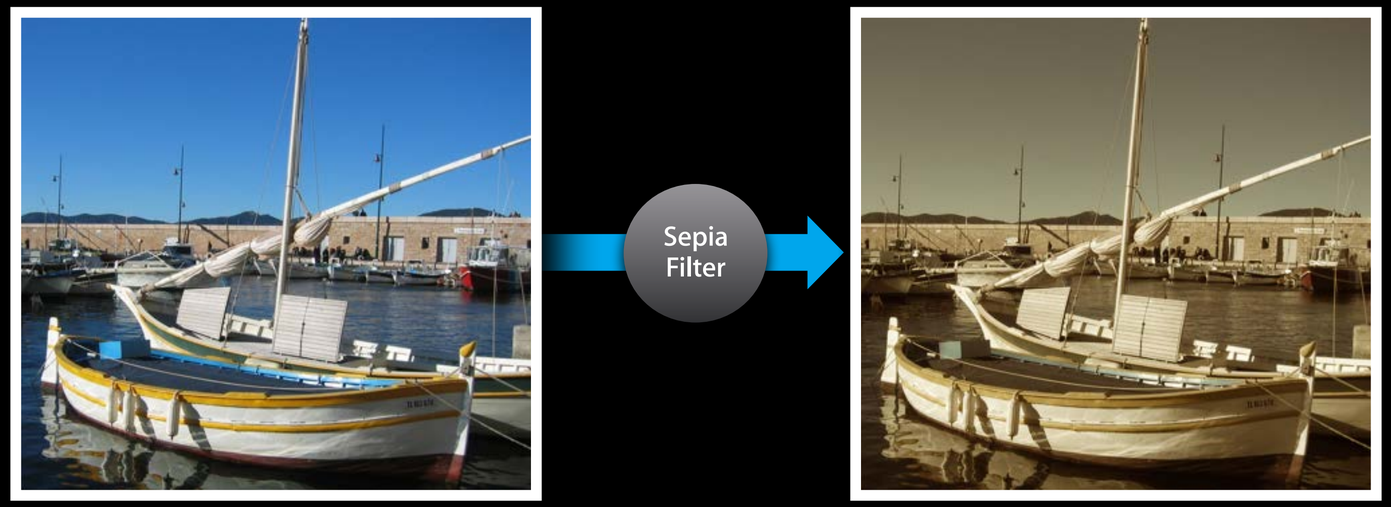
What We Will Discuss Today

- Key concepts
- Leveraging built-in filters
- Providing input images
- Rendering Core Image output
- Bridging Core Image and OpenCL

Key Concepts

Basic Concept

Filters perform per pixel operations on an image



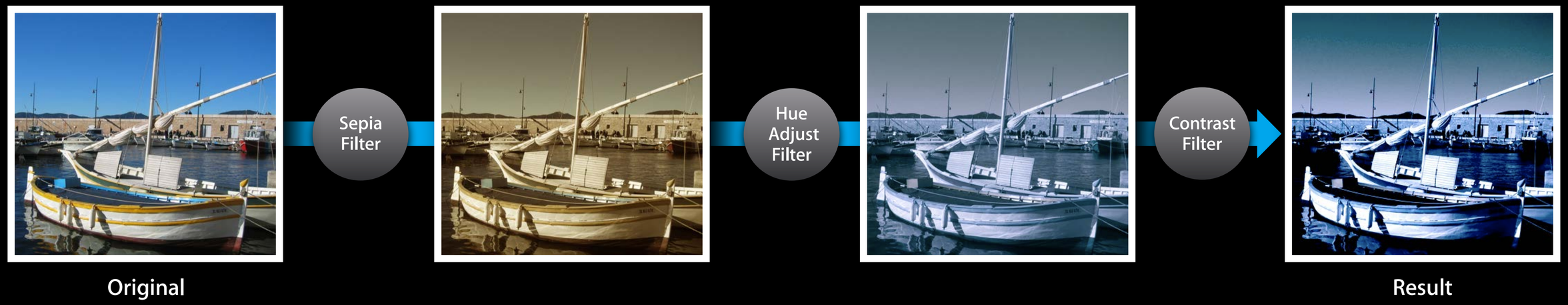
Original

Result

The final result is a new image

Basic Concept

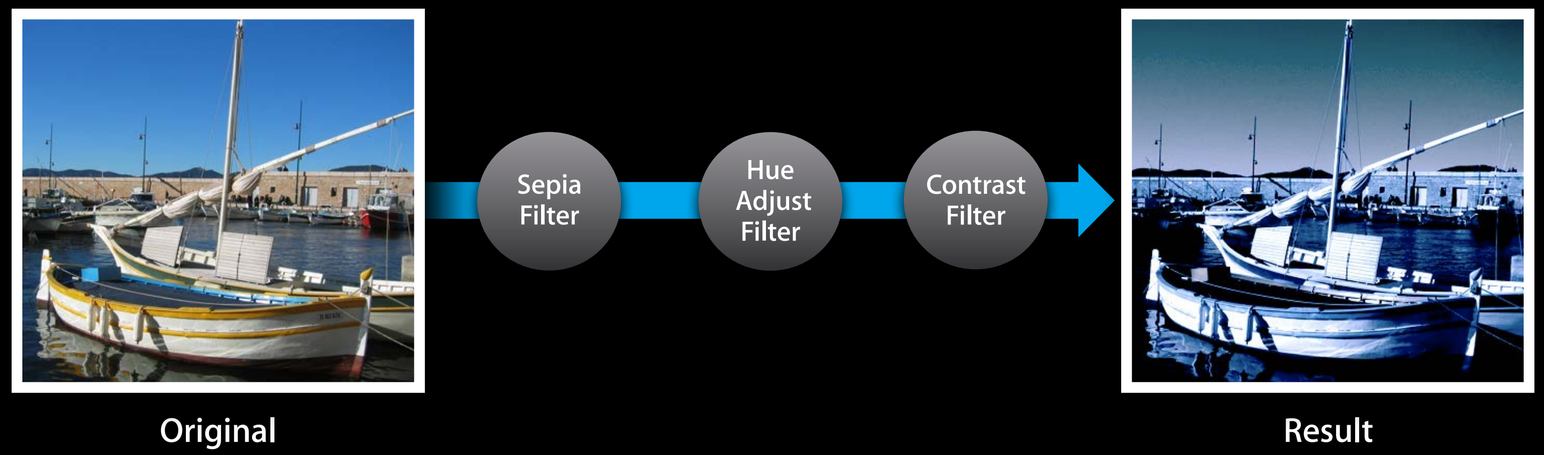
Filters can be chained together



This allows for complex effects

Basic Concept

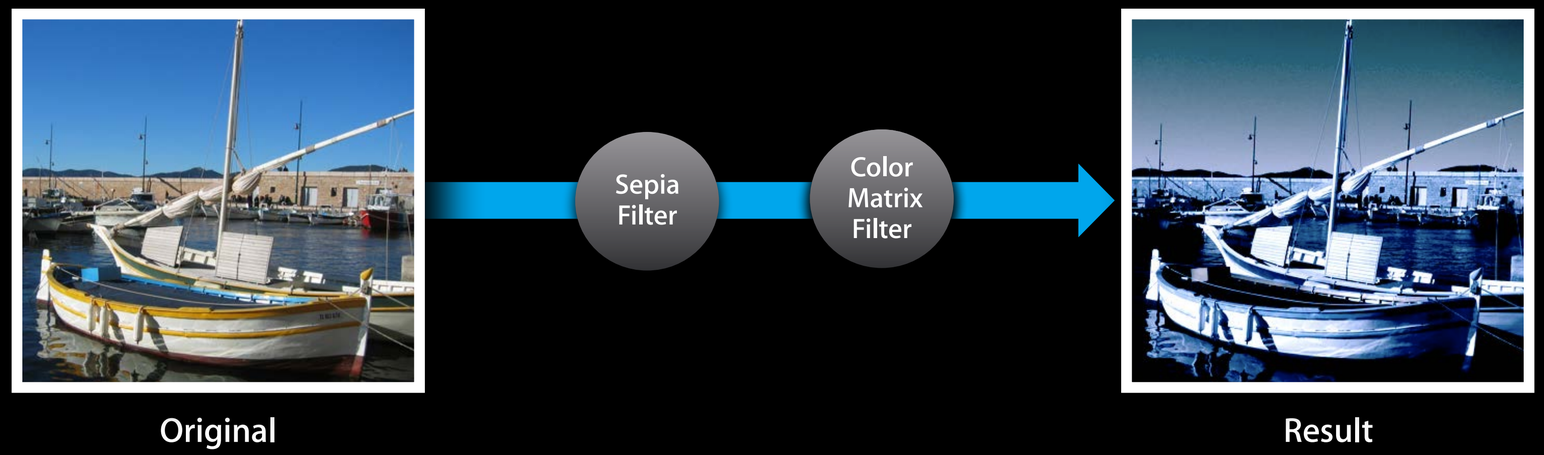
Filter chains are concatenated



This eliminates intermediate buffers

Basic Concept

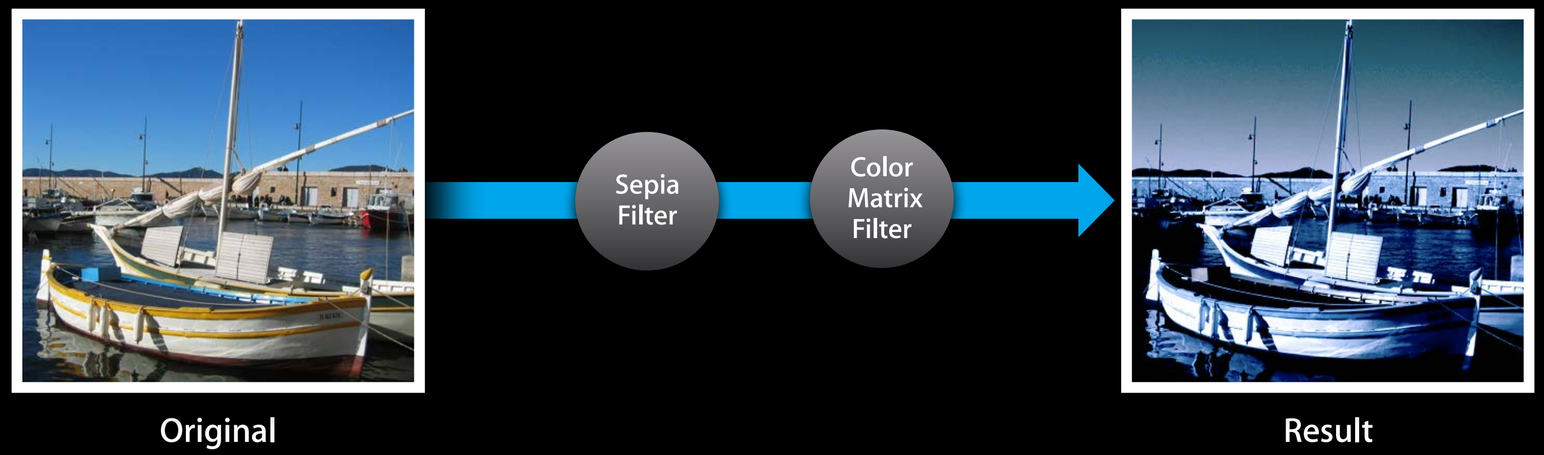
Filter chains are optimized at render time



This further improves performance

Basic Concept

Filter chains are optimized at render time



This further improves performance

Demo

Core Image Fun House on iOS

Demo

Core Image Fun House on iOS

The source is available at http://developer.apple.com/

Core Image Classes

Core Image Classes

- CIFilter
  - A mutable object that represents an effect
  - Has image or numeric input parameters
  - Produces one output image based on current inputs

Core Image Classes

- CIFilter
  - A mutable object that represents an effect
  - Has image or numeric input parameters
  - Produces one output image based on current inputs
- CIImage
  - An immutable object that represents the recipe for an image
  - Can represent a file from disk or the output of a CIFilter

Core Image Classes

- CIFilter
  - A mutable object that represents an effect
  - Has image or numeric input parameters
  - Produces one output image based on current inputs
- CIImage
  - An immutable object that represents the recipe for an image
  - Can represent a file from disk or the output of a CIFilter
- CIContext
  - A object through which Core Image draw results
  - Can be based on CPU or GPU

Platform Specifics

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable
Core API	CIFilter CIImage CIContext	CIFilter CIImage CIContext CIKernel CIFilterShape

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable
Core API	CIFilter CIImage CIContext	CIFilter CIImage CIContext CIKernel CIFilterShape
Performance	Render-time optimizations of filter graph	

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable
Core API	CIFilter CIImage CIContext	CIFilter CIImage CIContext CIKernel CIFilterShape
Performance	Render-time optimizations of filter graph	
Color Management	sRGB or non-color managed Unclamped linear working space	ICC or non-color managed Unclamped linear working space

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable
Core API	CIFilter CIImage CIContext	CIFilter CIImage CIContext CIKernel CIFilterShape
Performance	Render-time optimizations of filter graph	
Color Management	sRGB or non-color managed Unclamped linear working space	ICC or non-color managed Unclamped linear working space
Rendering	CPU or OpenGL ES 2.0	OpenCL on CPU

Platform Specifics

	iOS	OS X
Filters	100+ built-in filters	150+ built-in filters Developer extendable
Core API	CIFilter CIImage CIContext	CIFilter CIImage CIContext CIKernel CIFilterShape
Performance	Render-time optimizations of filter graph	
Color Management	sRGB or non-color managed Unclamped linear working space	ICC or non-color managed Unclamped linear working space
Rendering	CPU or OpenGL ES 2.0	OpenCL on CPU OpenCL on GPU

Demo

Core Image with OpenCL GPU on OS X Mavericks

Leveraging Built-In Filters

Lots of Useful Filters

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Color effects and adjustments

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Color effects and adjustments

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Color effects and adjustments

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Geometry and distortion effects

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Geometry and distortion effects

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Geometry and distortion effects

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenera
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMo
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp

CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Blur and sharpen effects

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Blur and sharpen effects

NEW

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Generators

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Generators

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Generators

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp

CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenera
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizon
CIConvolution9Vertical
CICopyMachineTransitio
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition

CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode

CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Detectors

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Detectors

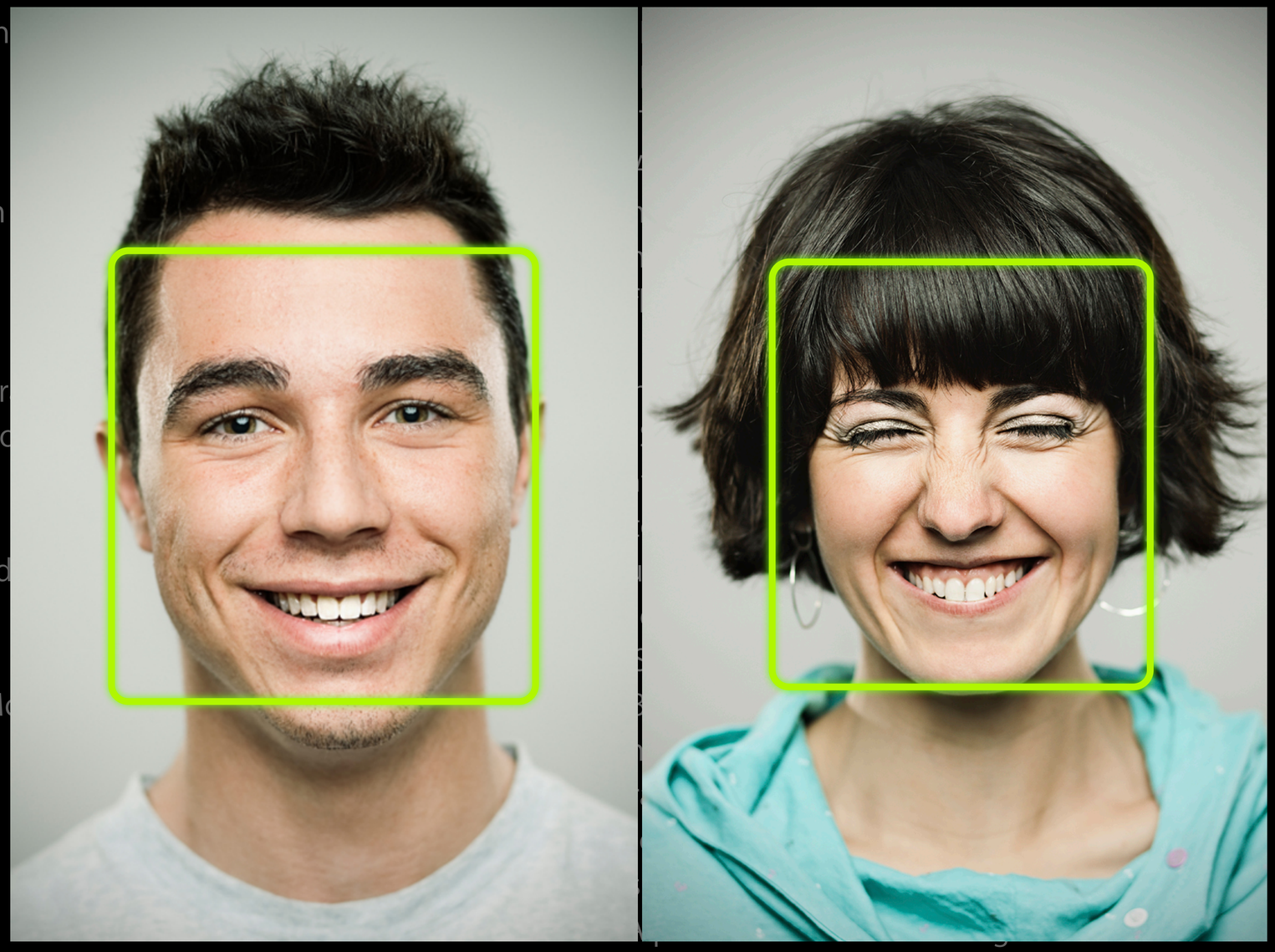
CIAdditionCompositin
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGener
CICircleSplashDistortio
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMod
CIColorControls
CIColorCube
CIColorDodgeBlendMo
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp

CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Detectors

CIAdditionCompositin
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGener
CICircleSplashDistortio
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMod
CIColorControls
CIColorCube
CIColorDodgeBlendMo
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp

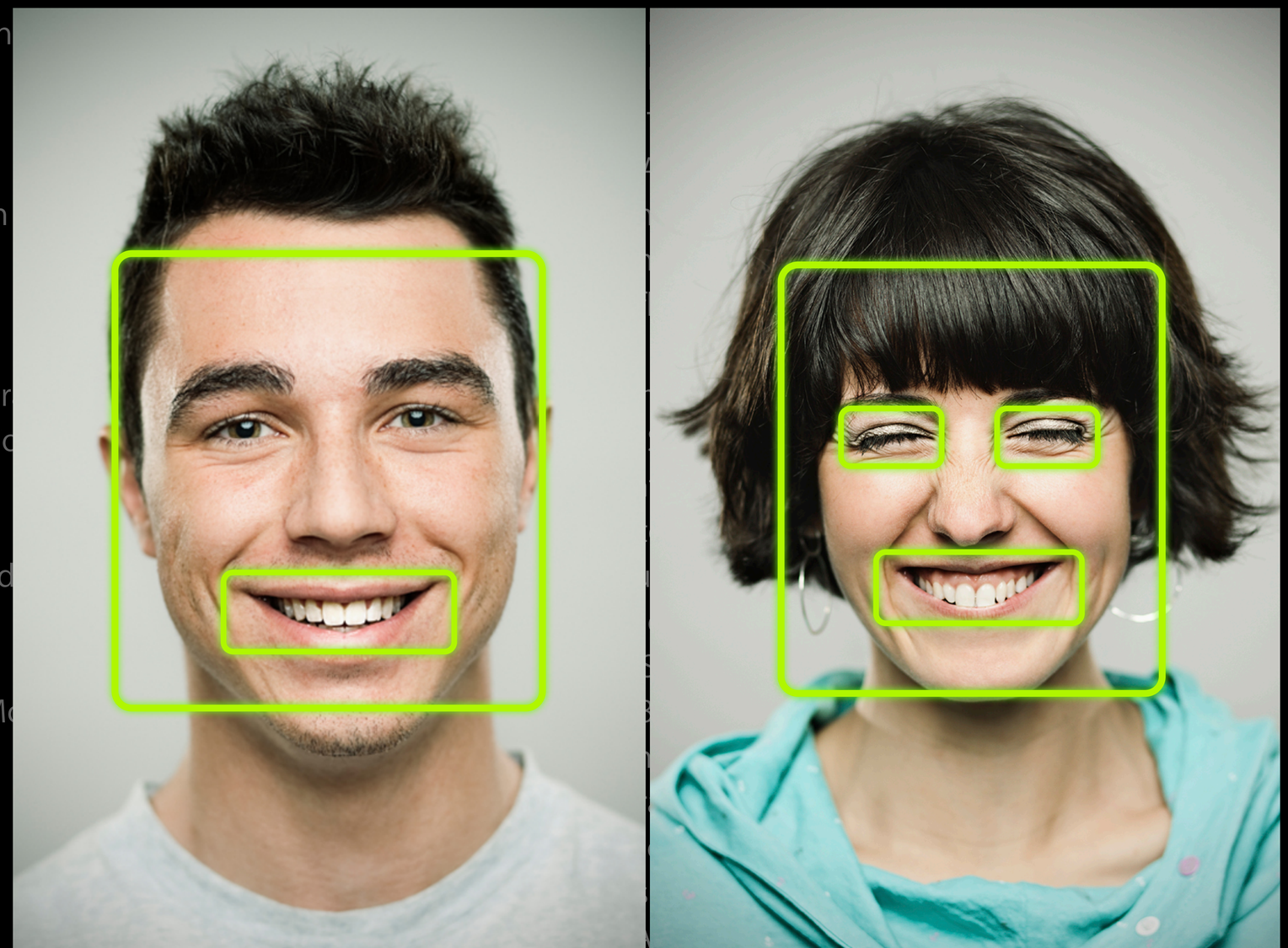


CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

Detectors

CIAdditionCompositin
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGener
CICircleSplashDistortic
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMod
CIColorControls
CIColorCube
CIColorDodgeBlendM
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp



CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

CIAdditionCompositing
CIAffineClamp
CIAffineTile
CIAffineTransform
CIBarsSwipeTransition
CIBlendWithMask
CIBloom
CIBumpDistortion
CICheckerboardGenerator
CICircleSplashDistortion
CICircularScreen
CIColorBlendMode
CIColorBurnBlendMode
CIColorControls
CIColorCube
CIColorDodgeBlendMode
CIColorInvert
CIColorMap
CIColorMatrix
CIColorMonochrome
CIColorClamp
CIColorCrossPolynomial
CIColorPolynomial
CIColorPosterize
CIConstantColorGenerator
CIConvolution3X3
CIConvolution5X5
CIConvolution9Horizontal
CIConvolution9Vertical
CICopyMachineTransition
CICrop
CIDarkenBlendMode
CIDifferenceBlendMode
CIDisintegrateWithMask
CIDissolveTransition
CIDotScreen
CIEightfoldReflectedTile
CIExclusionBlendMode
CIExposureAdjust
CIFaceDetector
CIFalseColor
CIFlashTransition
CIFourfoldReflectedTile
CIFourfoldRotatedTile
CIFourfoldTranslatedTile
CIGammaAdjust
CIGaussianBlur
CIGaussianGradient
CIGlideReflectedTile
CIGloom
CIHardLightBlendMode
CIHatchedScreen
CIHighlightShadowAdjust
CIHoleDistortion
CIHueAdjust
CIHueBlendMode
CILanczosScaleTransform
CILightenBlendMode
CILightTunnel
CILinearGradient
CILineScreen
CILuminosityBlendMode
CIMaskToAlpha
CIMaximumComponent
CIMaximumCompositing
CIMinimumComponent
CIMinimumCompositing
CIModTransition
CIMultiplyBlendMode
CIMultiplyCompositing
CIOverlayBlendMode
CIPerspectiveTile
CIPerspectiveTransform
CIPinchDistortion
CIPixellate
CIRadialGradient
CIRandomGenerator
CISaturationBlendMode
CIScreenBlendMode
CISepiaTone
CISharpenLuminance
CISixfoldReflectedTile
CISixfoldRotatedTile
CISoftLightBlendMode
CISourceAtopCompositing
CILinearToSRGBToneCurve
CISRGBToneCurveToLinear
CISourceInCompositing
CISourceOutCompositing
CISourceOverCompositing
CIStarShineGenerator
CIStraightenFilter
CIStripesGenerator
CISwipeTransition
CITemperatureAndTint
CIToneCurve
CITriangleKaleidoscope
CITwelvefoldReflectedTile
CITwirlDistortion
CIUnsharpMask
CIVibrance
CIVignette
CIVortexDistortion
CIWhitePointAdjust
CIQRCodeGenerator

Lots of Useful Filters

- How we choose built-in filters
  - Must be broadly useful to clients
    - For example CIConvolution5X5 can be used for blur, sharpen, edge detection, and more
  - Must be performant on the target platform

Chaining Multiple Filters

Chaining Multiple Filters

- Apply first filter to input image

Chaining Multiple Filters

- Apply first filter to input image

```
output = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
          kCIInputImageKey, image,
          @"inputIntensity", @0.8,
          nil].outputImage;
```

Chaining Multiple Filters

- Apply first filter to input image

```
output = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
          kCIInputImageKey, image,
          @"inputIntensity", @0.8,
          nil].outputImage;
```

- Apply next filter

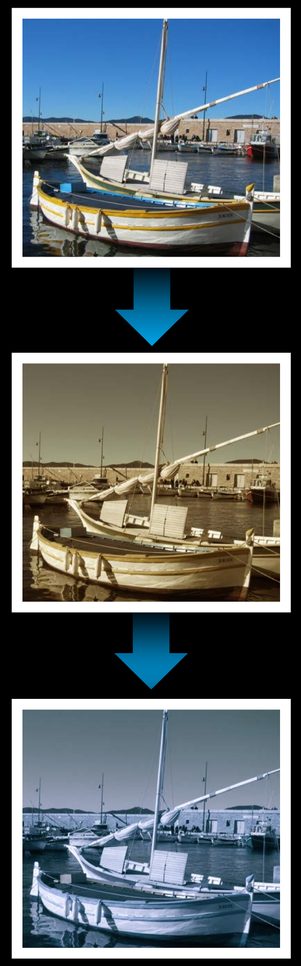
Chaining Multiple Filters

- Apply first filter to input image

```
output = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
          kCIInputImageKey, image,
          @"inputIntensity", @0.8,
          nil].outputImage;
```

- Apply next filter

```
output = [CIFilter filterWithName:@"CIHueAdjust" keysAndValues:
          kCIInputImageKey, output,
          @"inputAngle", @0.8,
          nil].outputImage;
```



Chaining Multiple Filters

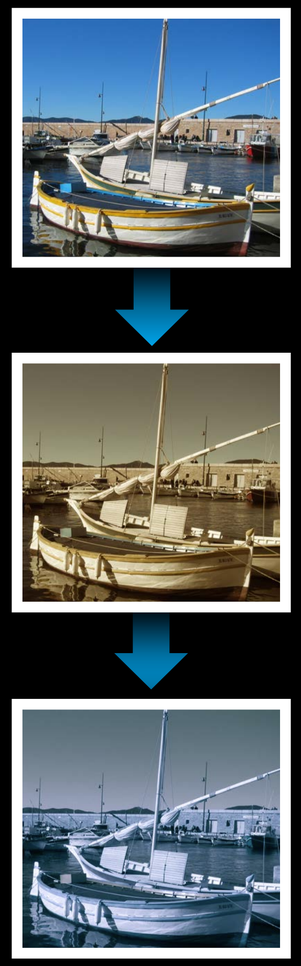
- Apply first filter to input image

```
output = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
          kCIInputImageKey, image,
          @"inputIntensity", @0.8,
          nil].outputImage;
```

- Apply next filter

```
output = [CIFilter filterWithName:@"CIHueAdjust" keysAndValues:
          kCIInputImageKey, output,
          @"inputAngle", @0.8,
          nil].outputImage;
```

- No pixel processing is performed while building the chain



Chaining Multiple Filters

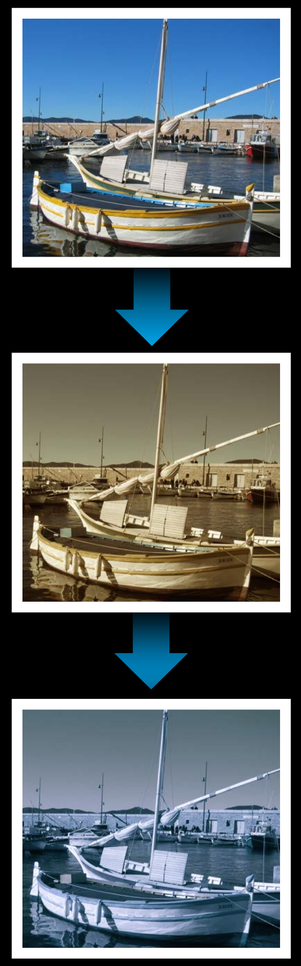
- Apply first filter to input image

```
output = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
          kCIInputImageKey, image,
          @"inputIntensity", @0.8,
          nil].outputImage;
```

- Apply next filter

```
output = [CIFilter filterWithName:@"CIHueAdjust" keysAndValues:
          kCIInputImageKey, output,
          @"inputAngle", @0.8,
          nil].outputImage;
```

- No pixel processing is performed while building the chain
  - Work is deferred until render is requested



Building Custom Filters from Built-in Filters

- Core Image on iOS 100+ filters
  - Custom filters are not supported
  - But you can creatively combine filters to achieve the effect you need
  - Core Image will efficiently combine the filter graph
- The new filters in iOS 7 are quite useful for this

Building Custom Filters from Built-In Filters

A filter graph can be wrapped as a subclass of CIFilter

- Your CIFilter subclass will need
  - Declare `@properties` for its input parameters such as `inputImage`
  - Override `-(void) setDefaults`
  - Override `-(CIImage*) outputImage`
  - Core Image implements some of its built-in CIFilters using this technique

Building Custom Filters from Built-In Filters

An example from CI's source

```
@interface CIColorInvert: CIFilter {
    CIImage *inputImage;
}
@property (retain, nonatomic) CIImage *inputImage;
@end

@implementation CIColorInvert
@synthesize inputImage;
- (CIImage *)outputImage {
    return [CIFilter filterWithName:@"CIColorMatrix" keysAndValues:
        kCIInputImageKey, inputImage,
         @"inputRVector", [CIVector vectorWithX:-1 Y:0 Z:0],
         @"inputGVector", [CIVector vectorWithX:0 Y:-1 Z:0],
         @"inputBVector", [CIVector vectorWithX:0 Y:0 Z:-1],
         @"inputBiasVector", [CIVector vectorWithX:1 Y:1 Z:1],
         nil].outputImage;
}
```

Building Custom Filters from Built-in Filters

Sobel edge detector

- A Sobel Edge Detector is just a special 3X3 convolution

Horizontal Sobel

1	0	-1
2	0	-2
1	0	-1

Vertical Sobel

-1	-2	-1
0	0	0
1	2	1

- Want to add a bias after the convolution so that:
  - Areas with no edges show up as grey
  - Areas with edges show up black and white
  - Note that using a bias will make the image infinite

Demo

Sobel in Core Image Fun House

Building Custom Filters from Built-in Filters

Using CIFilters in Sprite Kit

- The Sprite Kit APIs allow one CIFilter to be associated with an object

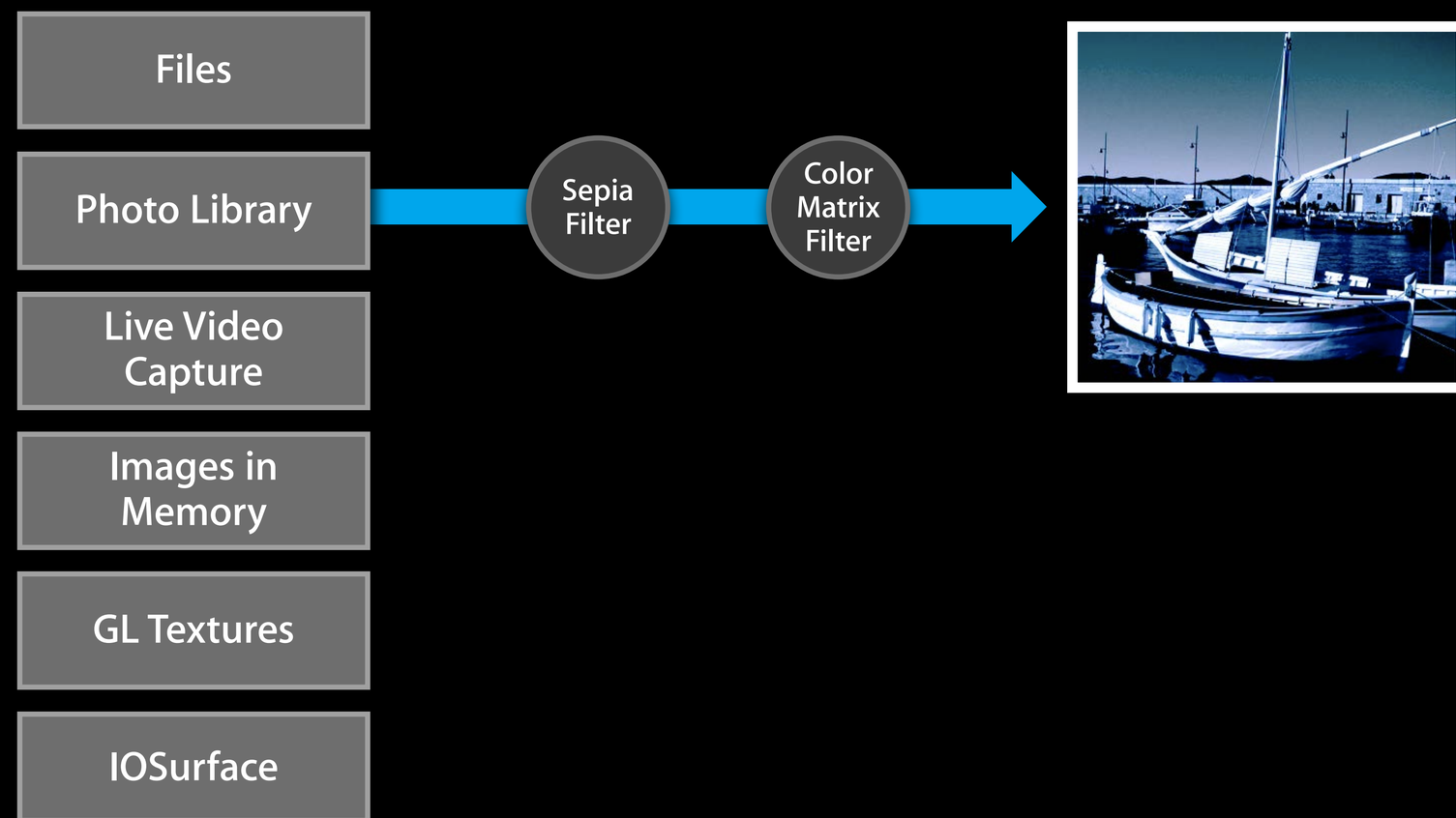
```
SKEffectNode setFilter:
SKTexture textureWithCIFilter:
SKTransition transitionWithFilter:
```

- If you want an object to have a complex filter graph your subclass must:
  - Have an inputImage parameter
  - Have an inputTime parameter for transitions
  - Other inputs allowed but must be setup when the filter is passed to Sprite Kit

Providing Input Images

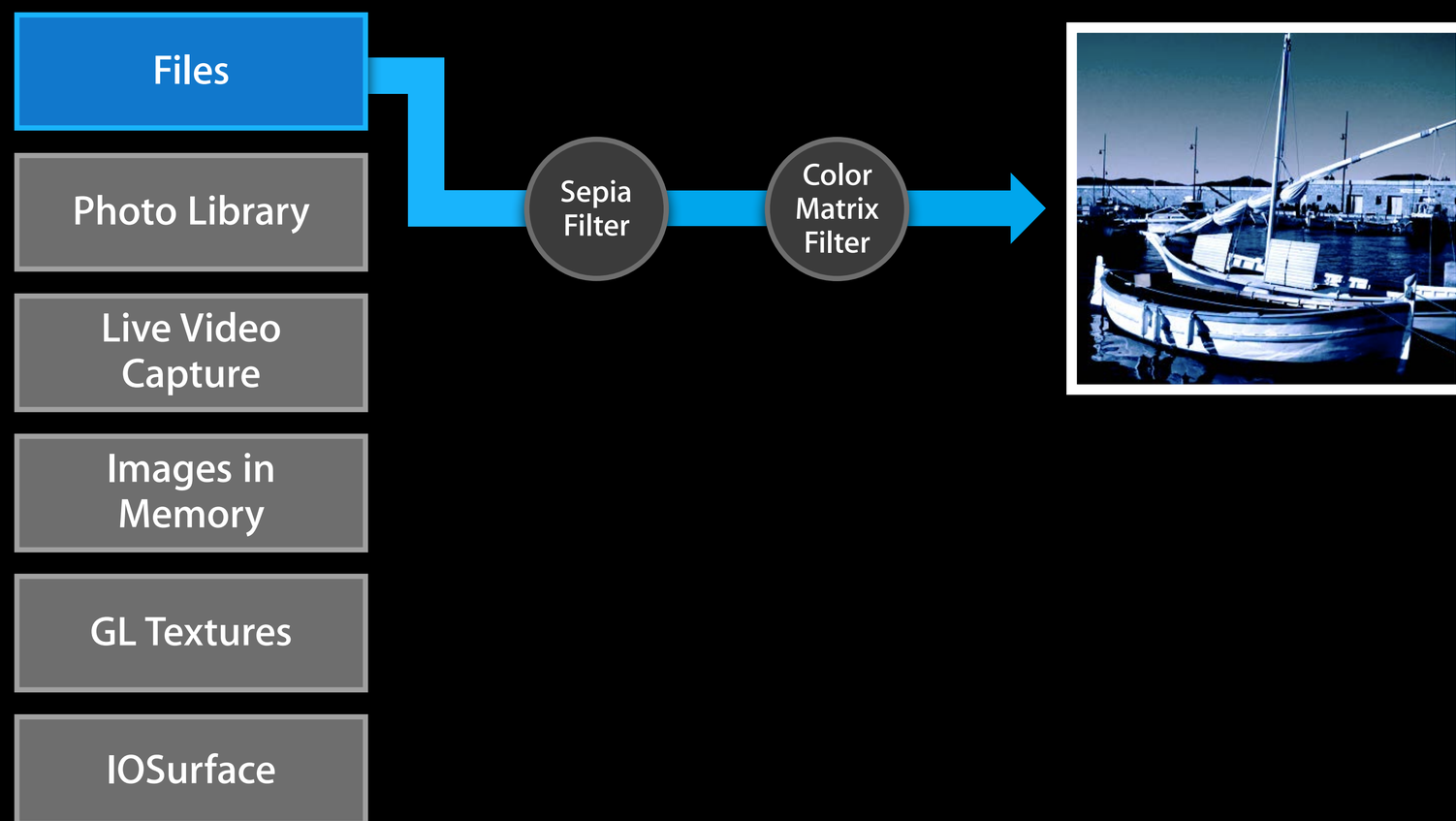
Providing Input Images

Flexible inputs



Providing Input Images

Flexible inputs: Files



Providing Input Images

Flexible inputs: Files

Files

Photo

Live Video Capture

Images in Memory

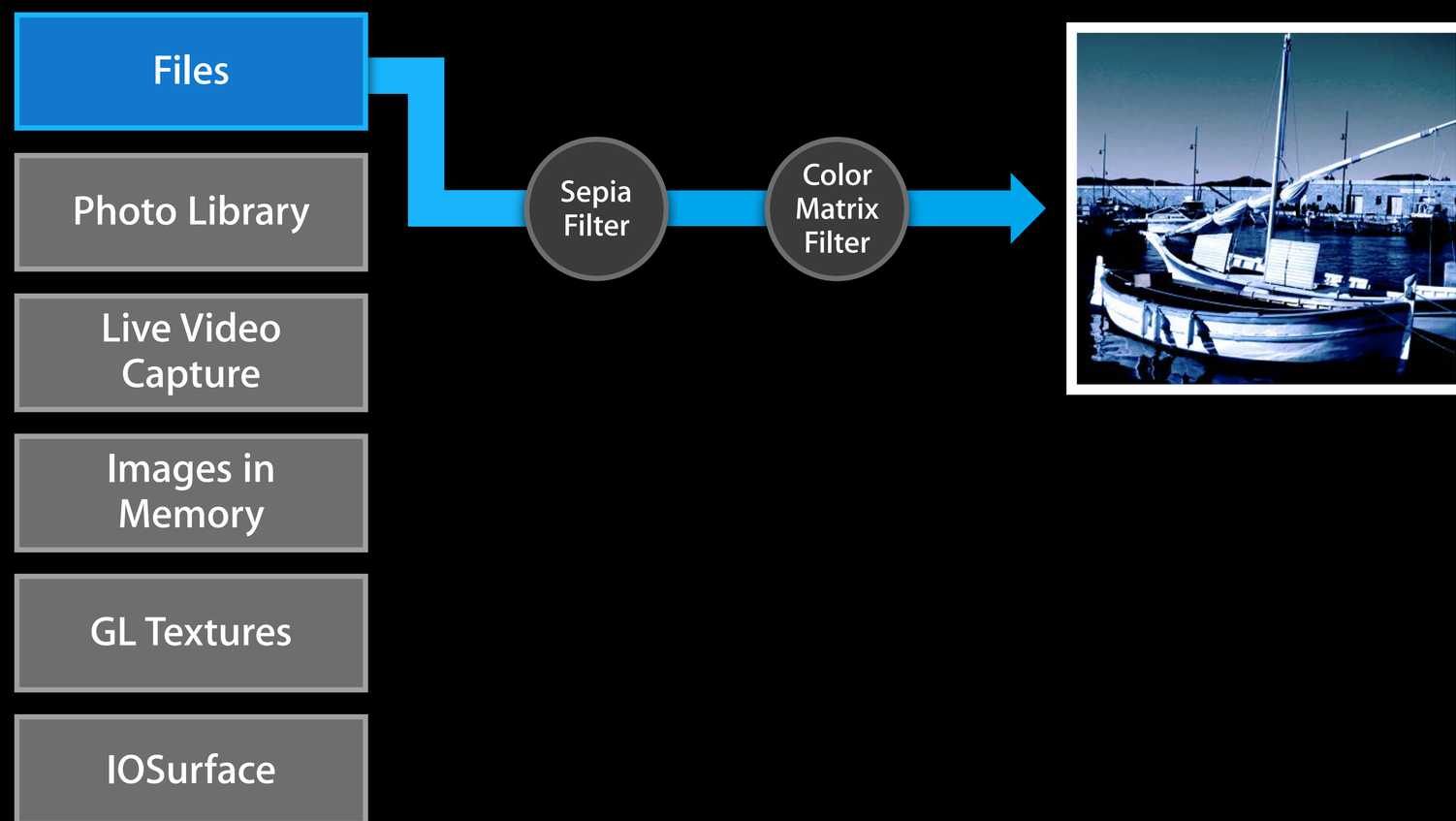
GL Textures

IOSurface

```
// Create a CIImage object from a URL
ciimage = [CIImage imageWithURL:url];
```

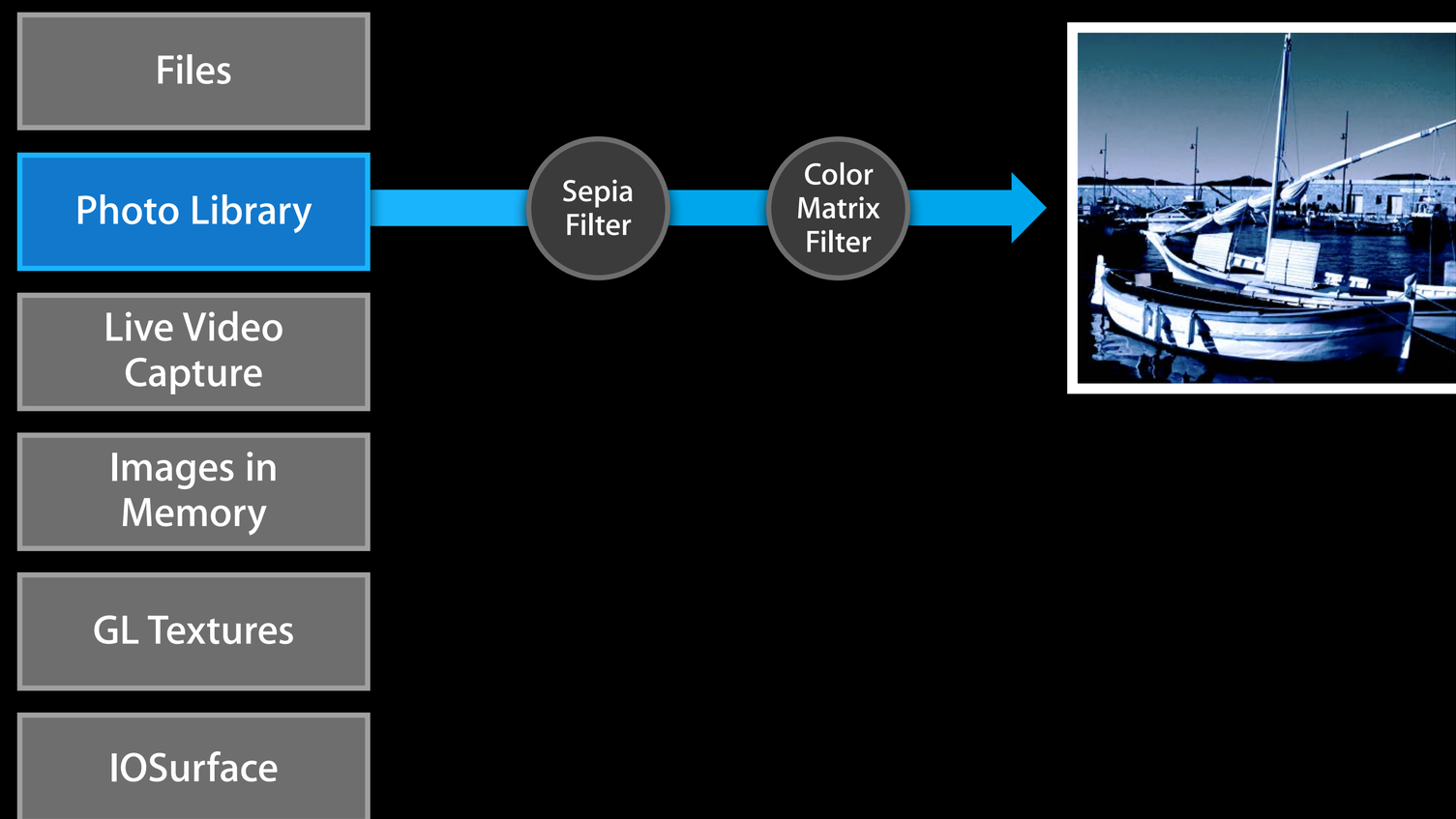
Providing Input Images

Flexible inputs: Files



Providing Input Images

Flexible inputs: Photo Library



Providing Input Images

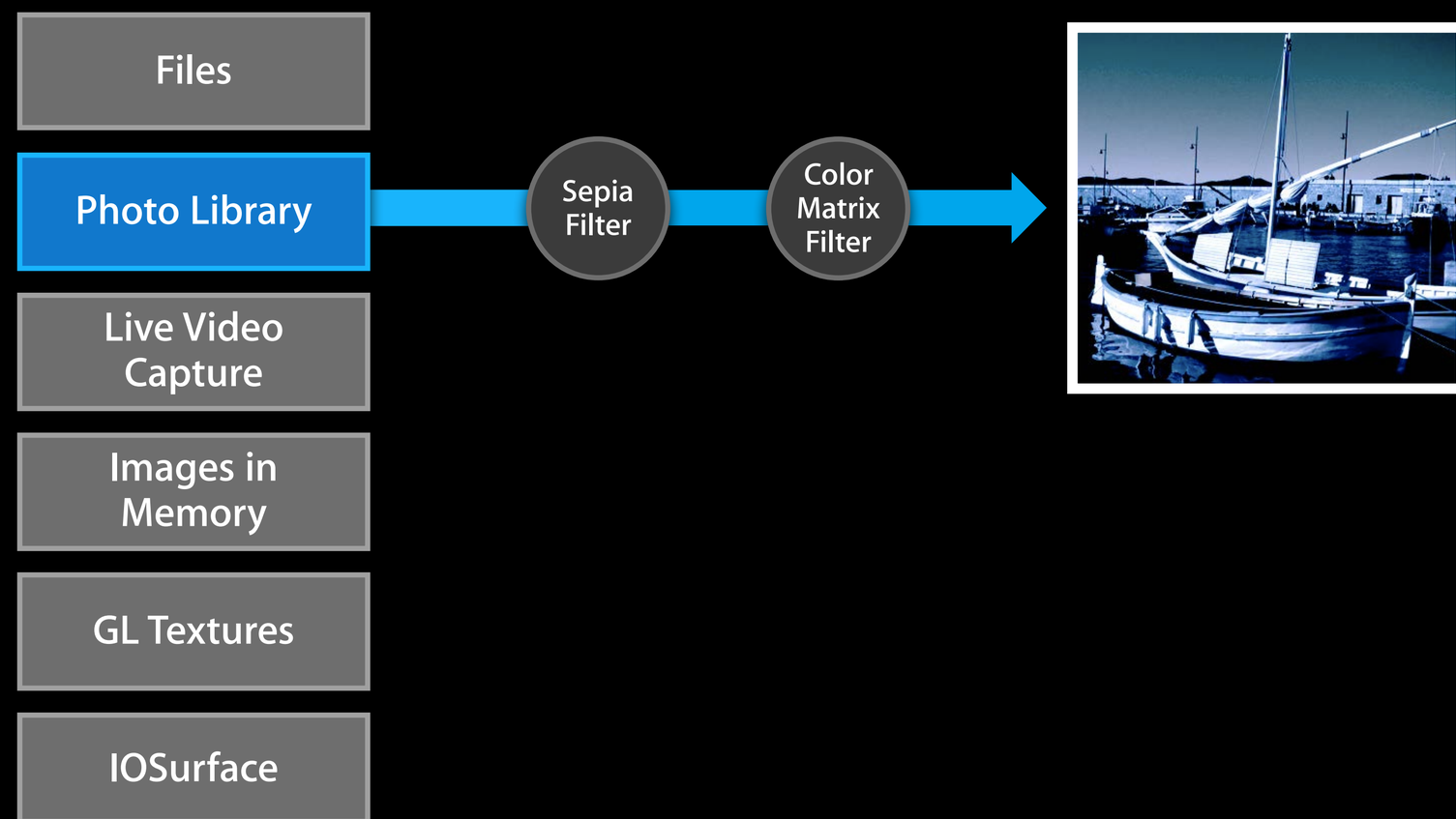
Flexible inputs: Photo Library

```
// Create a CIImage object from the Photo Library
ALAssetsLibrary *library = [ALAssetsLibrary new];
[library assetForURL:assetURL
   resultBlock:^(ALAsset *__strong asset) {
       ALAssetRepresentation *rep = [asset defaultRepresentation];
       CGImageRef cgimage = [rep fullScreenImage];
       ciimage = [CIImage imageWithCGImage:cgimage];
    }
    failureBlock:nil];
```

IOSurface

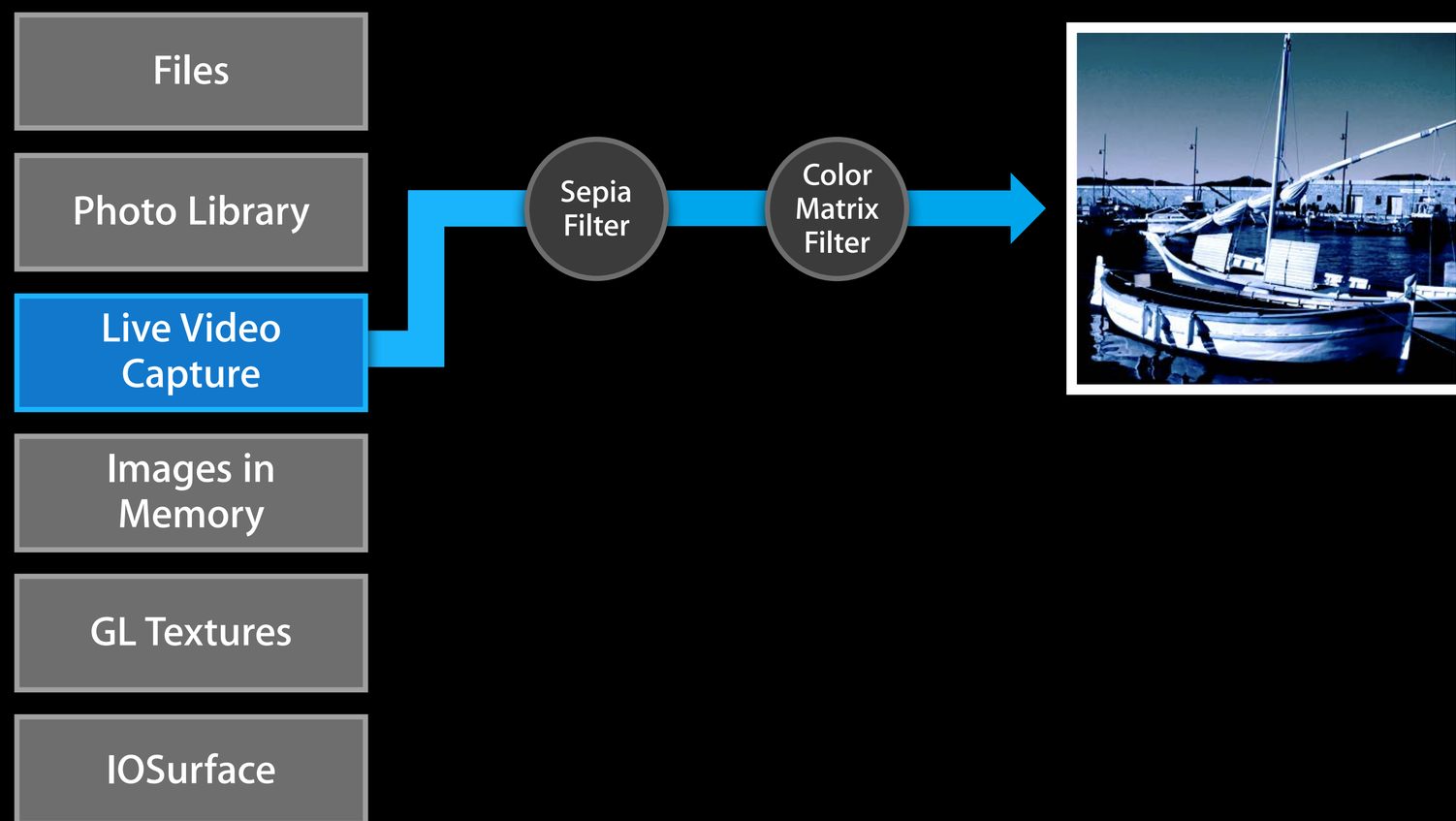
Providing Input Images

Flexible inputs: Photo Library



Providing Input Images

Flexible inputs: Live Video



Providing Input Images

Flexible inputs: Live Video

```
// Create a CIImage object from AVCapture
- (void)captureOutput:(AVCaptureOutput *)output
didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
     fromConnection:(AVCaptureConnection *)connection
{
  CVImageBufferRef cvimg = CMSampleBufferGetImageBuffer(sampleBuf);
  CIImage *ciimage =
        [CIImage imageWithCVPixelBuffer:(CVPixelBufferRef)cvimg];
  ...
```

IOSurface

Initializing a CIImage's Metadata

- The `[image properties]` method gets metadata properties of an image
  - Returns dictionary with same key/values as `CGImageSourceCopyPropertiesAtIndex`
  - One notable key is `kCGImagePropertyOrientation`
- Properties are automatic if you use `imageWithURL:` or `imageWithData:`
  - Otherwise properties can be specified using `kCIImageProperties` option

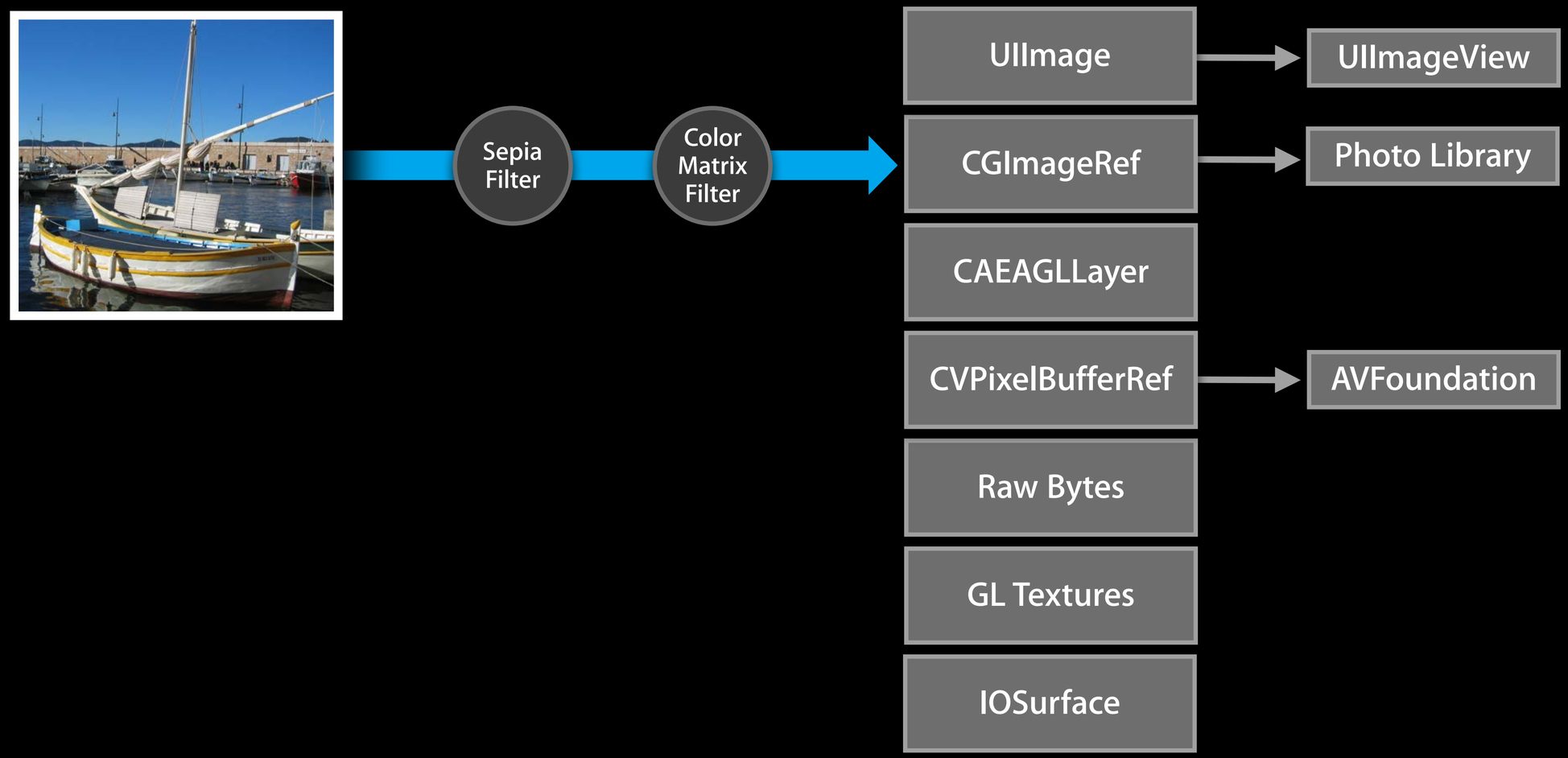
YCbCr-Based CIImages

- A CIImage can based on a bi-planar YCbCr 420 data
  - On OS X use an IOSurface
  - On iOS use an CVPixelBuffer
- Core Image will take care to:
  - Combine full-res Y and subsampled CbCr planes into one
  - Apply correct 3x4 matrix to convert to RGB working space
    - Check out Poynton's "Digital Video and HD Algorithms and Interfaces"
- On OS X, you may want to tell CI to use a rec709 working space
  - OS X uses linear Generic RGB by default

Rendering Core Image Output

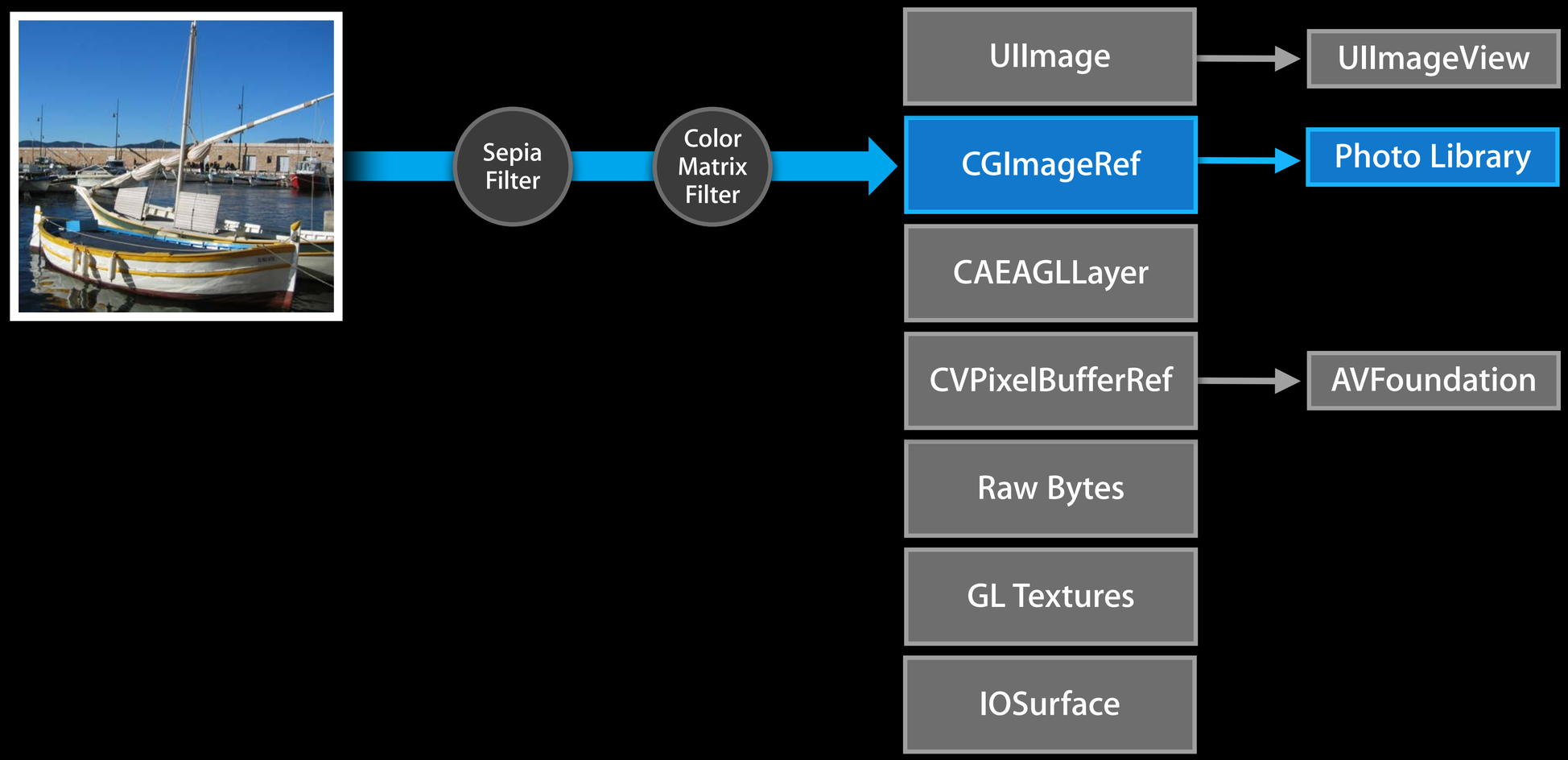
Rendering a CIImage

Flexible outputs



Rendering a CIImage

Flexible outputs: Photo Library



Rendering a CIImage

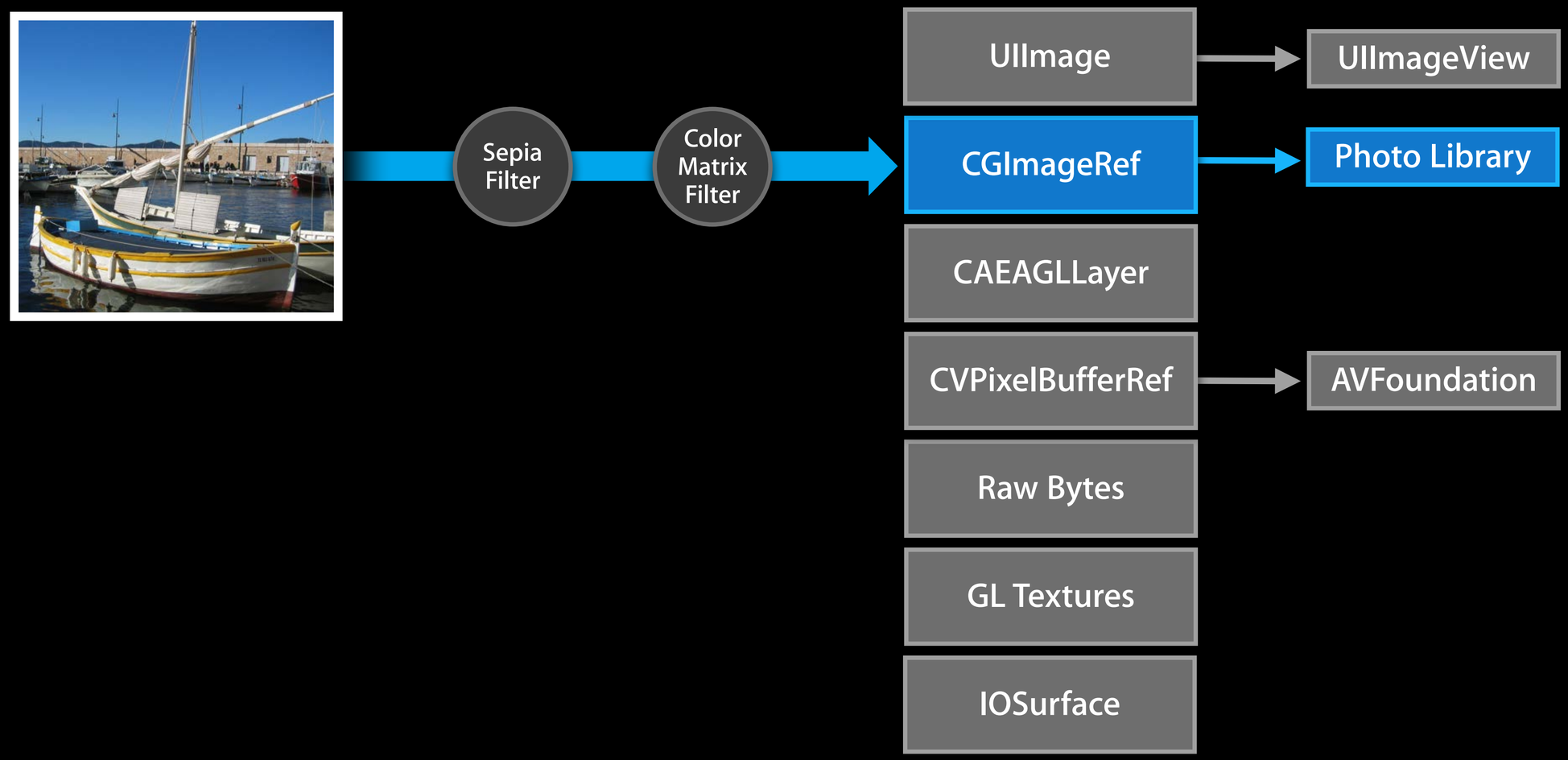
Flexible outputs: Photo Library

```
// Create a CPU context so we can save in background
NSDictionary *opts = @{ kCIContextUseSoftwareRenderer : @YES };
CIContext *ctx = [CIContext contextWithOptions:opts];

// Create CGImage and add to photo library
CGImageRef cgimg = [ctx createCGImage:ciimg fromRect:ciimg.extent];
ALAssetsLibrary *library = [ALAssetsLibrary new];
[library writeImageToSavedPhotosAlbum:cgimg
     metadata:[ciimage properties]
     completionBlock:^(NSURL *url NSError *err) {
               CGImageRelease(cgimg);
      }];
```

IOSurface

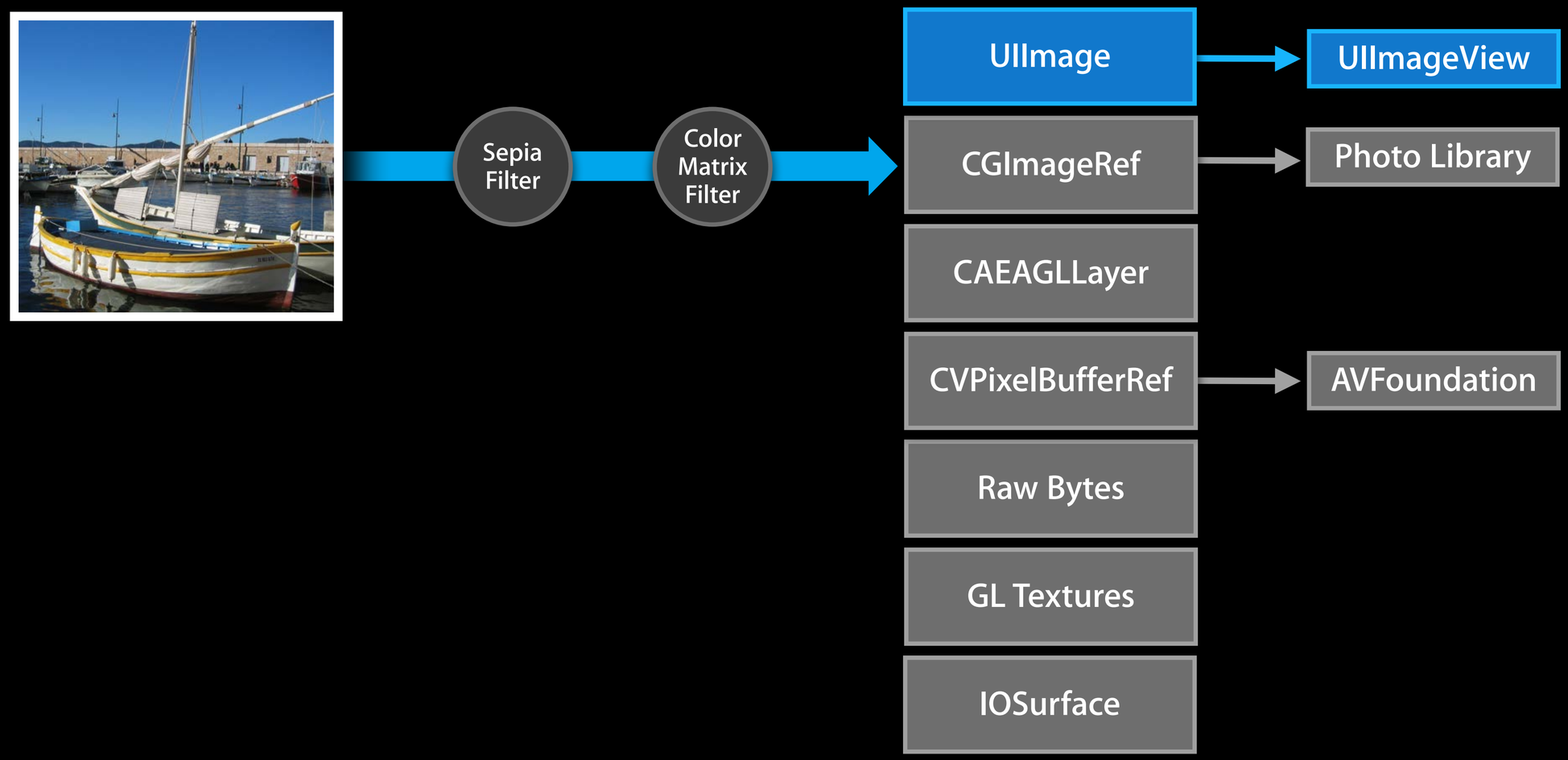
Rendering a CIImage

Flexible outputs: Photo Library



Rendering a CIImage

Flexible outputs: UIImageView



Rendering a CIImage

Flexible outputs: UIImageView

```
// Create a UIImage using the filter output
UIImage *image = [UIImage imageWithCIImage:filter.outputImage];

// Use the UIImage in an UIImageView
imageView.image = uiimage;
```

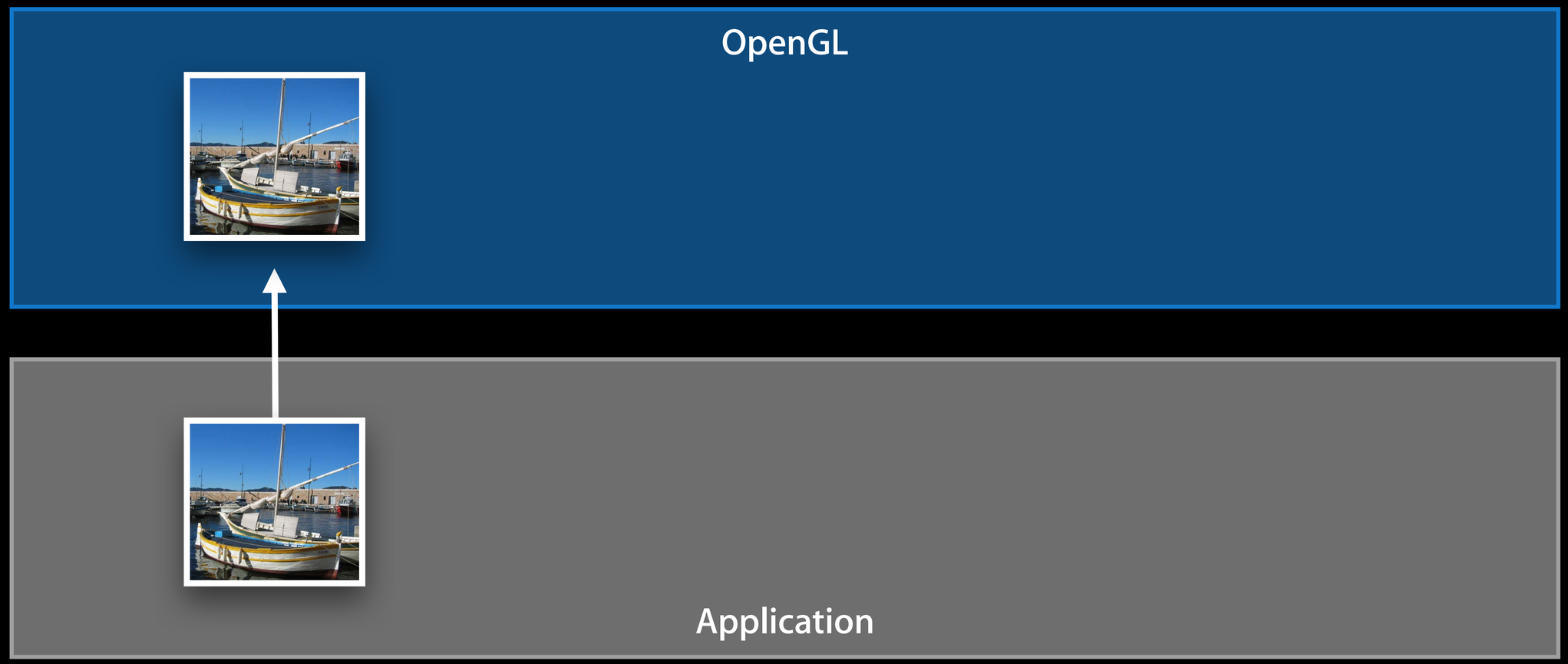
Raw Bytes

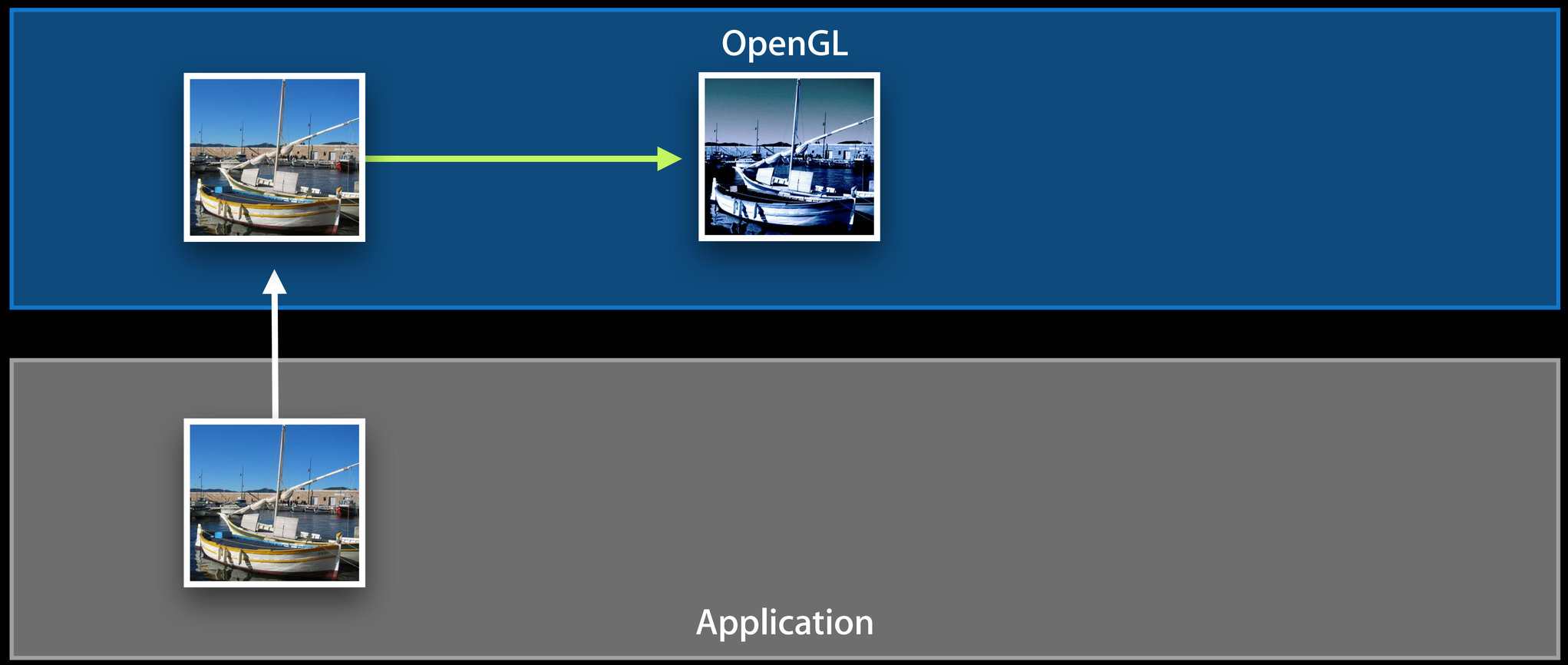
GL Textures

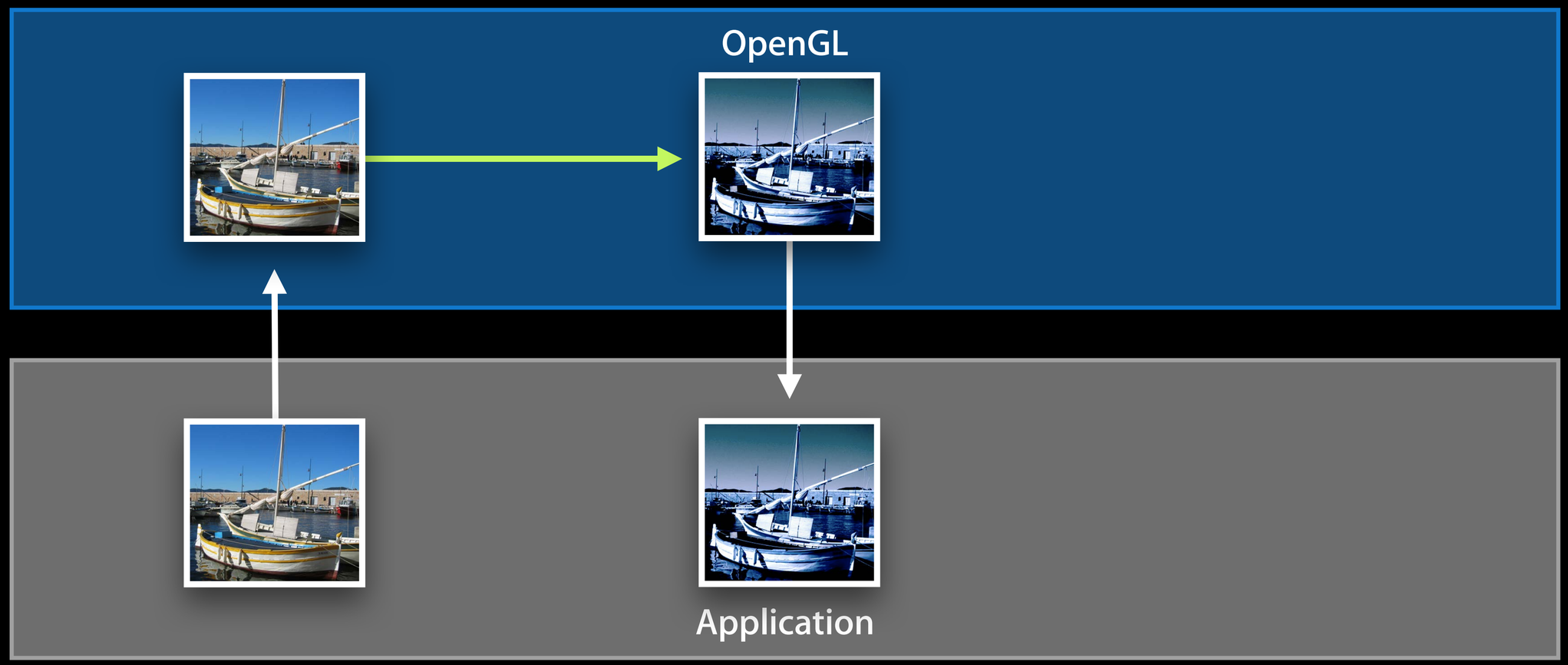
IOSurface

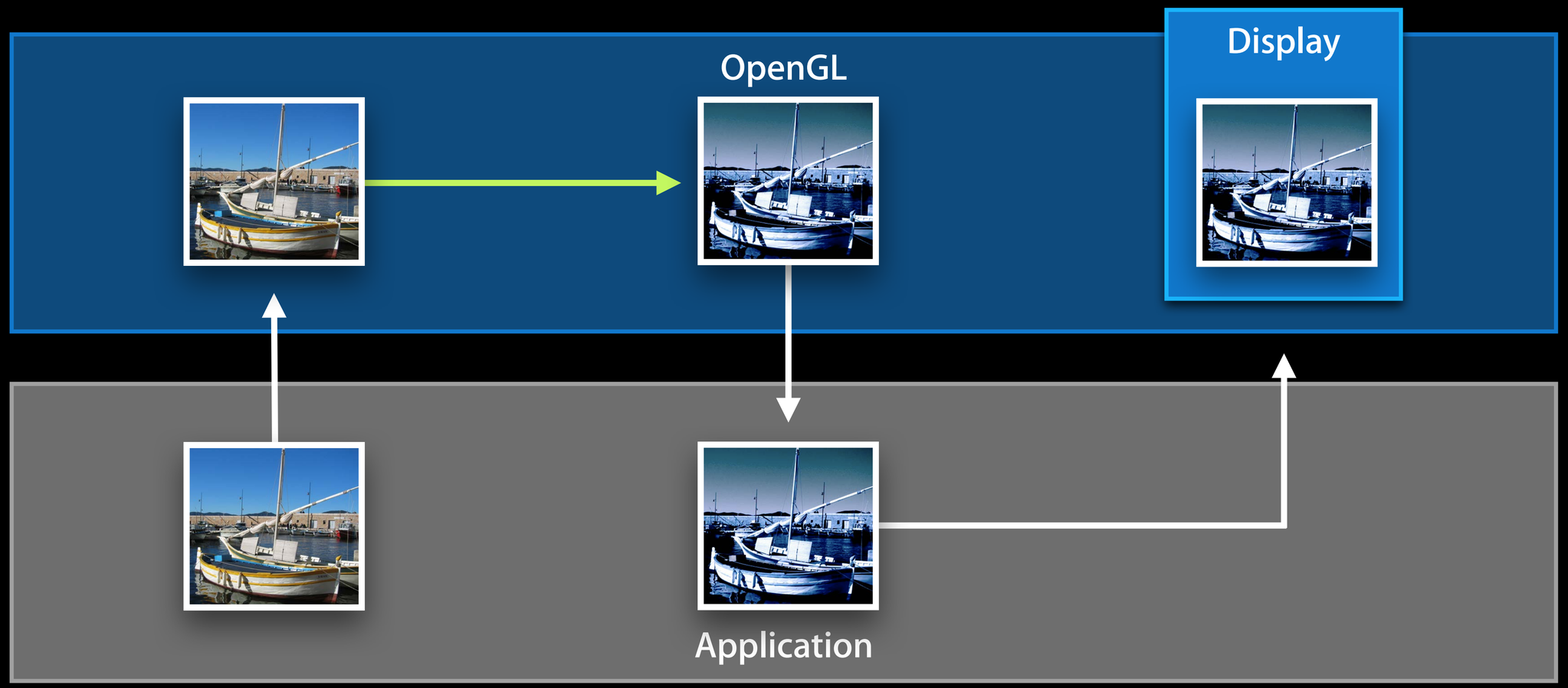
OpenGL

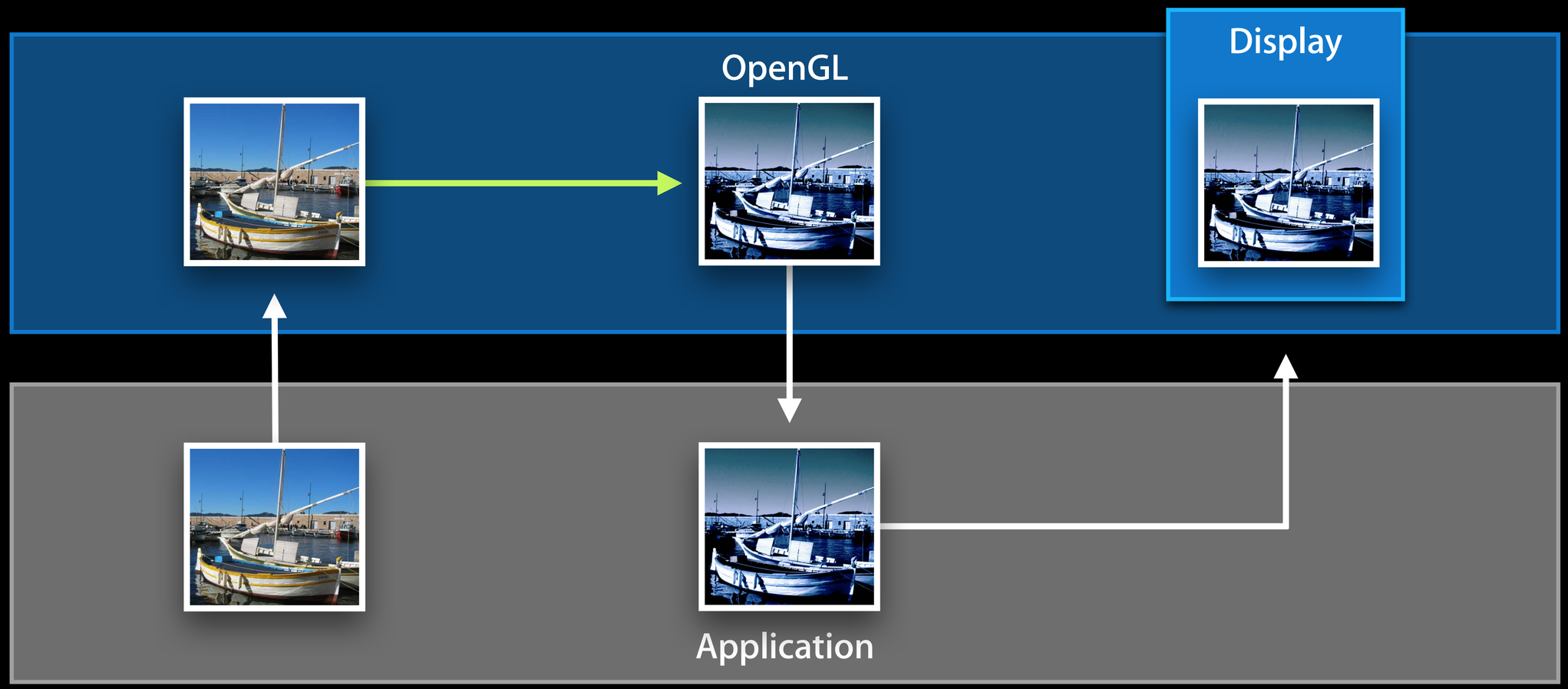
Application

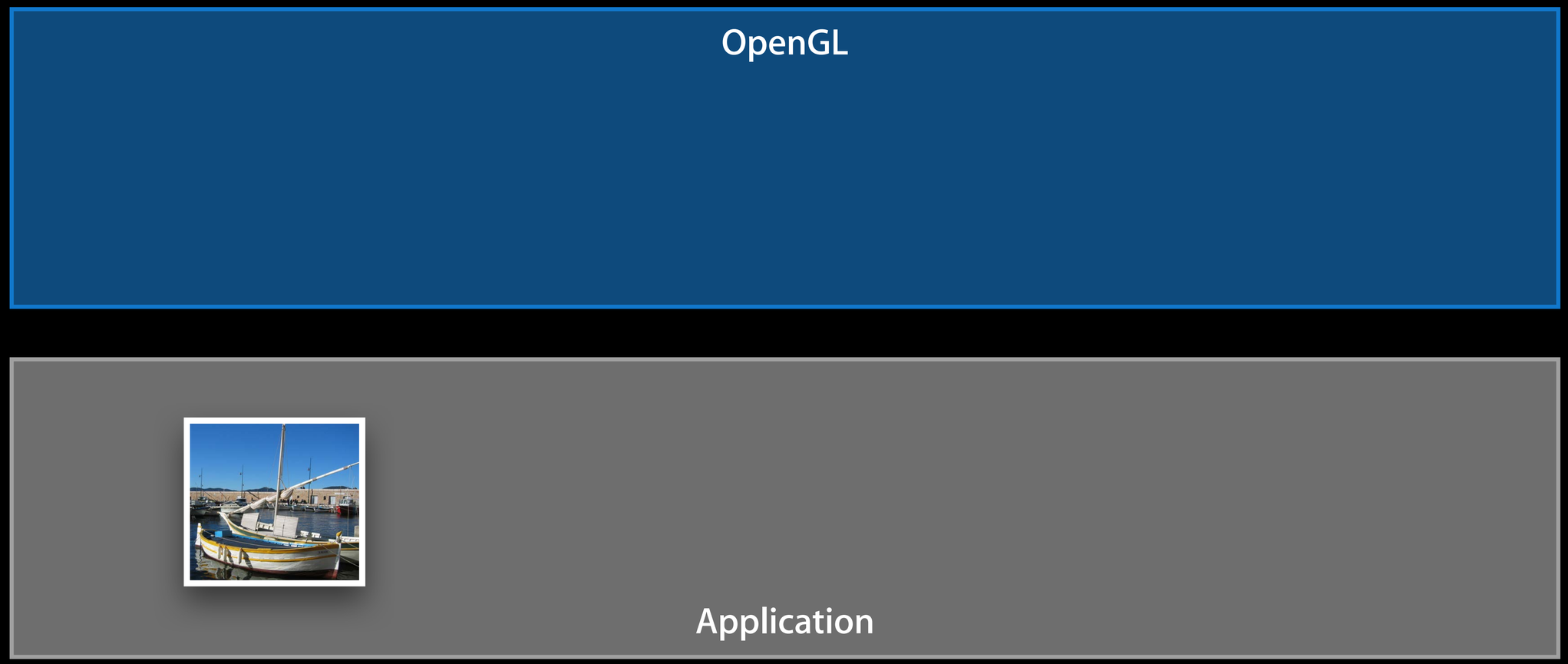

OpenGL
Application

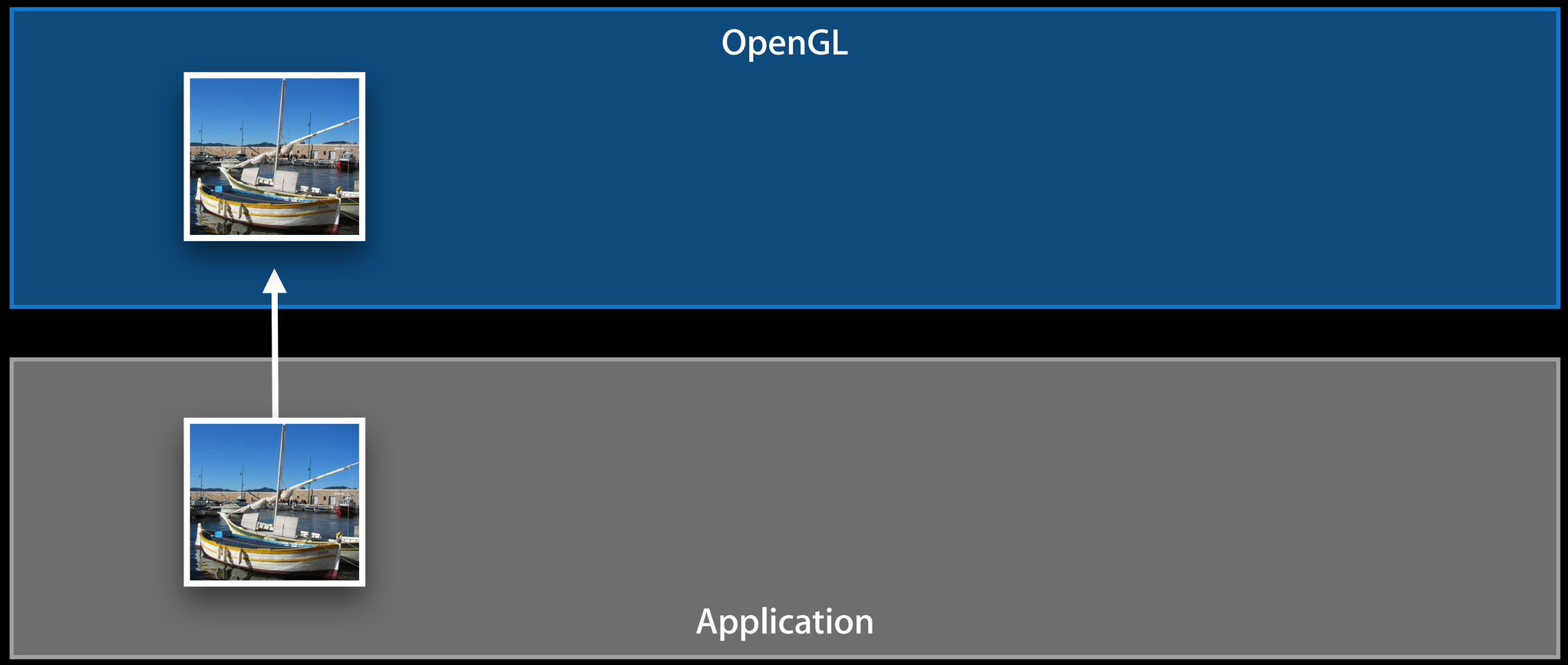

OpenGL
Application

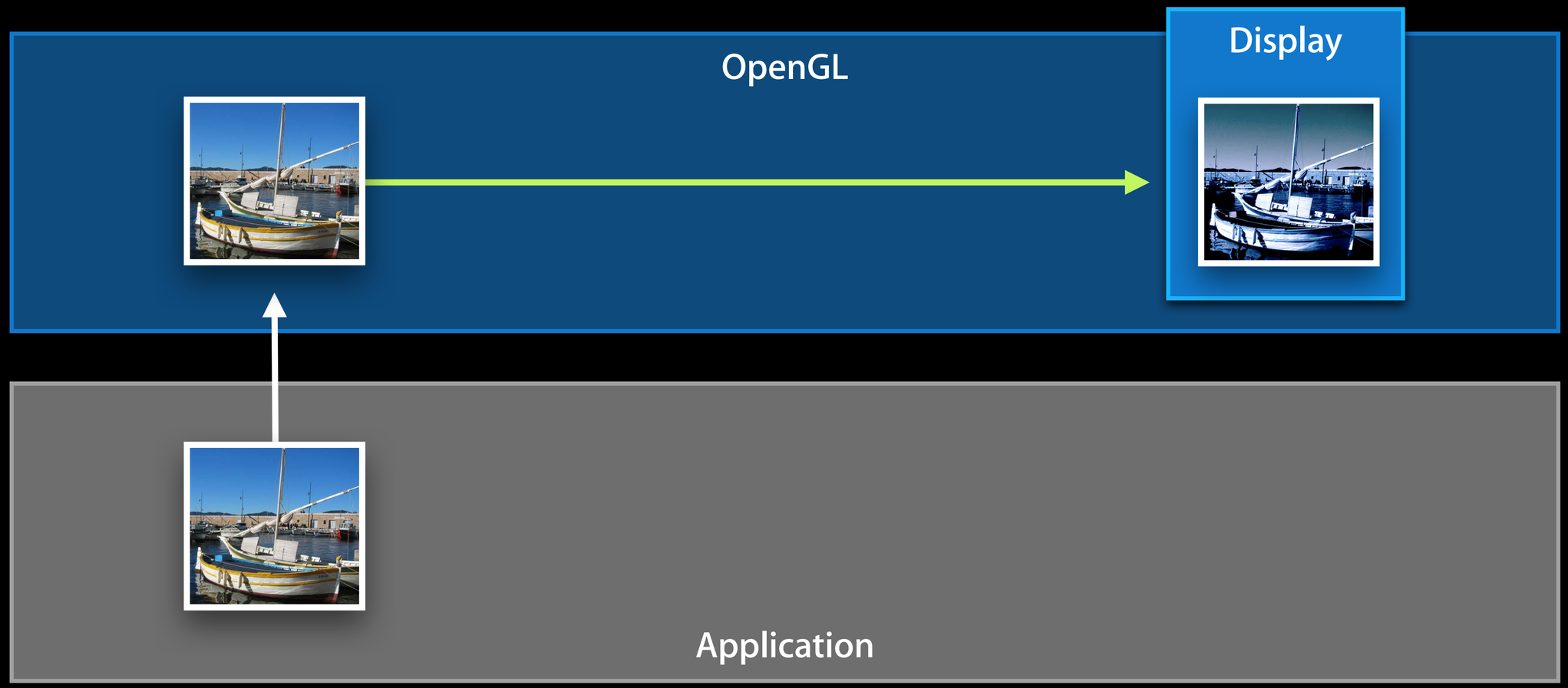

OpenGL
Application


OpenGL
Display
Application


OpenGL
Display
Application

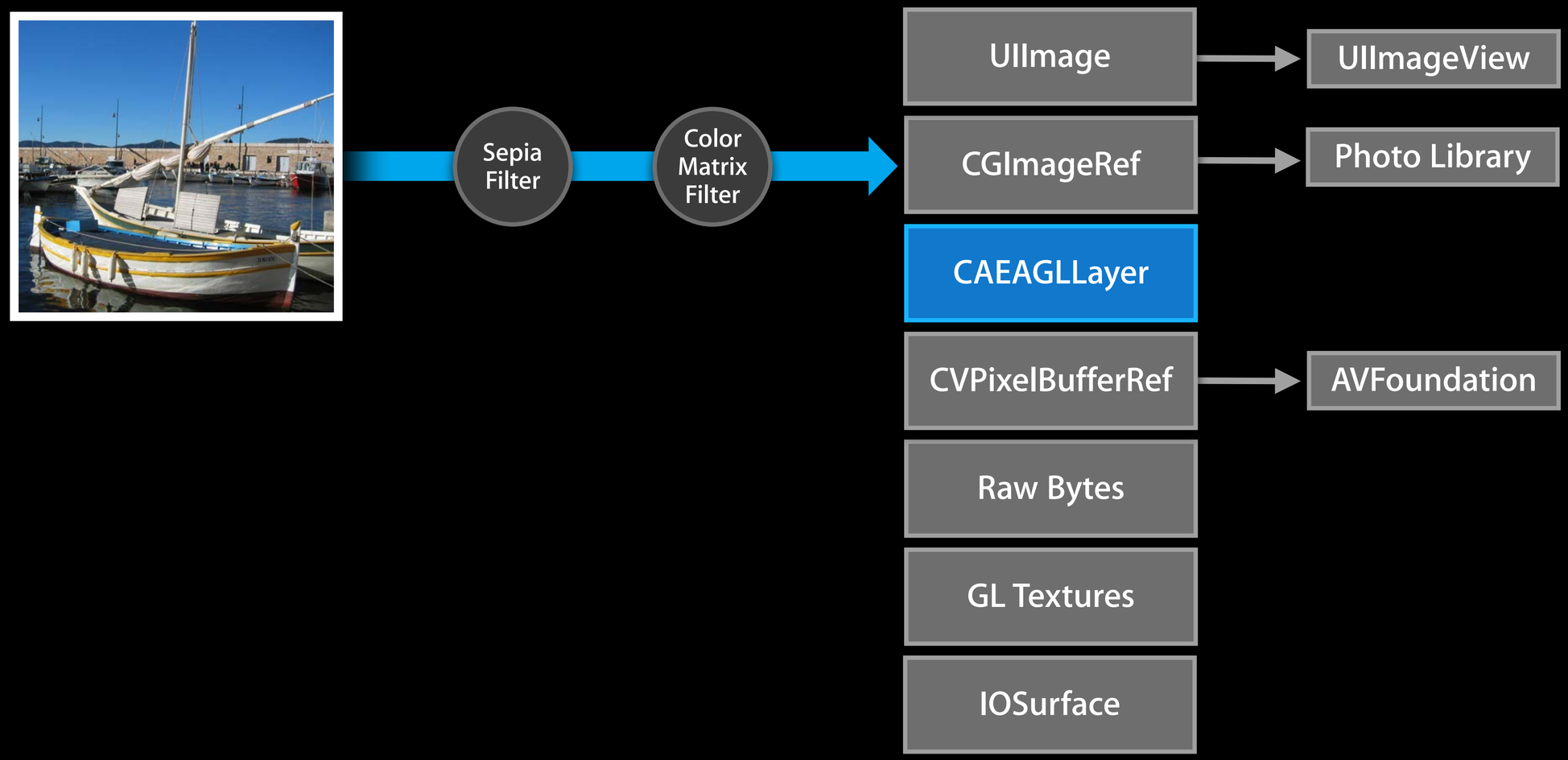

OpenGL
Application


OpenGL
Application


OpenGL
Display
Application

Rendering a CIImage

Flexible outputs: CAEAGLLayer



Rendering a CIImage

Flexible outputs: CAEAGLLayer

```
// Create an AEGL-backed CIContext
EAGLContext *eaglctx = [[EAGLContext alloc]
                 initWithAPI:kEAGLRenderingAPIOpenGLES2];

cictx = [CIContext contextWithEAGLContext:eaglctx];
```

Raw Bytes

GL Textures

IOSurface

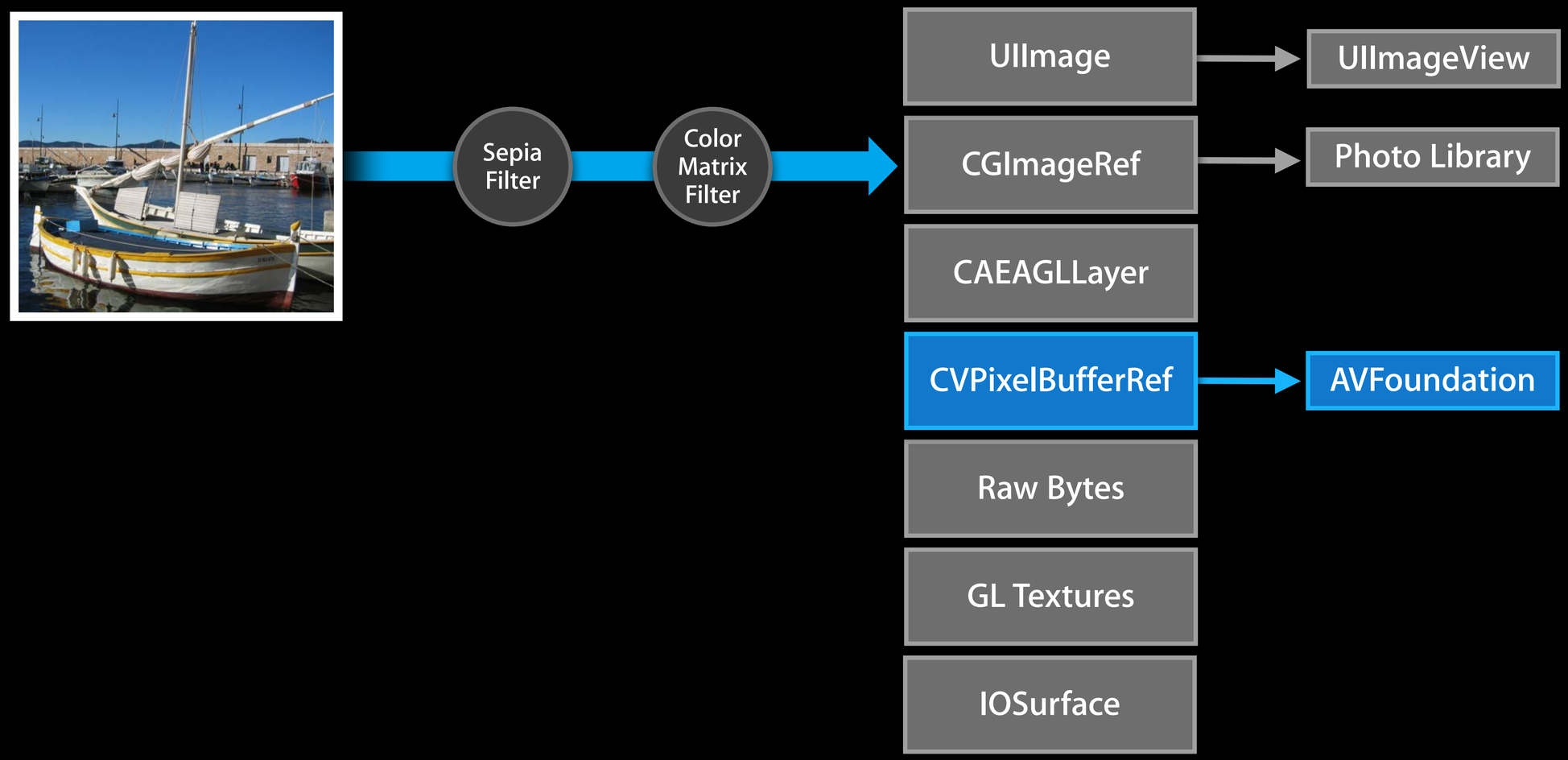
Rendering a CIImage

Flexible outputs: CAEAGLLayer

```
- (void) updateScreen
{
  // setup filter chain
  CIImage *ciimg = [model imageToRender];
  // setup GL's blend mode state
  glEnable(GL_BLEND); glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA);

  [cictx drawImage:ciimg atPoint:CGPointZero fromRect:ciimg.extent];

  glBindRenderbuffer(GL_RENDERBUFFER, render_buffer);
  [eaglContext presentRenderbuffer:GL_RENDERBUFFER];
}
```

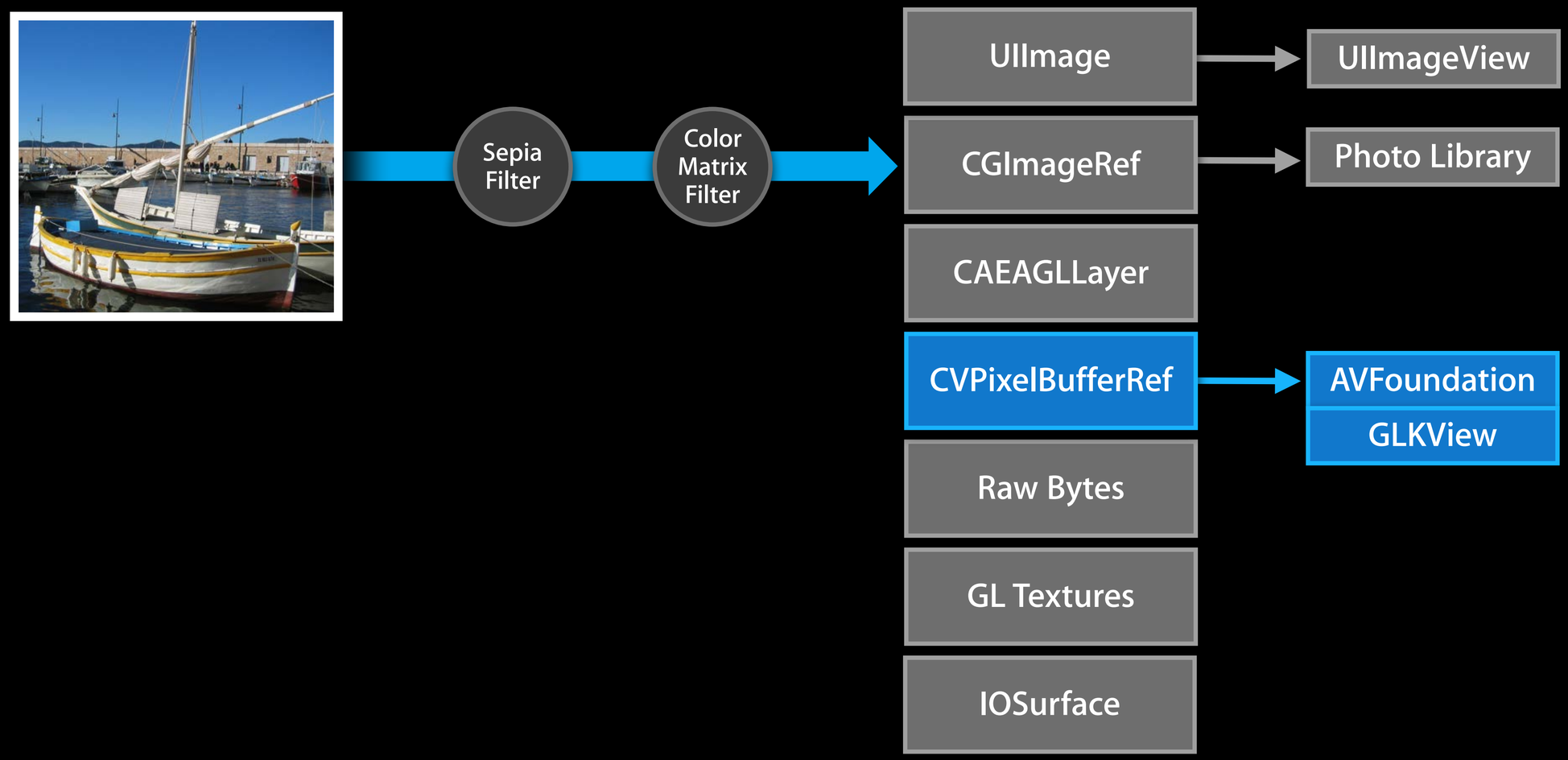
Rendering a CIImage

Flexible outputs: AVFoundation



Rendering a CIImage

Flexible outputs: AVFoundation



Demo

Using Core Image Fun House to filter and save live video

Tips and Best Practices

- CIImage and CIFilter objects are autoreleased
  - Use autorelease pools to reduce memory pressure
- Don’t create a CIContext every time you render
- Core Animation and Core Image both can use the GPU
  - Avoid CA animations while rendering CIImages with a GPU context

Tips and Best Practices

- CPU and GPU CIContexts have limits on image sizes
  - Check the context limits by using:

```
-(CGSize) inputImageMaximumSize;
-(CGSize) outputImageMaximumSize;
```

- Use smaller images when possible
  - Performance scales with the number of output pixels
  - You can use Core Graphics or ImageIO APIs to crop or downsample

```
CGImageCreateWithImageInRect
CGImageSourceCreateThumbnailAtIndex
```

  - ALAssetLibrary can return a screen-sized image

```
[[asset defaultRepresentation] fullScreenImage]
```

Bridging Core Image and OpenCL

Bridging Core Image and OpenCL

- Core Image on OS X Mavericks
  - Uses OpenCL to process images on GPU
- Improved performance due to
  - Advanced OpenCL compiler
  - Reduced state management
- No change to application is needed to get improved performance
- All built-in and custom CIKernels written in CI's kernel language are automagically translated to OpenCL

Bridging Core Image and OpenCL

- Core Image kernel language has advantages
  - Write once and it will work across device classes and image formats
  - Automatically supports tiling of large images
  - Automatically optimized and concatenated by Core Image
- But some interesting image processing problems
  - Cannot be well expressed in CI's kernel language
  - Can be expressed in OpenCL's language
- How can you bridge Core Image and OpenCL to get the best of both?

Bridging Core Image and OpenCL

Alexandre Naaman
Provider of Clarity and Vision

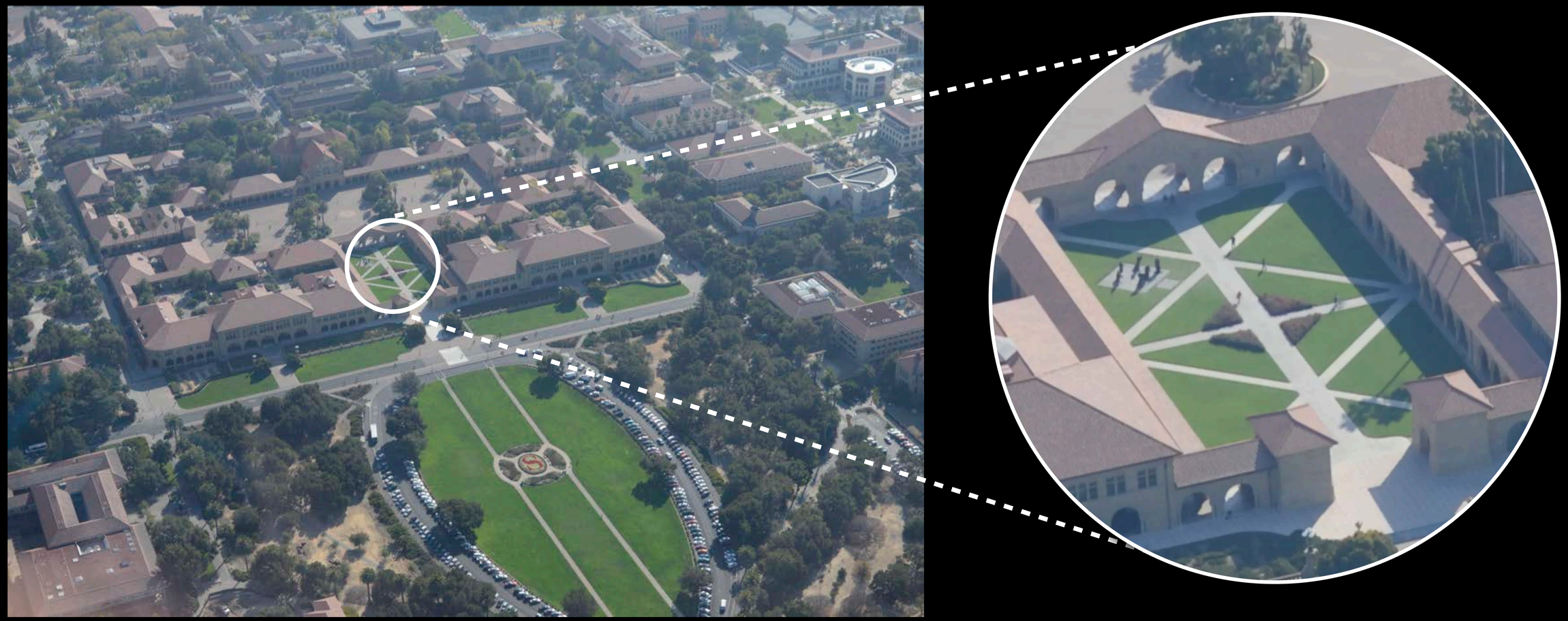
De-Haze

De-Haze

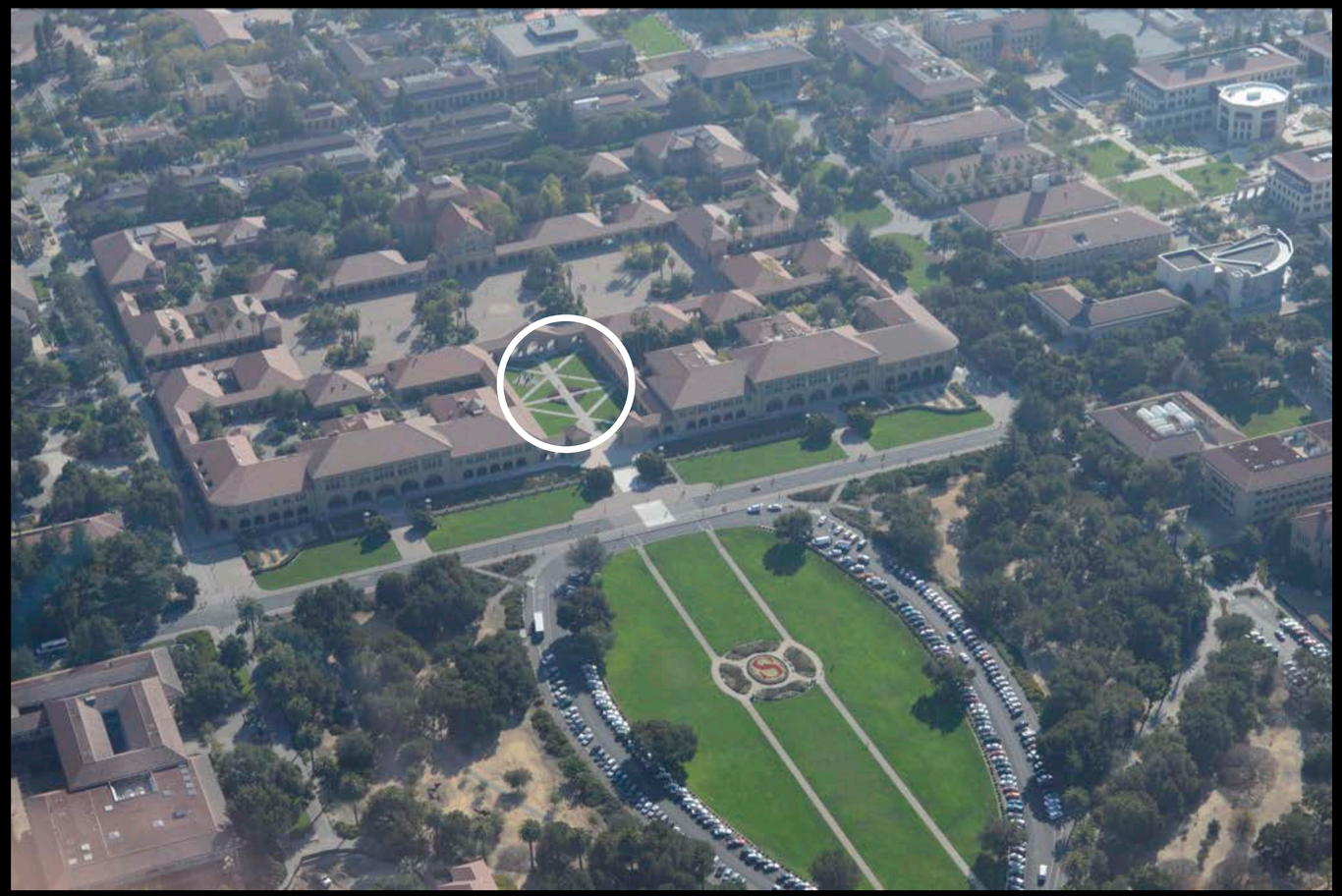
De-Haze

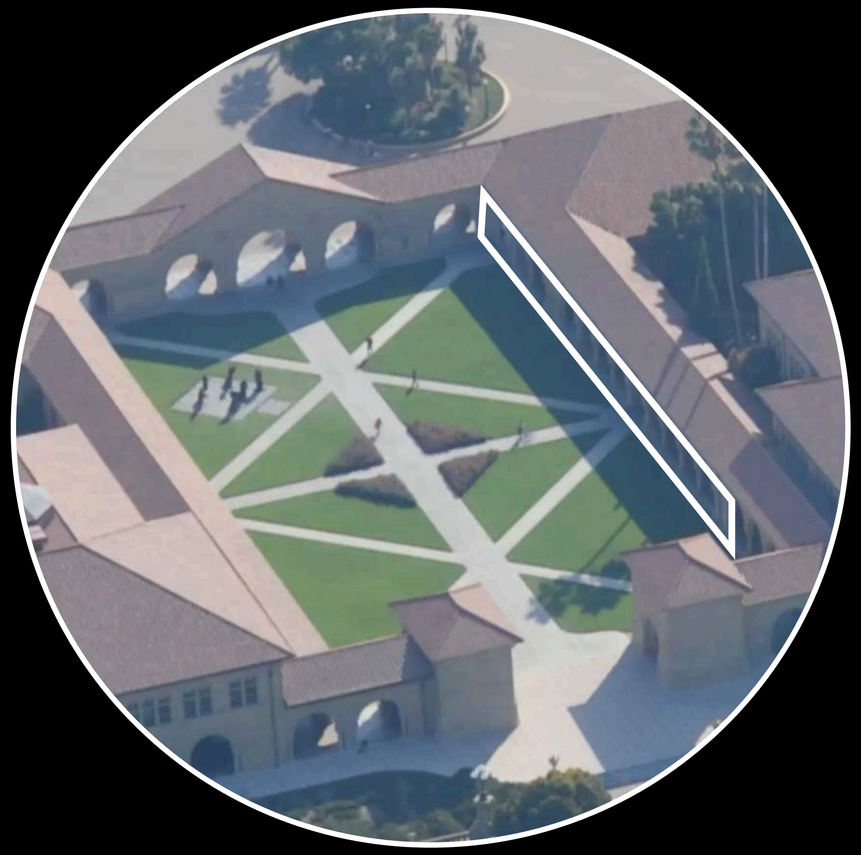
Theory

Theory

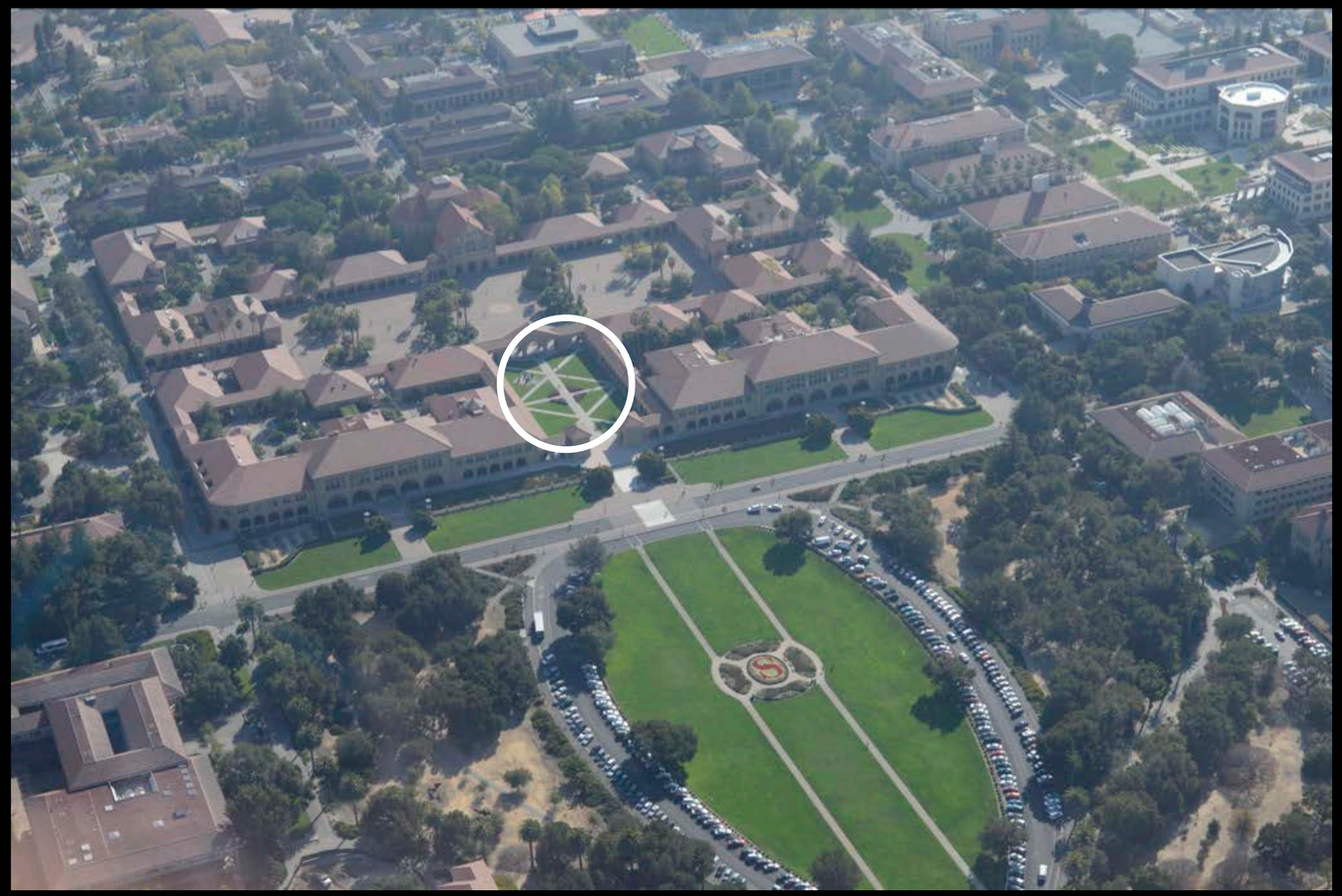


Theory

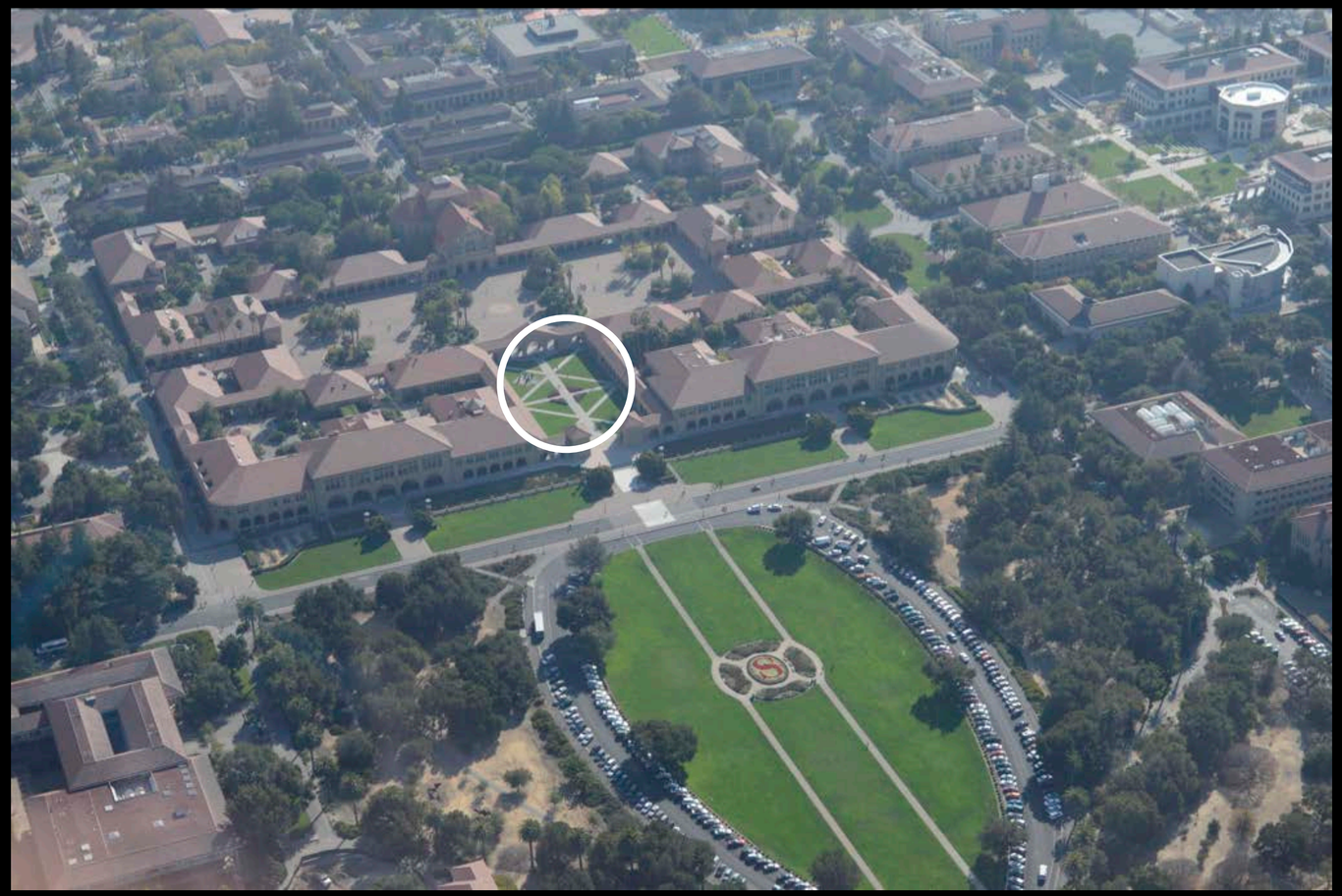




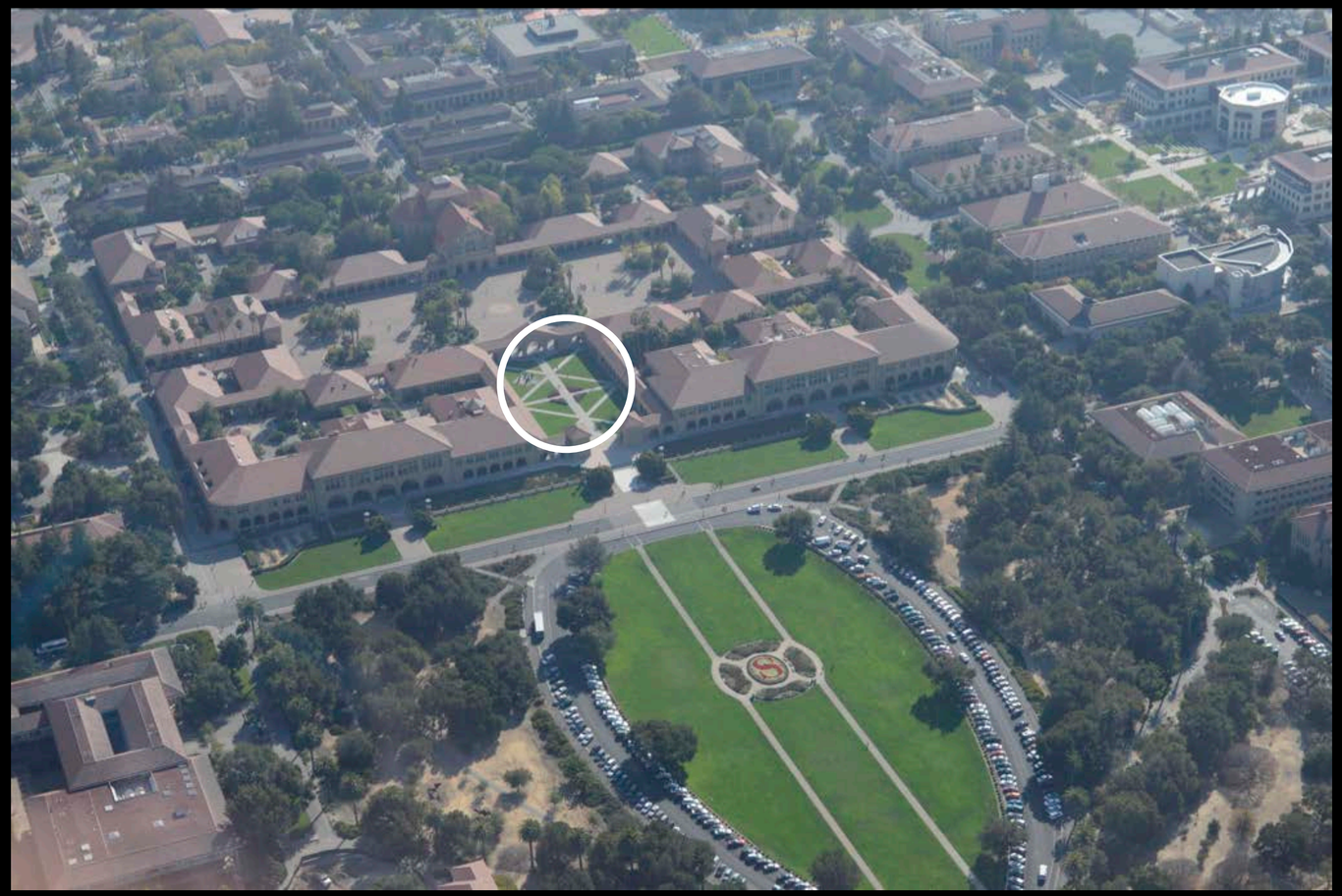
Theory



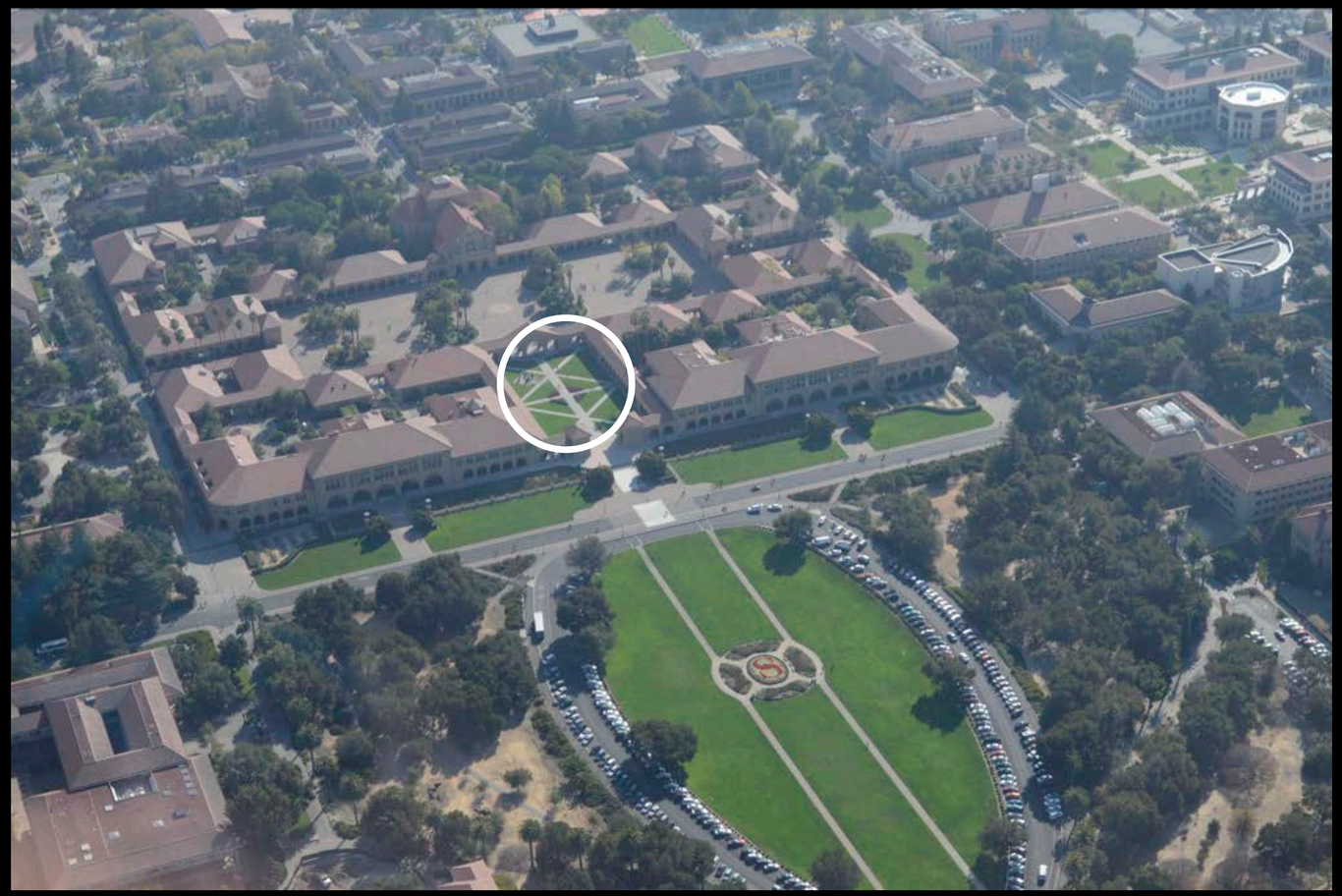
Theory



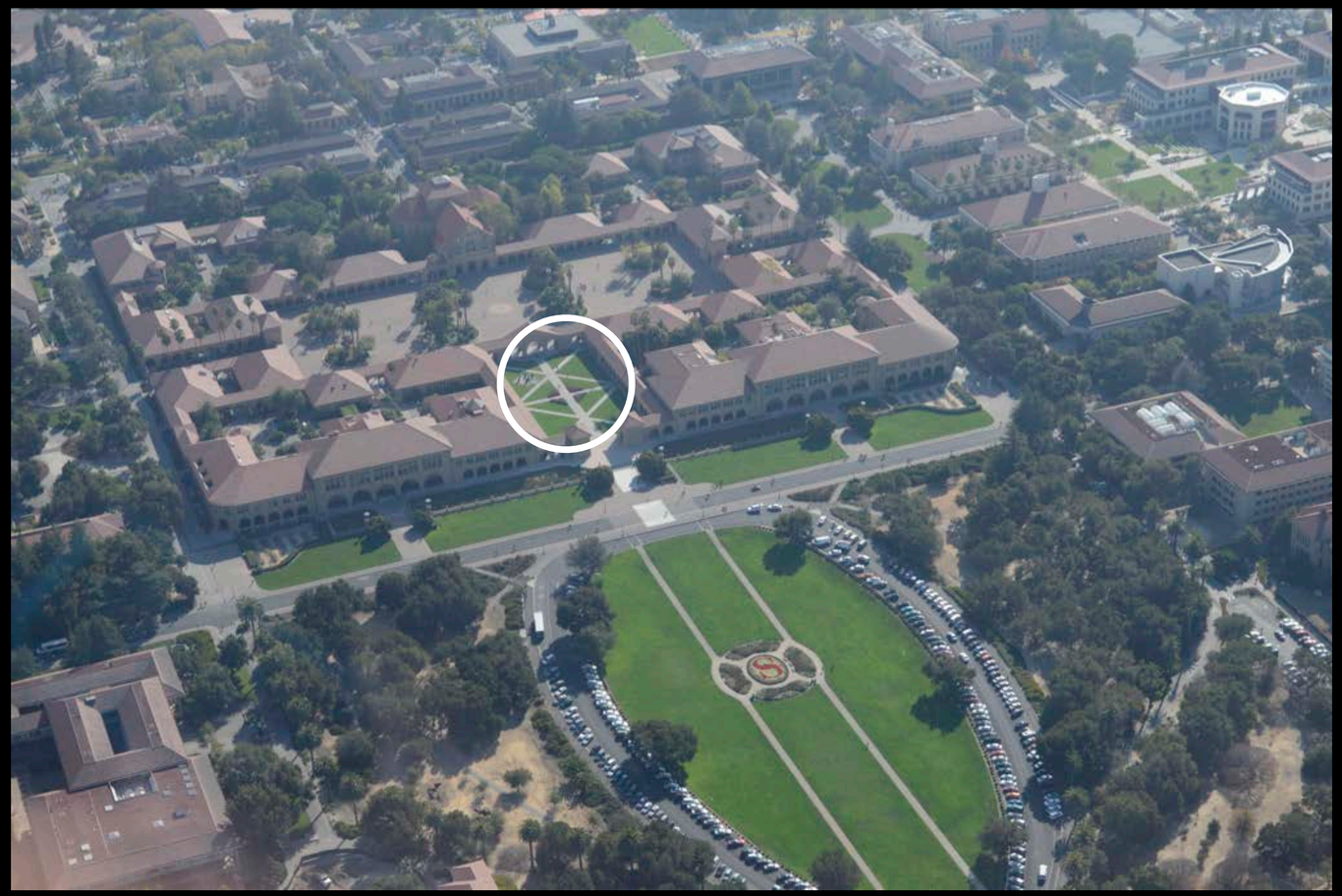
Theory



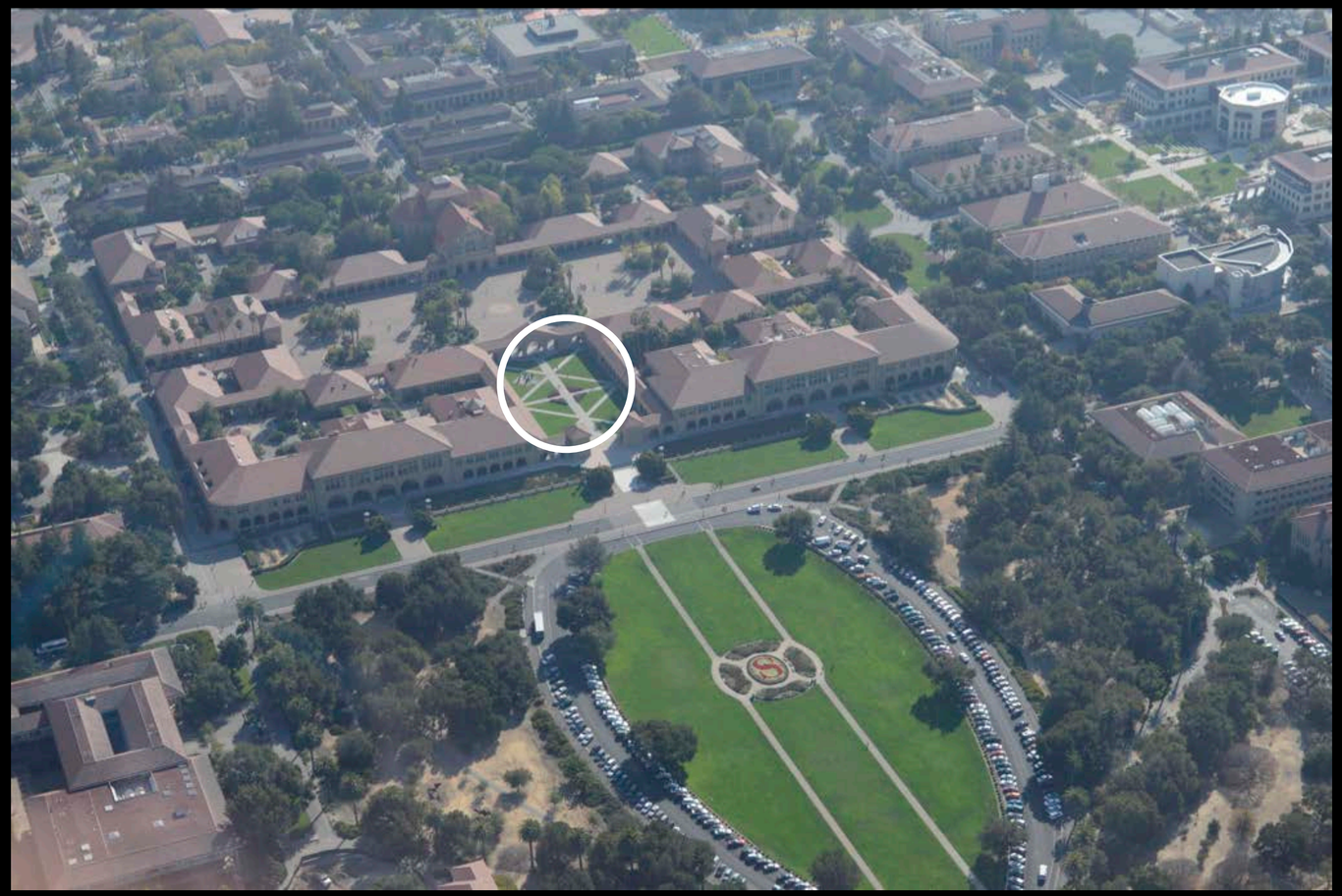
Theory



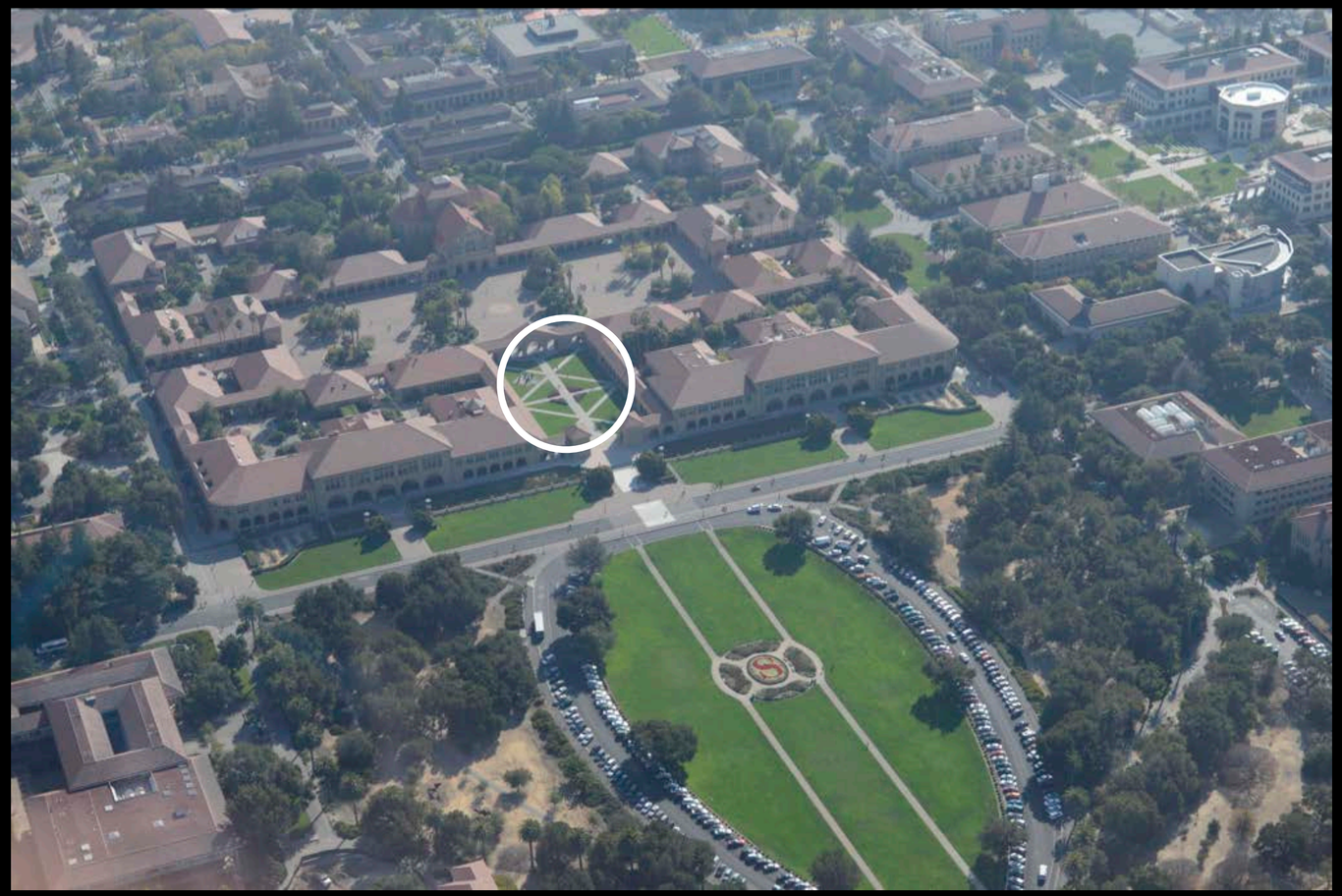
Theory



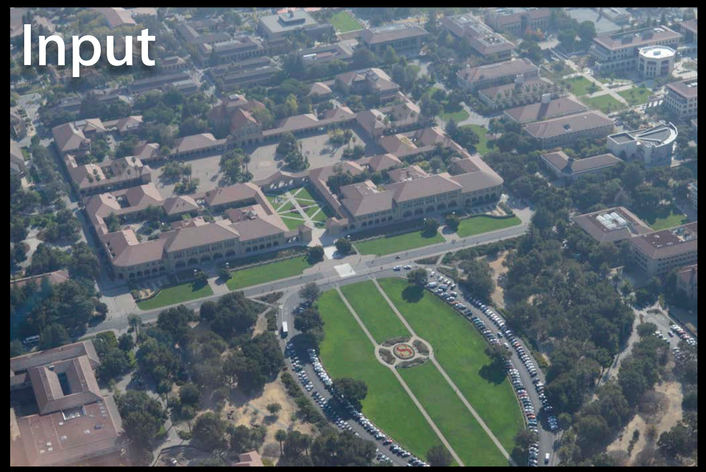
Theory



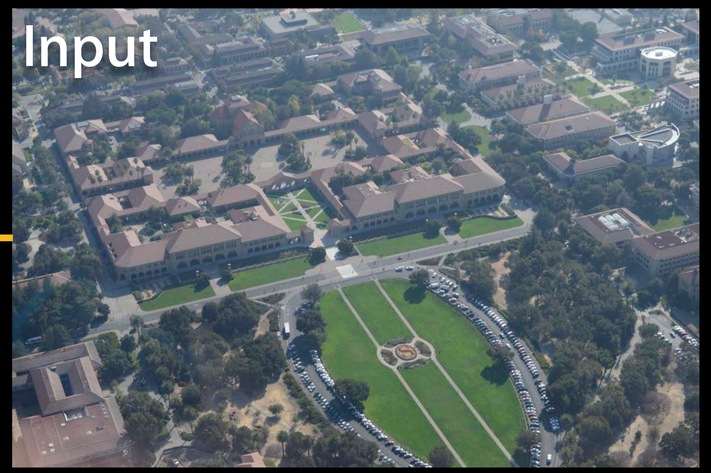
Theory



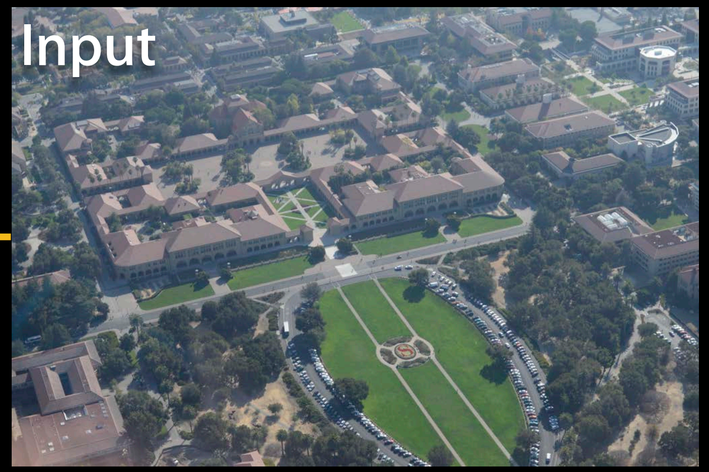
Workflow



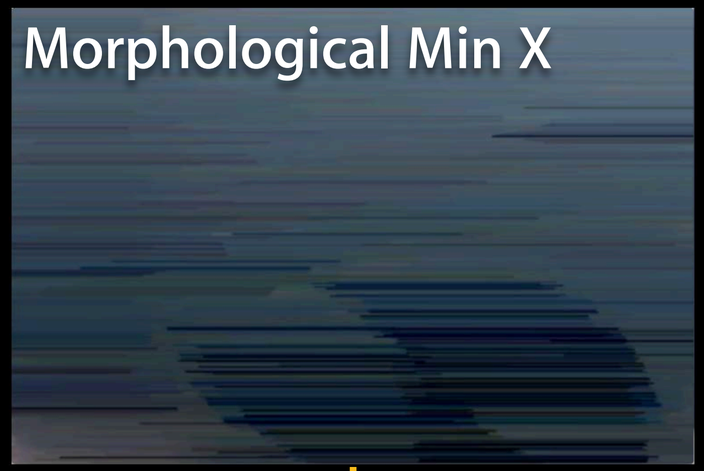
Workflow

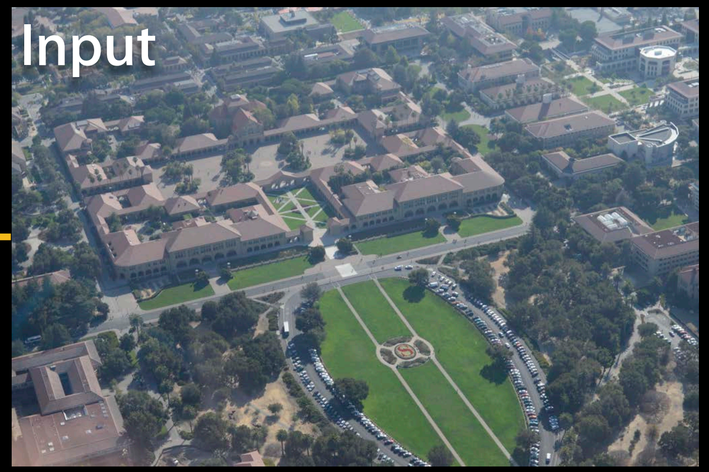


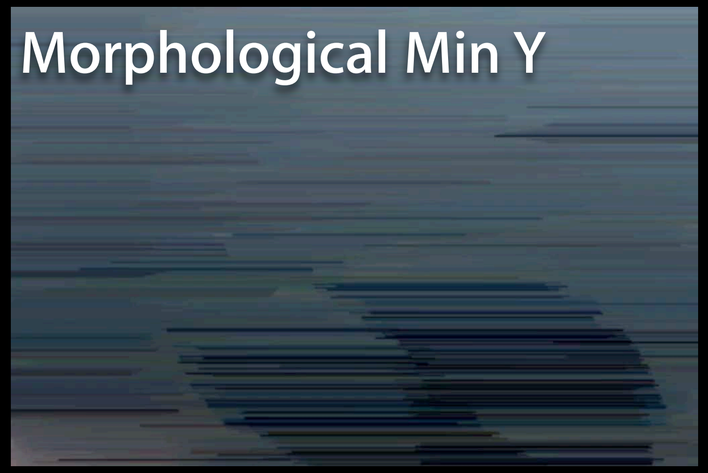
Workflow



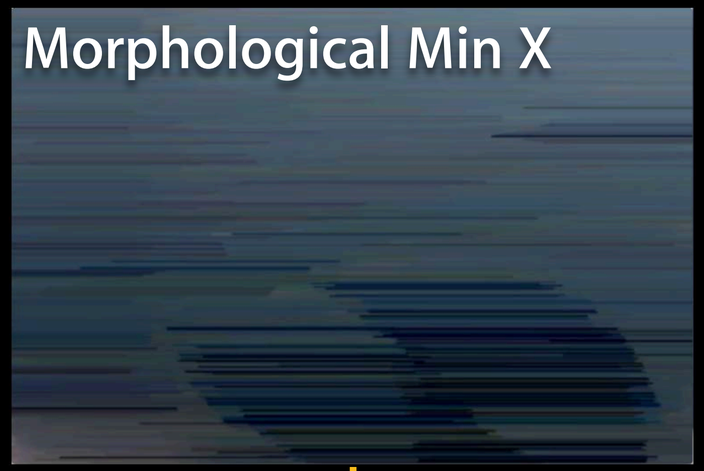
Workflow

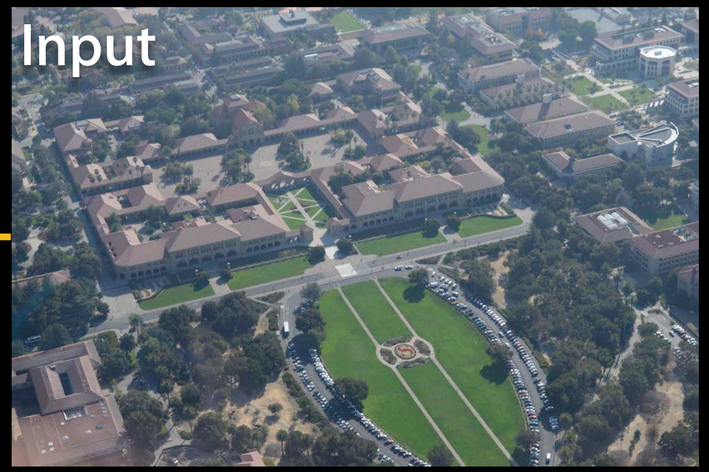




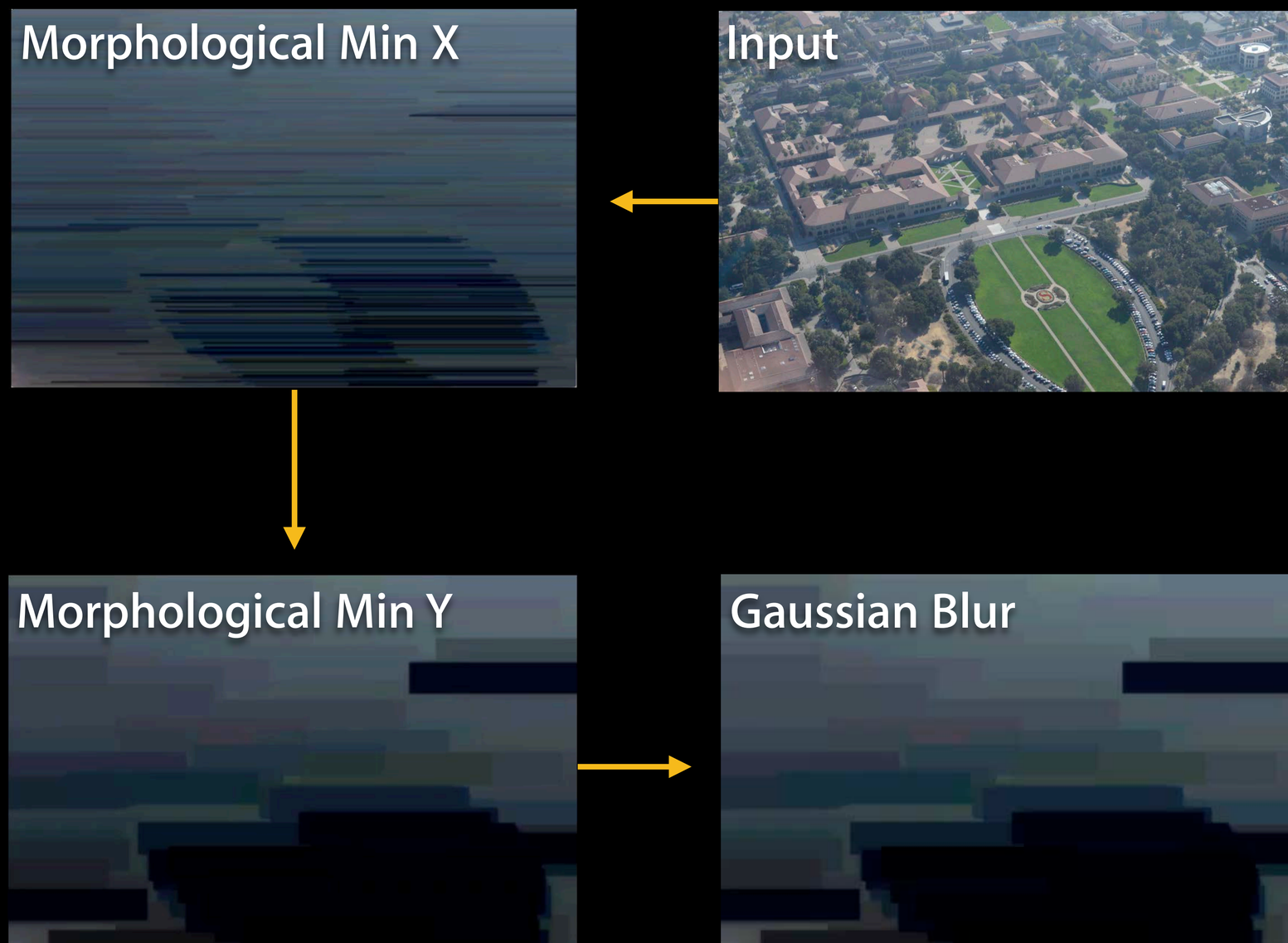


Workflow

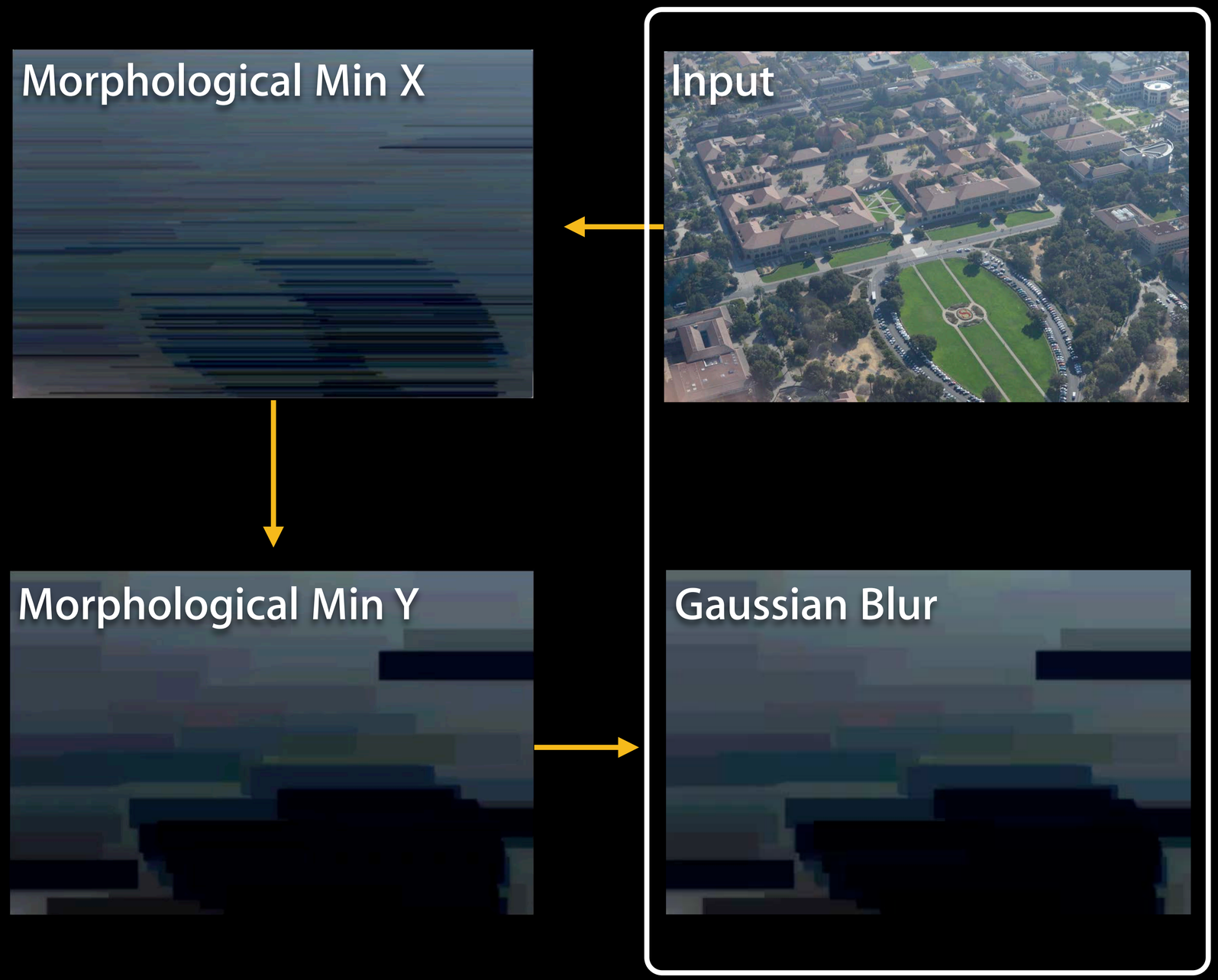




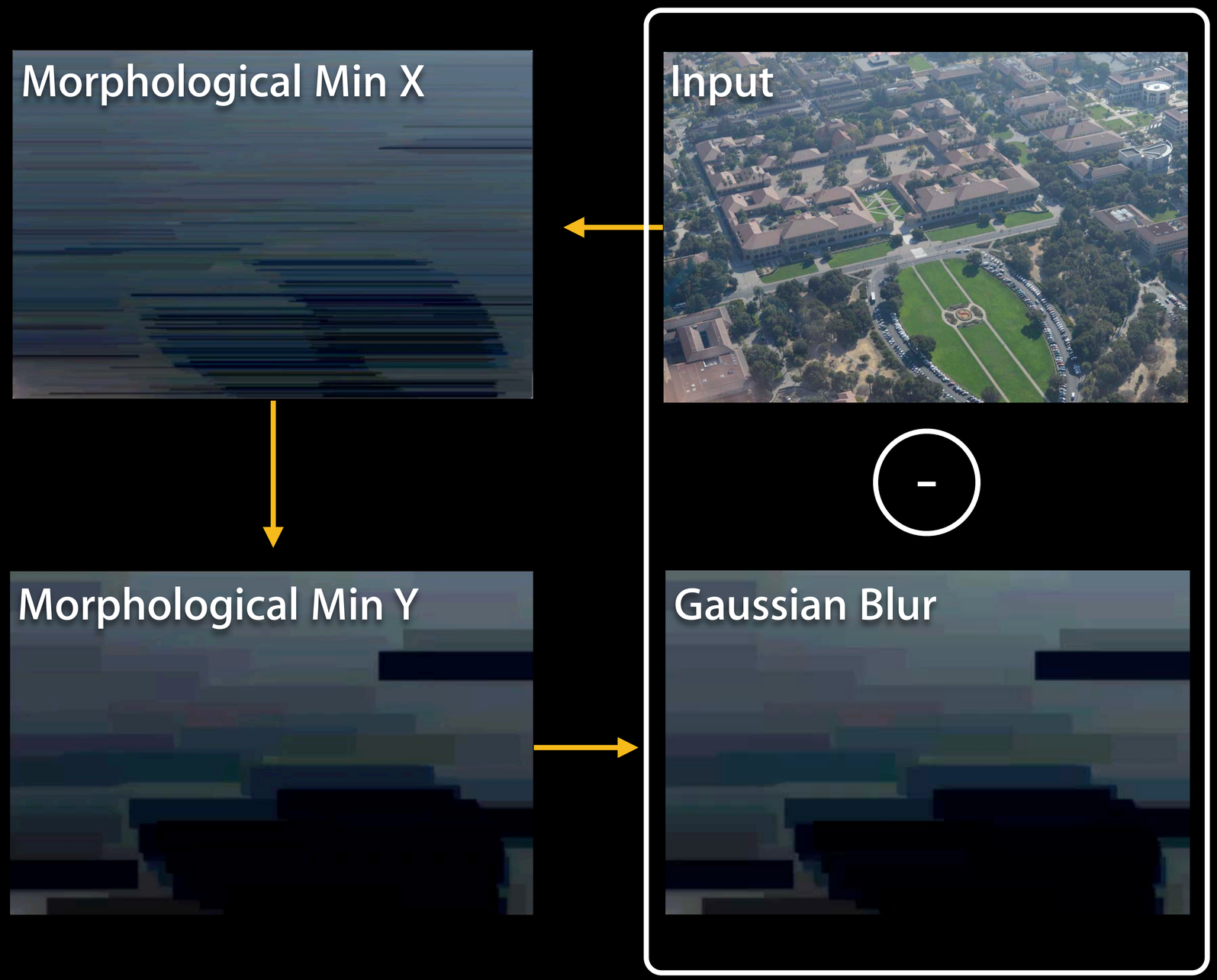
Workflow


Morphological Min X
Input
Morphological Min Y
Gaussian Blur

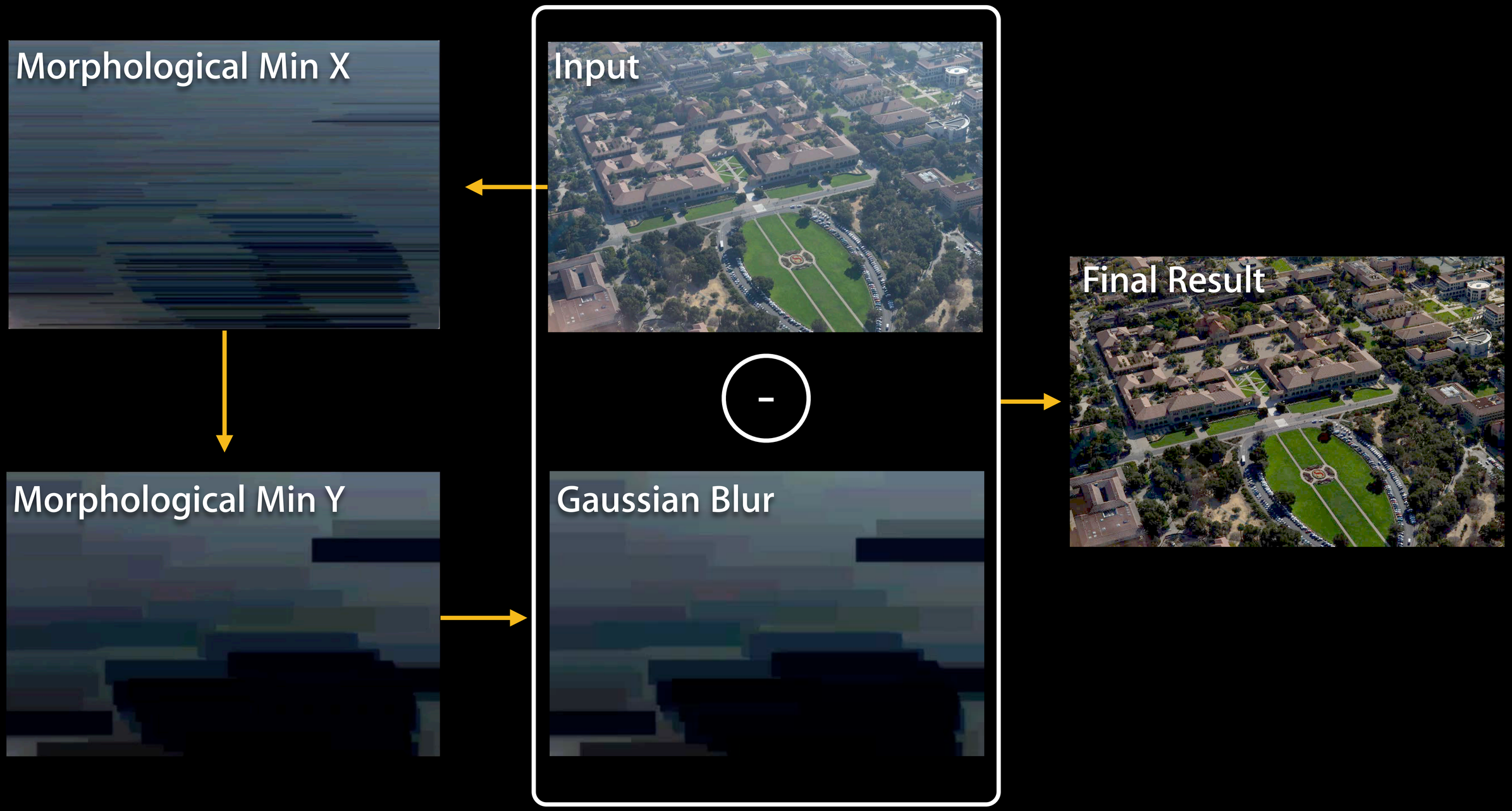
Workflow


Morphological Min X
Input
Morphological Min Y
Gaussian Blur

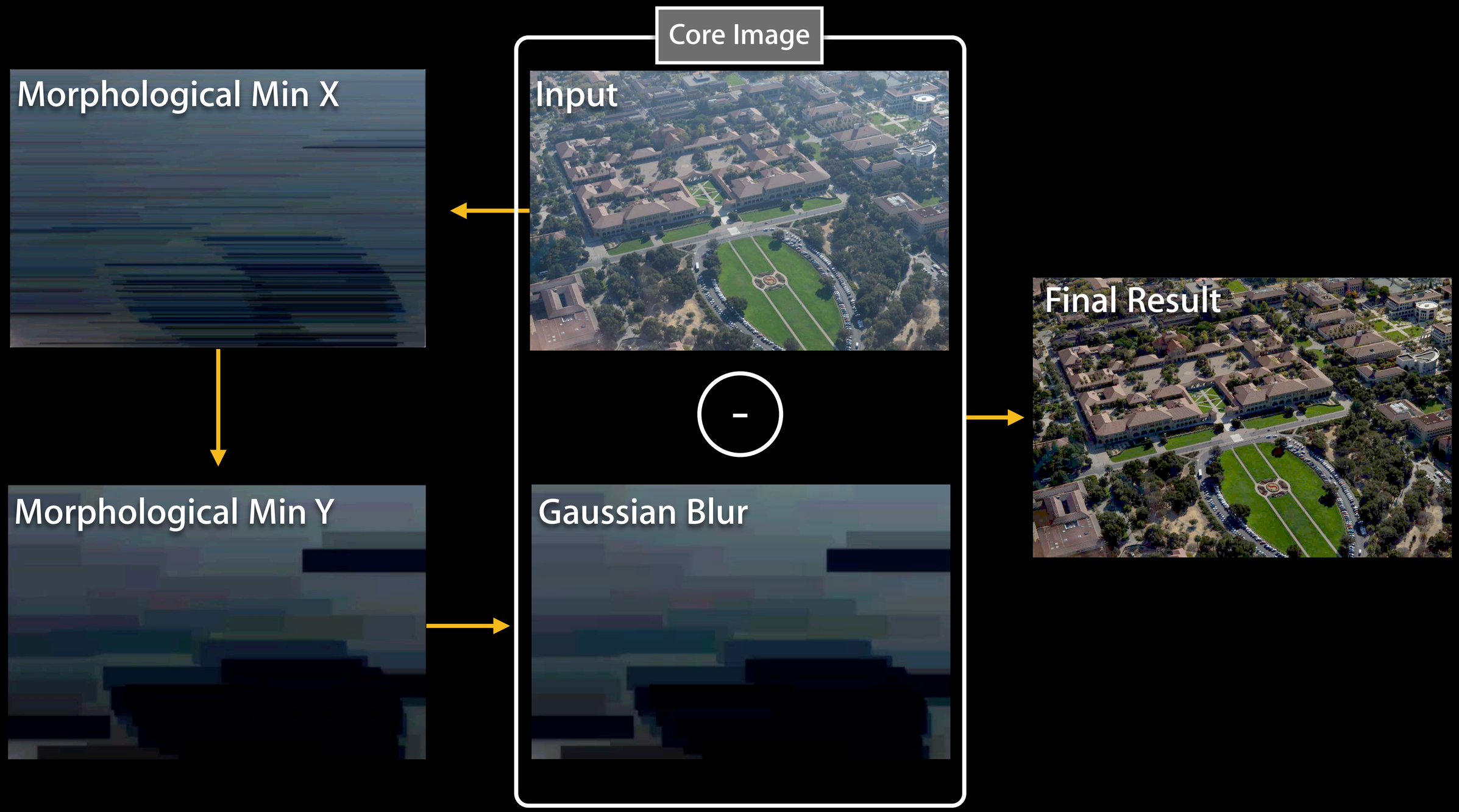
Workflow


Morphological Min X
Input
-
Morphological Min Y
Gaussian Blur

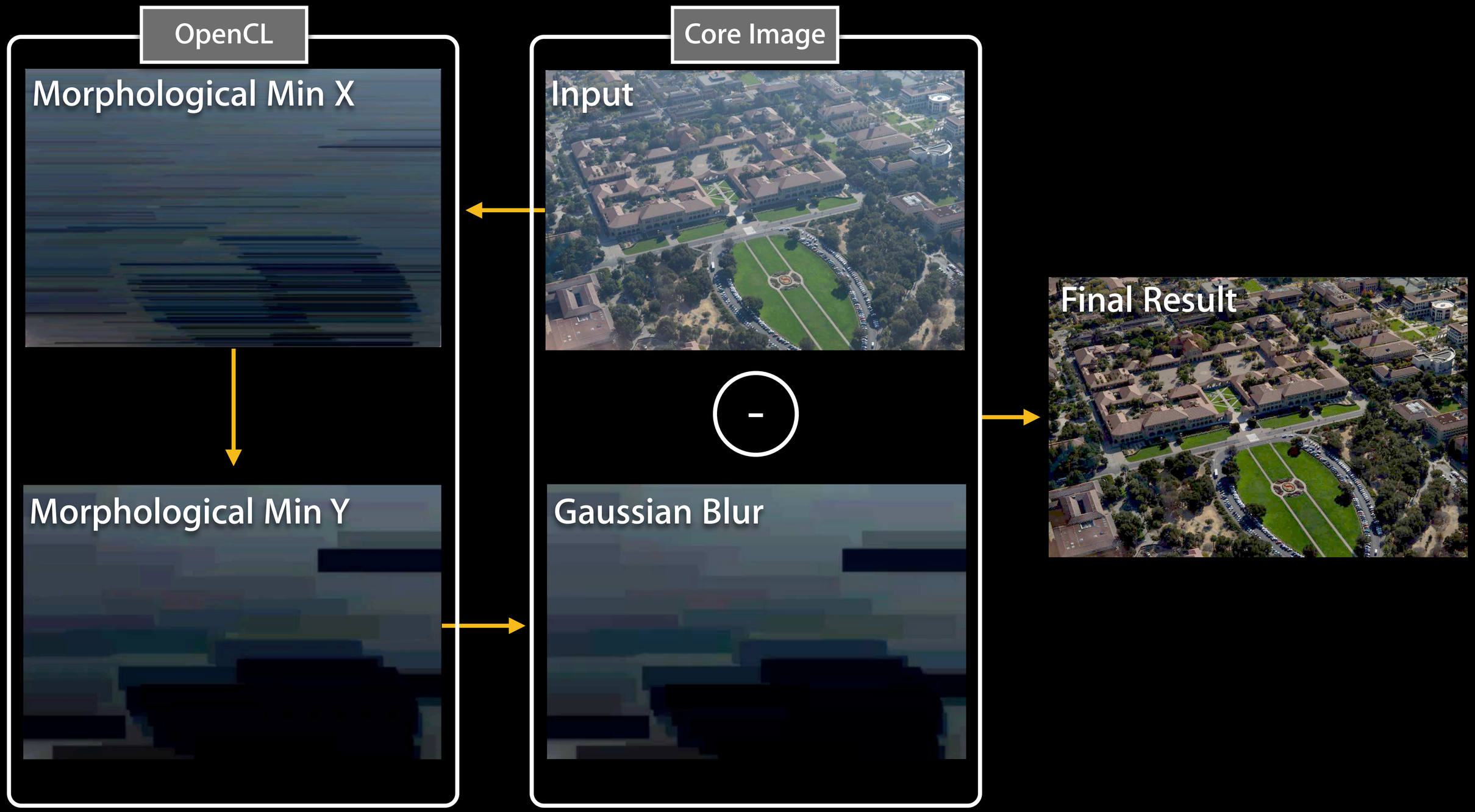
Workflow


Morphological Min X
Input
Final Result
Morphological Min Y
Gaussian Blur

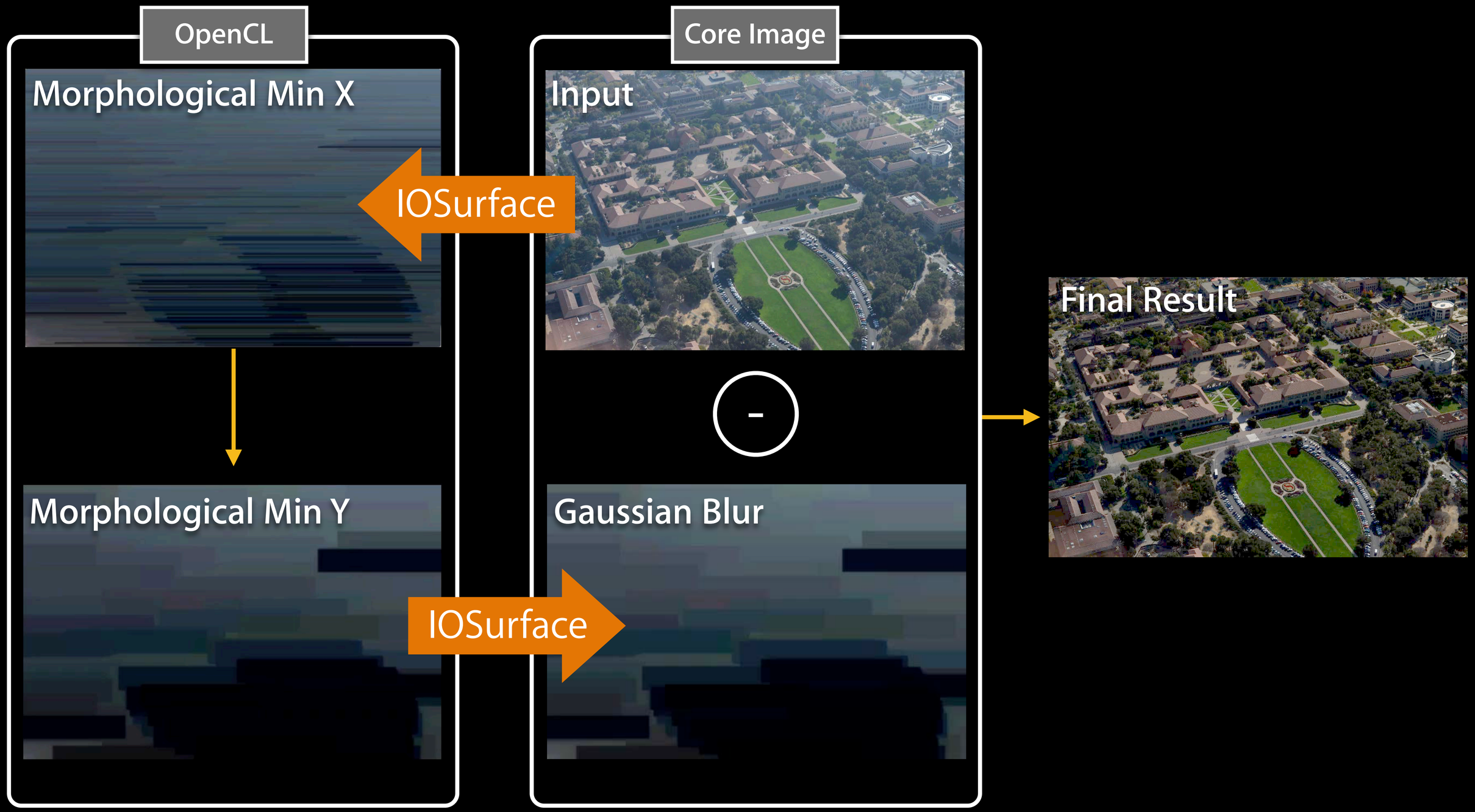
Workflow

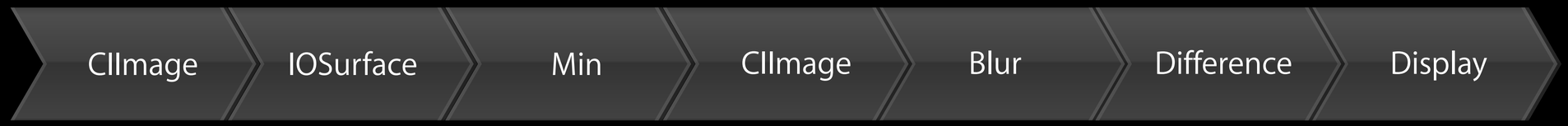

Core Image
Morphological Min X
Input
Morphological Min Y
Gaussian Blur
Final Result

Workflow


OpenCL
Morphological Min X
Morphological Min Y
Core Image
Input
-
Gaussian Blur
Final Result

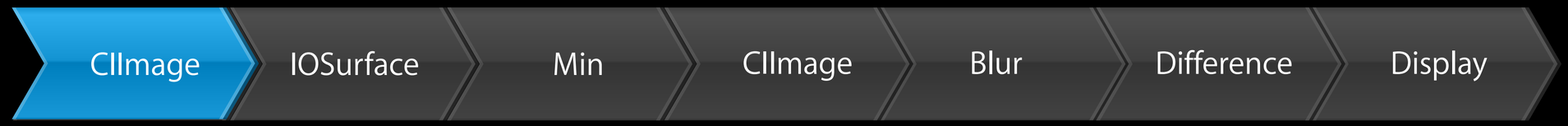
Workflow


OpenCL
Morphological Min X
IOSurface
Morphological Min Y
IOSurface
Core Image
Input
-
Gaussian Blur
Final Result



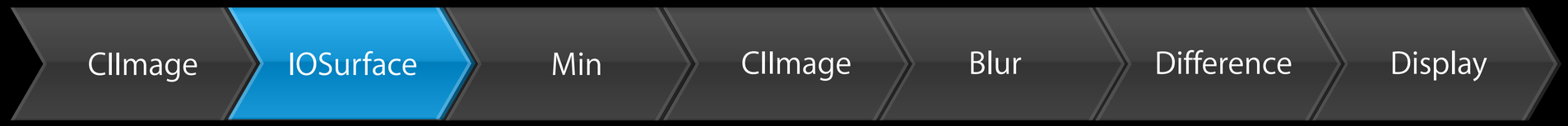
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



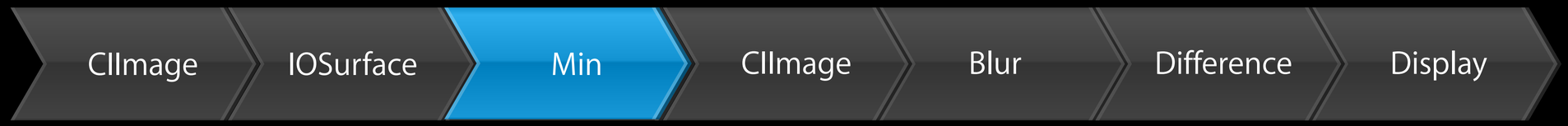
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result

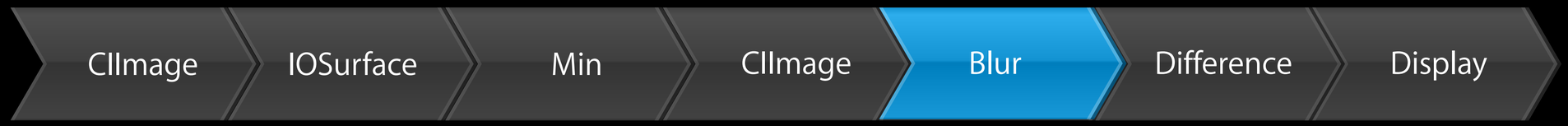


Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result

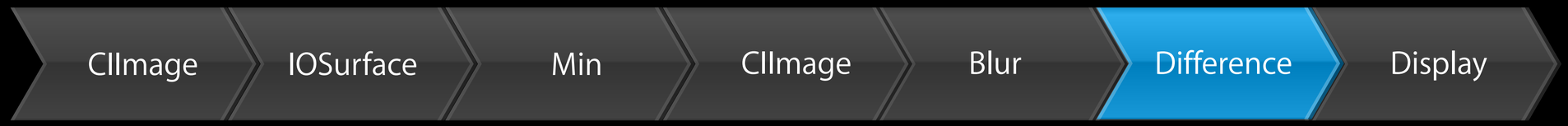
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



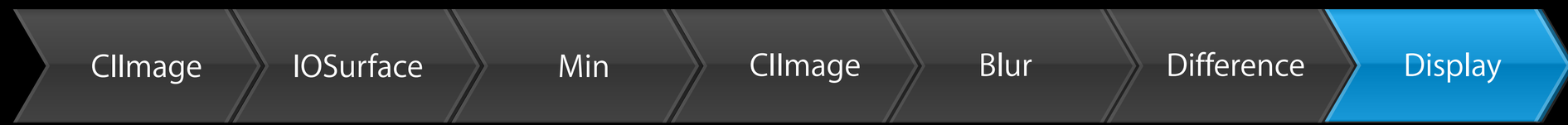
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



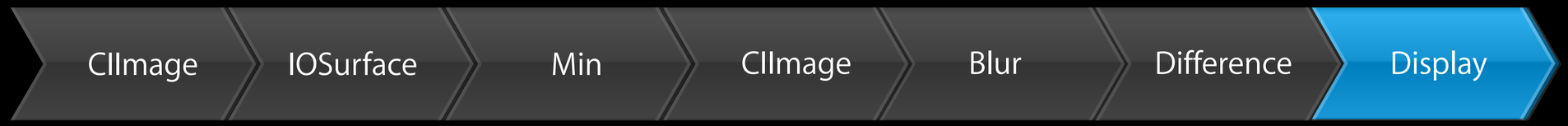
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



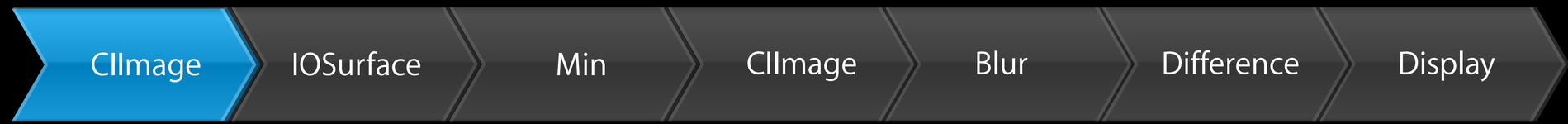
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



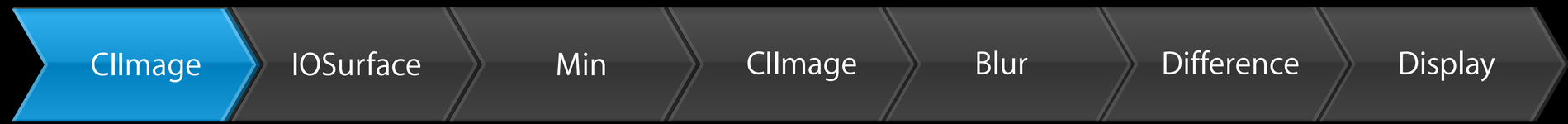
Task Breakdown

1. Use Core Image to downsample input image
2. Render to IOSurface
3. Use OpenCL to compute minimum
4. Create CIImage from output IOSurface
5. Blur resulting CIImage
6. Perform difference blending of blurred image with original input image
7. Render final result



Import Image and Downsample

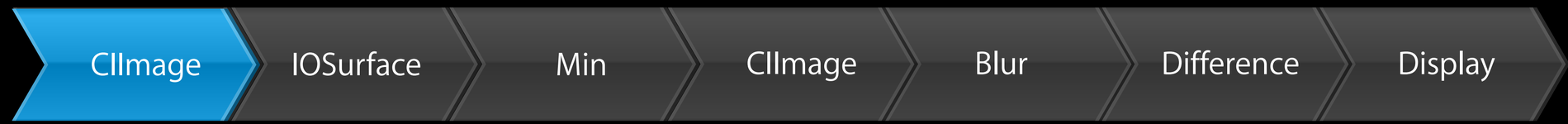
```
NSURL *url = [NSURL fileURLWithPath : filename];

CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



Import Image and Downsample

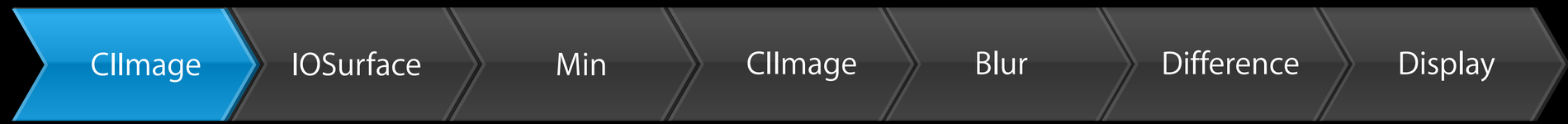
```
NSURL *url = [NSURL fileURLWithPath : filename];
```

```
CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



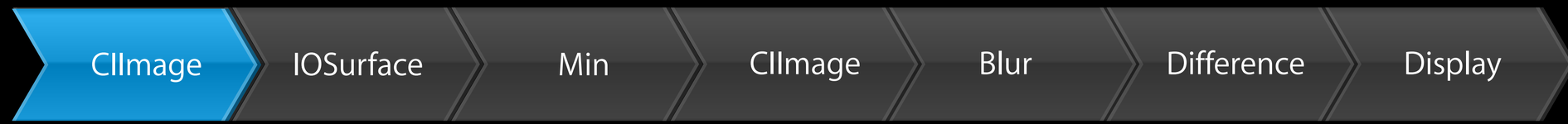
Import Image and Downsample

```
NSURL *url = [NSURL fileURLWithPath : filename];

CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



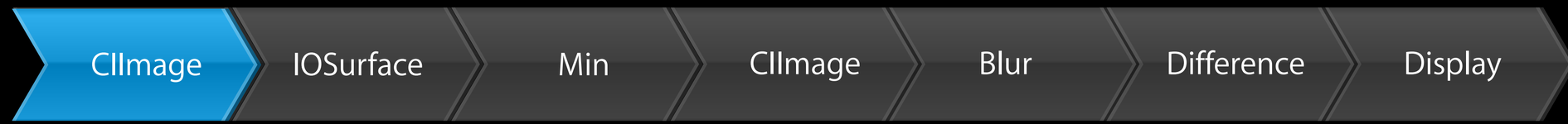
Import Image and Downsample

```
NSURL *url = [NSURL fileURLWithPath : filename];

CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



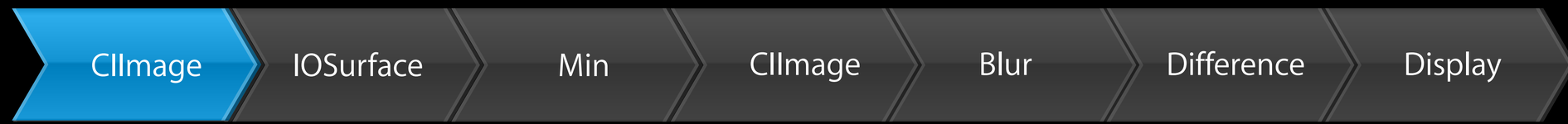
Import Image and Downsample

```
NSURL *url = [NSURL fileURLWithPath : filename];

CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



Import Image and Downsample

```
NSURL *url = [NSURL fileURLWithPath : filename];

CIImage *inputImage = [CIImage imageWithURL : url];

CIImage *scaledImage = [inputImage imageByApplyingTransform :
                         CGAffineTransformMakeScale ( scale , scale ) ];

CGRect insetRect = CGRectInset ( [scaledImage extent], 1.0f, 1.0f );

scaledImage = [scaledImage imageByCroppingToRect : insetRect];
```



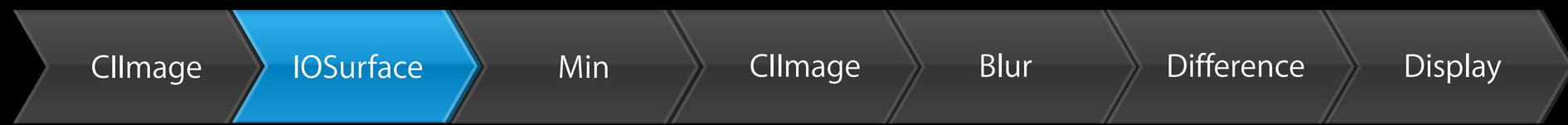
Render to IOSurface

- Create IOSurface

```
CFDictionaryRef props = (CFDictionaryRef)@{...}; // bpr, bpe, w, h, format
IOSurfaceRef inputSurface = IOSurfaceCreate ( props );
```

- Render scaledImage into newly created IOSurface

```
CGLContext cgl_ctx = ...; // your OpenGL context
CIContext *context = [CIContext contextWithCGLContext : cgl_ctx];

[context render:scaledImage toIOSurface:inputSurface
        bounds:[scaledImage extent] colorSpace:[inputImage colorSpace]];
```



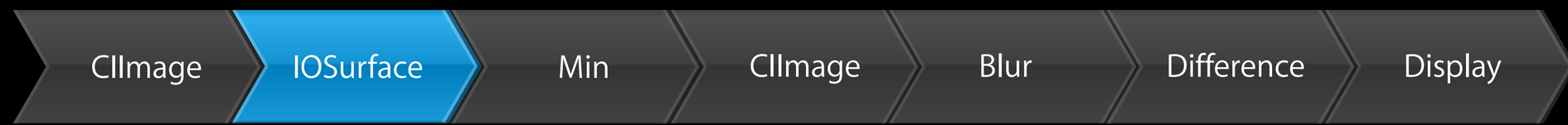
Render to IOSurface

- Create IOSurface

```
CFDictionaryRef props = (CFDictionaryRef)@{...}; // bpr, bpe, w, h, format
IOSurfaceRef inputSurface = IOSurfaceCreate ( props );
```

- Render scaledImage into newly created IOSurface

```
CGLContext cgl_ctx = ...; // your OpenGL context
CIContext *context = [CIContext contextWithCGLContext : cgl_ctx];

[context render:scaledImage toIOSurface:inputSurface
        bounds:[scaledImage extent] colorSpace:[inputImage colorSpace]];
```



Render to IOSurface

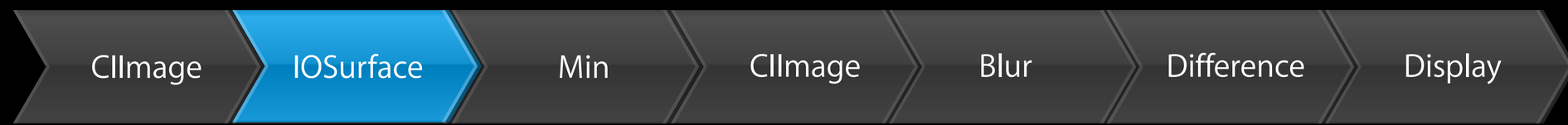
- Create IOSurface

```
CFDictionaryRef props = (CFDictionaryRef)@{...}; // bpr, bpe, w, h, format
IOSurfaceRef inputSurface = IOSurfaceCreate ( props );
```

- Render scaledImage into newly created IOSurface

```
CGLContext cgl_ctx = ...; // your OpenGL context
CIContext *context = [CIContext contextWithCGLContext : cgl_ctx];
```

```
[context render:scaledImage toIOSurface:inputSurface
        bounds:[scaledImage extent] colorSpace:[inputImage colorSpace]];
```



Render to IOSurface

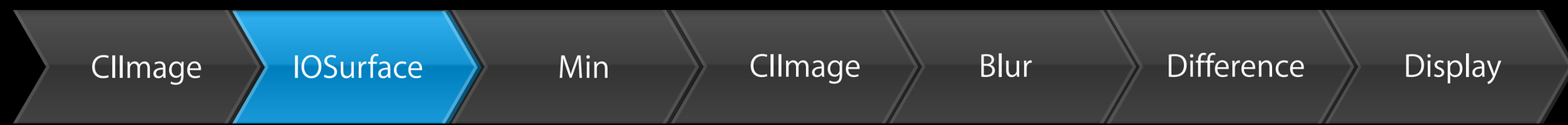
- Create IOSurface

```
CFDictionaryRef props = (CFDictionaryRef)@{...}; // bpr, bpe, w, h, format
IOSurfaceRef inputSurface = IOSurfaceCreate ( props );
```

- Render scaledImage into newly created IOSurface

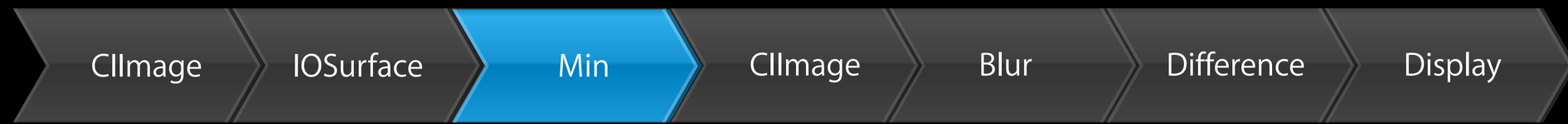
```
CGLContext cgl_ctx = ...; // your OpenGL context
CIContext *context = [CIContext contextWithCGLContext : cgl_ctx];
```

```
[context render:scaledImage toIOSurface:inputSurface
        bounds:[scaledImage extent] colorSpace:[inputImage colorSpace]];
```



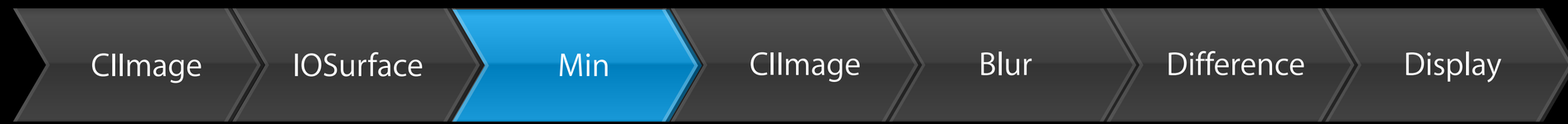
OpenCL: Create Context and Mem Objects

```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



OpenCL: Create Context and Mem Objects

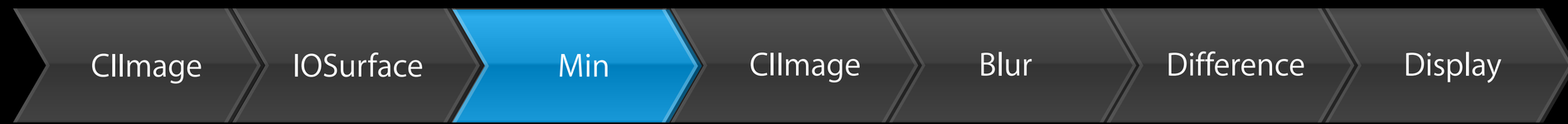
```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



OpenCL: Create Context and Mem Objects

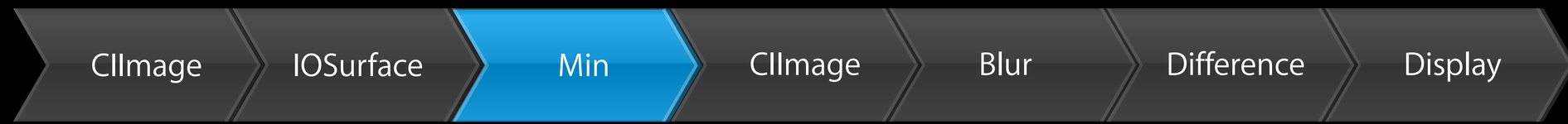
```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );
```

```
cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



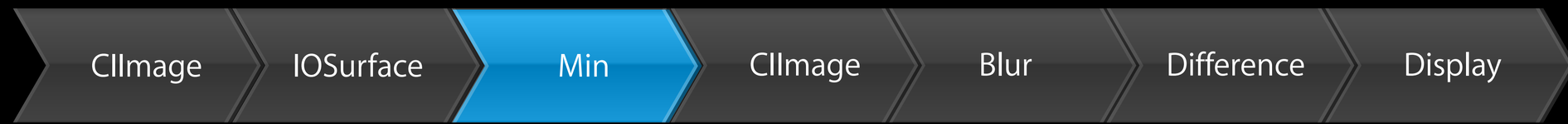
OpenCL: Create Context and Mem Objects

```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
            CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



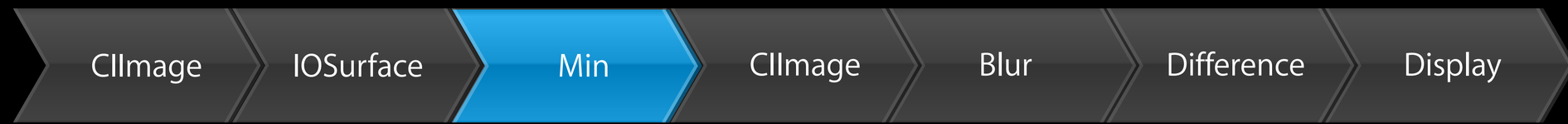
OpenCL: Create Context and Mem Objects

```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



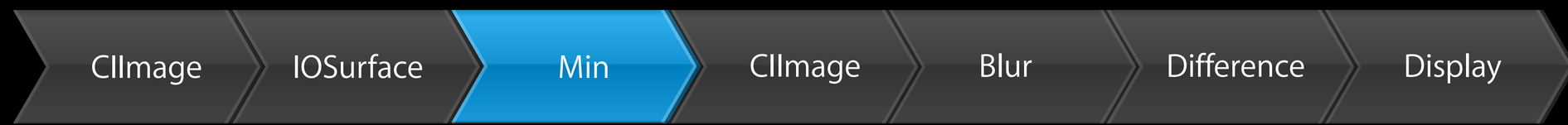
OpenCL: Create Context and Mem Objects

```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



OpenCL: Create Context and Mem Objects

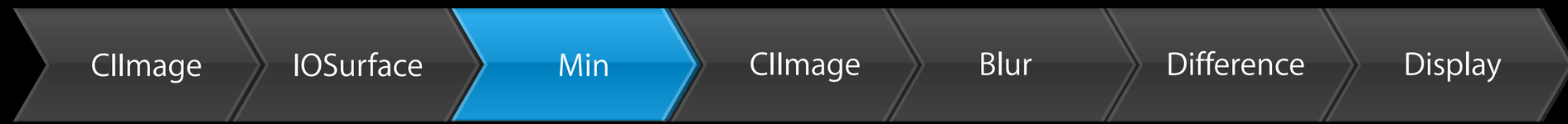
```
CGLContext cgl_ctx = ...; // your OpenGL context
cl_context_properties properties[] =
                        { CL_CONTEXT_PROPERTY_USE_CGL_SHAREGROUP_APPLE,
                          (intptr_t) CGLGetShareGroup ( cgl_ctx ), 0 };
cl_context clctx = clCreateContext ( properties, 0, NULL, NULL, NULL, &err );

cl_image_format format = { CL_BGRA, CL_UNORM_INT8 };
cl_mem inputImage = clCreateImageFromIOSurface2DAPPLE ( clctx,
             CL_MEM_READ_ONLY, &format, width, height, inputSurface, &err);

cl_image_desc d = { CL_MEM_OBJECT_IMAGE_2D, width, height, 0,0,0,0,0,0,NULL};
cl_mem intermediateImage = clCreateImage ( clctx, CL_MEM_READ_WRITE,
                                           &format, &d, NULL, &err);

cl_mem outputImage = clCreateImageFromIOSurface2DAPPLE ( ... ); // write only
```



OpenCL: Kernel Code

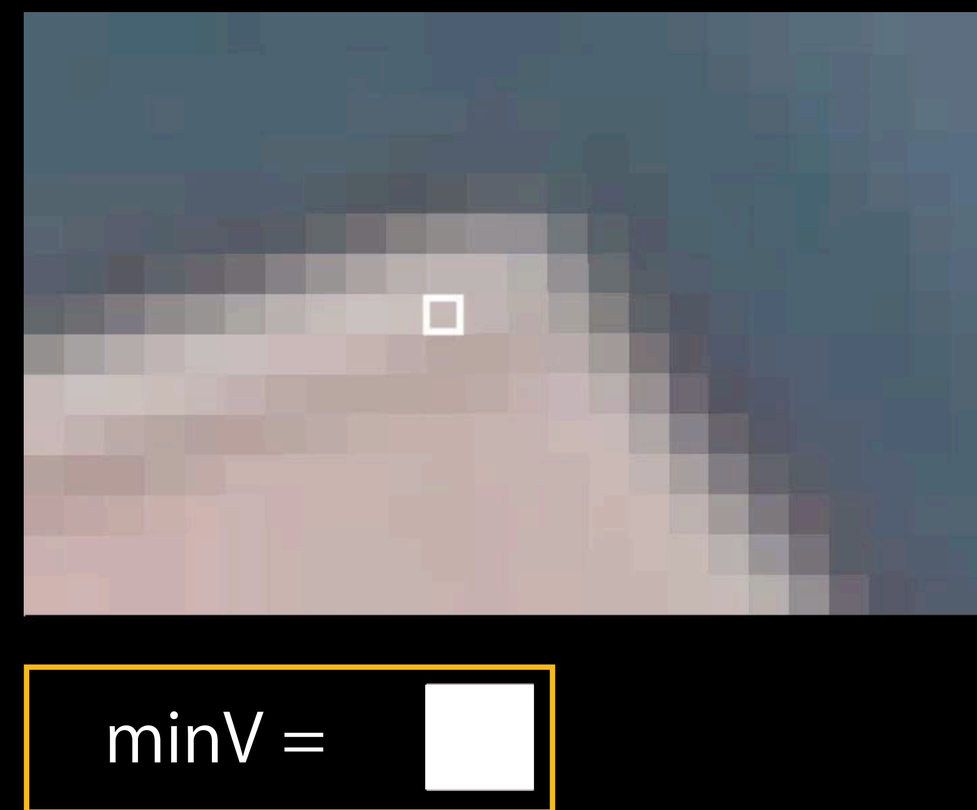
```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{

}
```

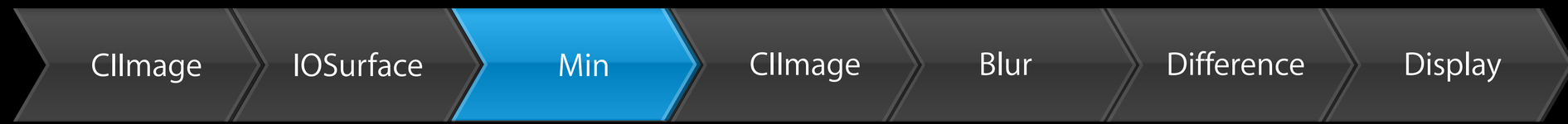
minV =



OpenCL: Kernel Code

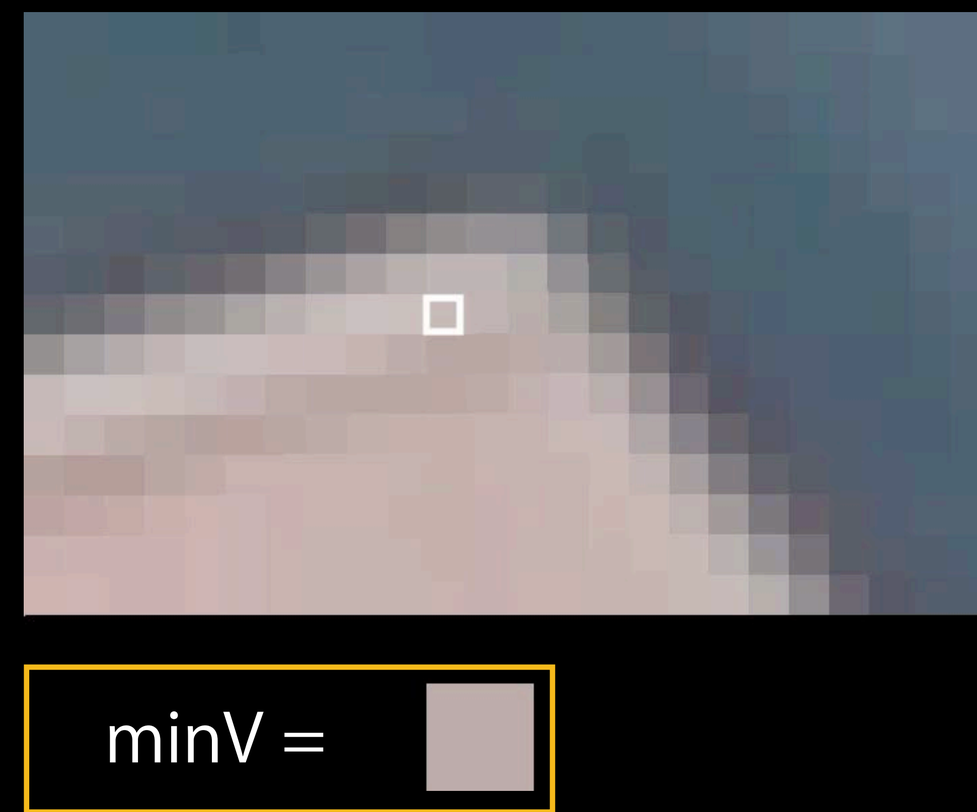
```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{

}
```

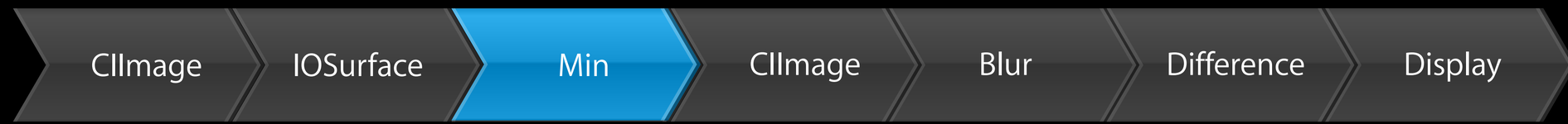




OpenCL: Kernel Code

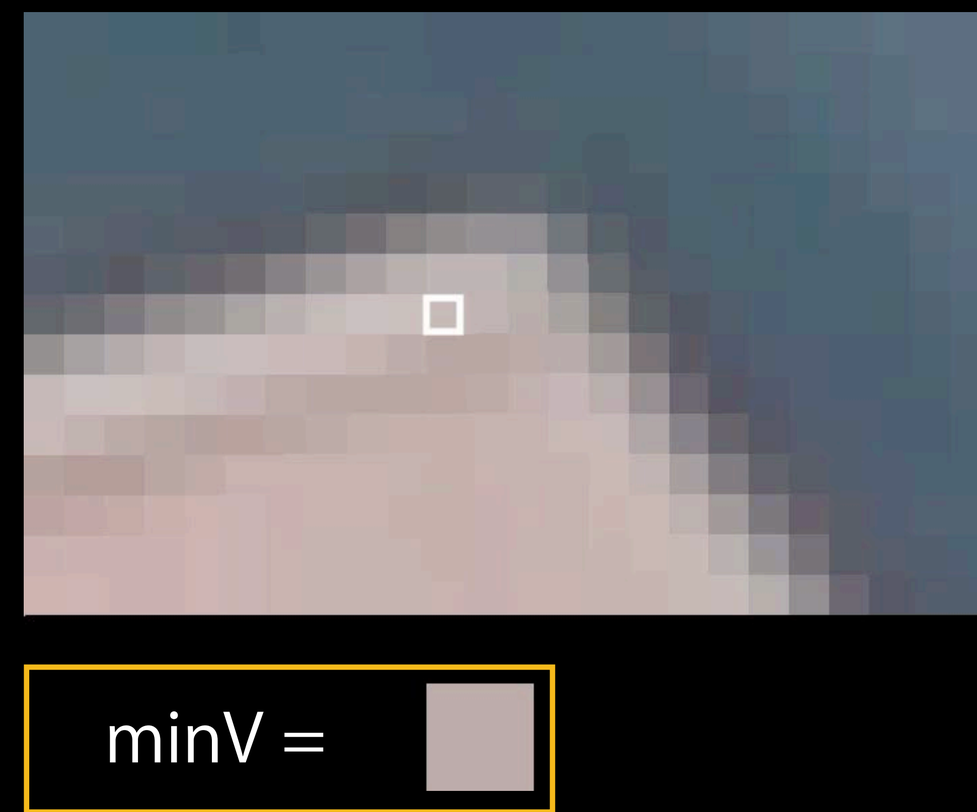
```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{

}
```

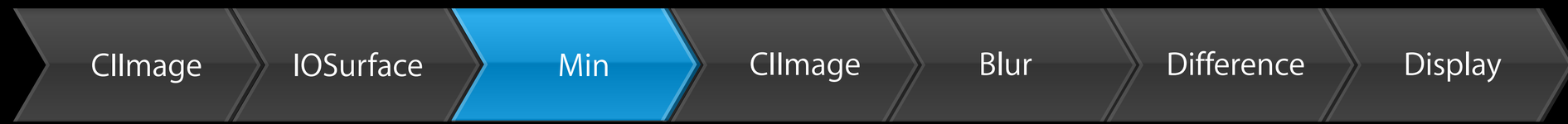




OpenCL: Kernel Code

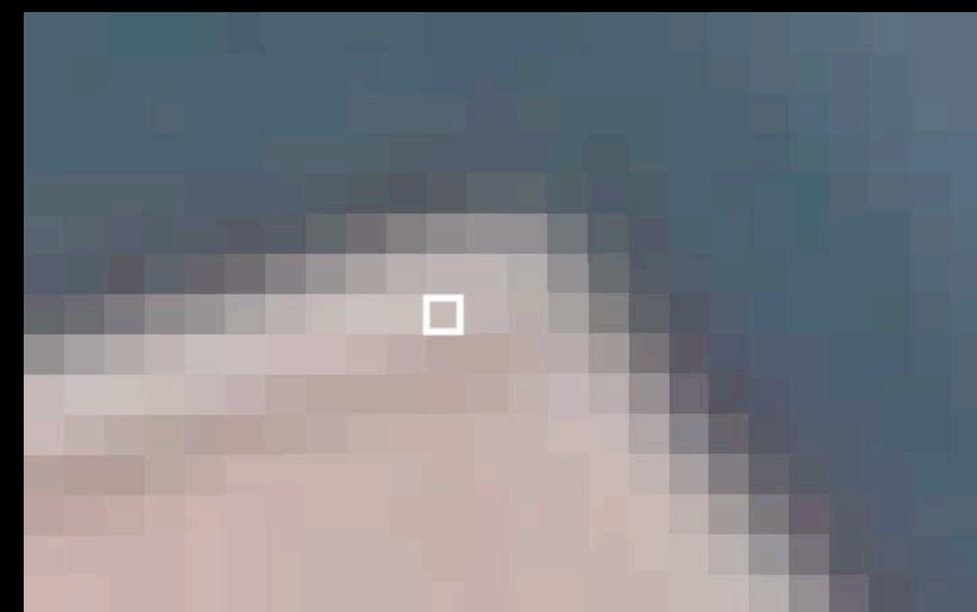
```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{

}
```



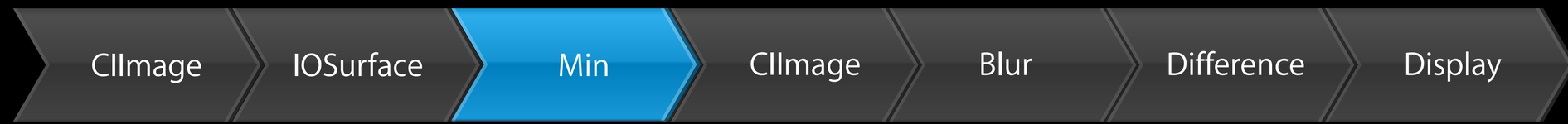


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

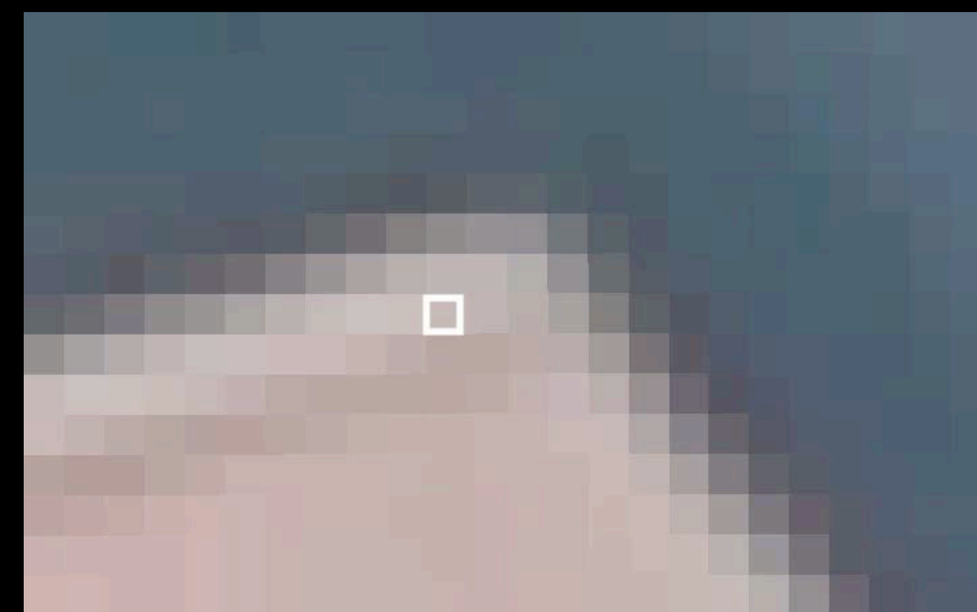


minV =

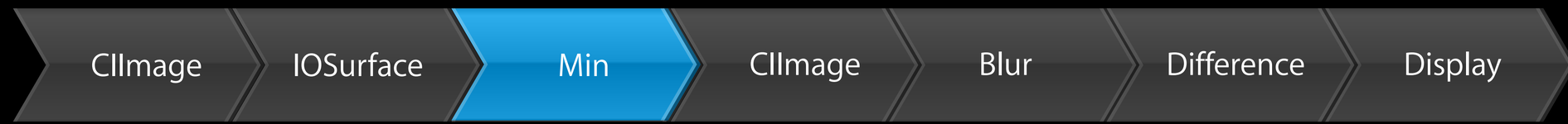


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

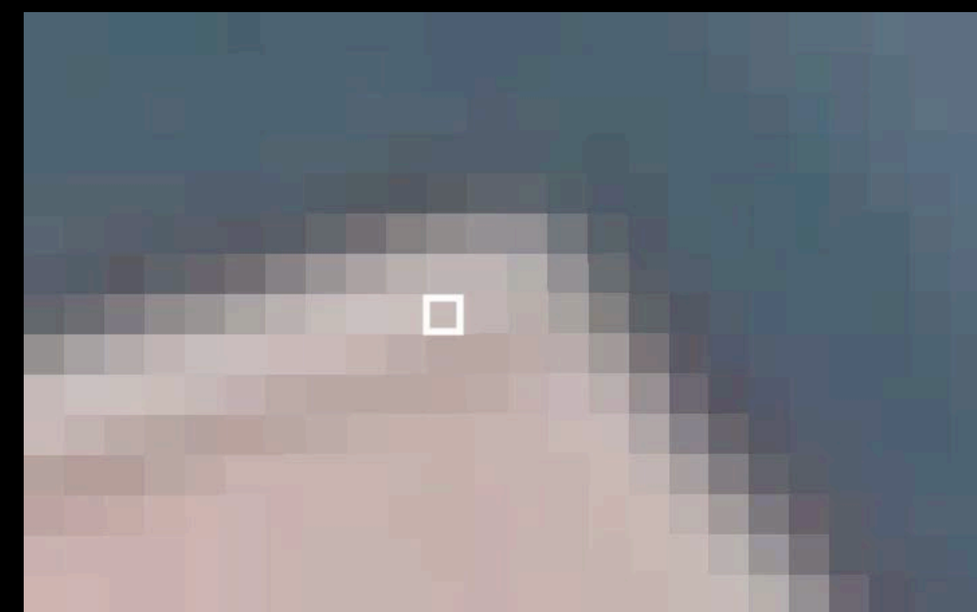


minV =

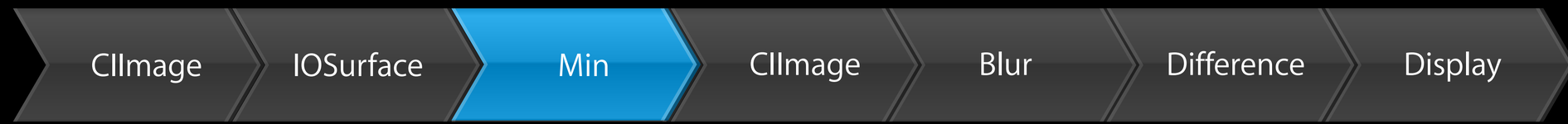


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

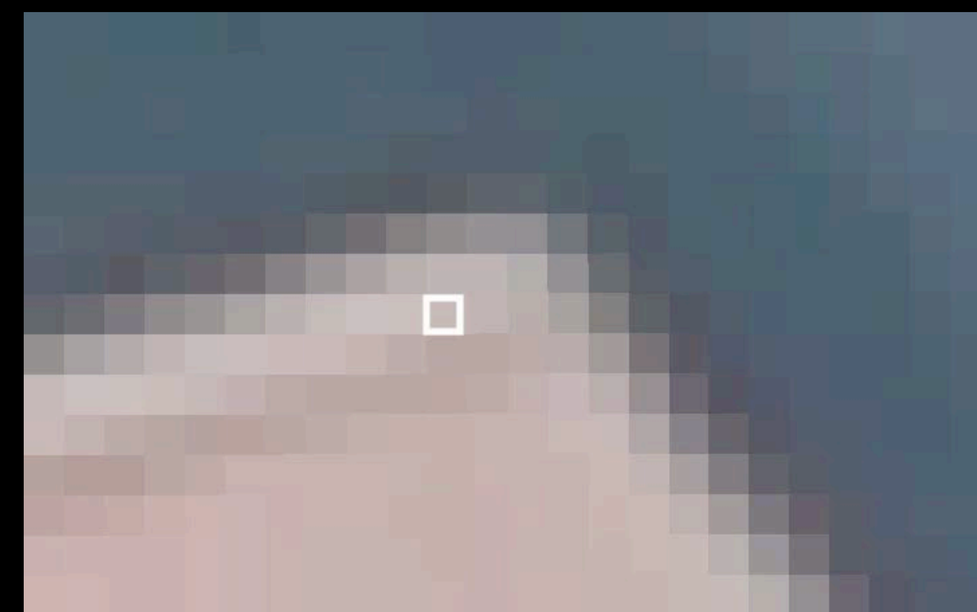


minV =

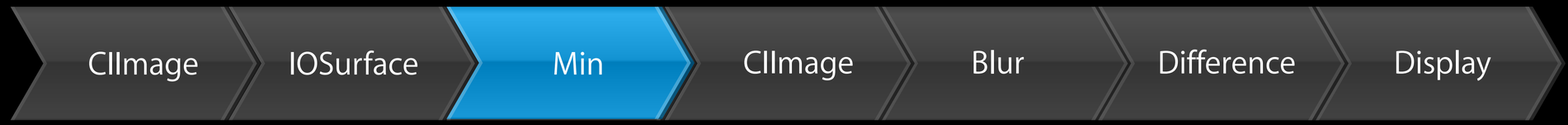


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

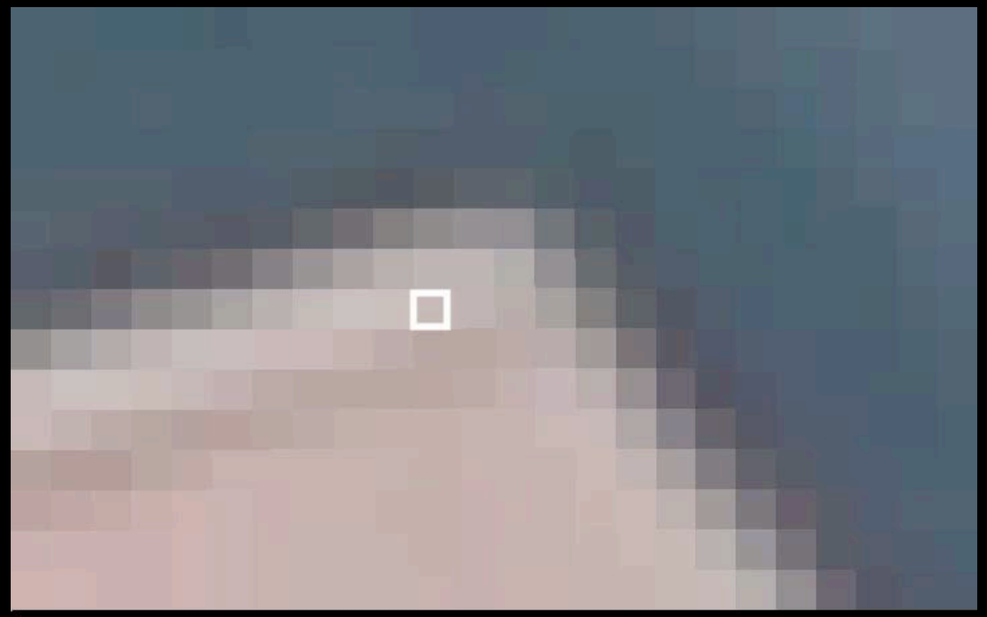


minV =

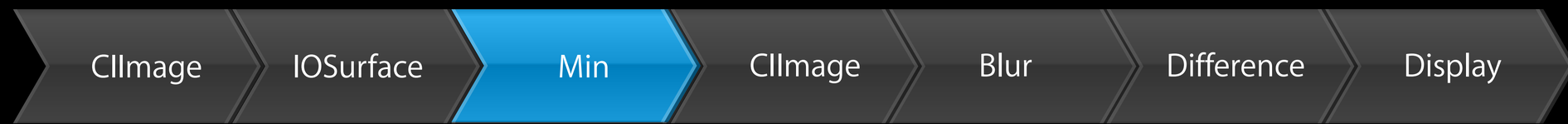


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

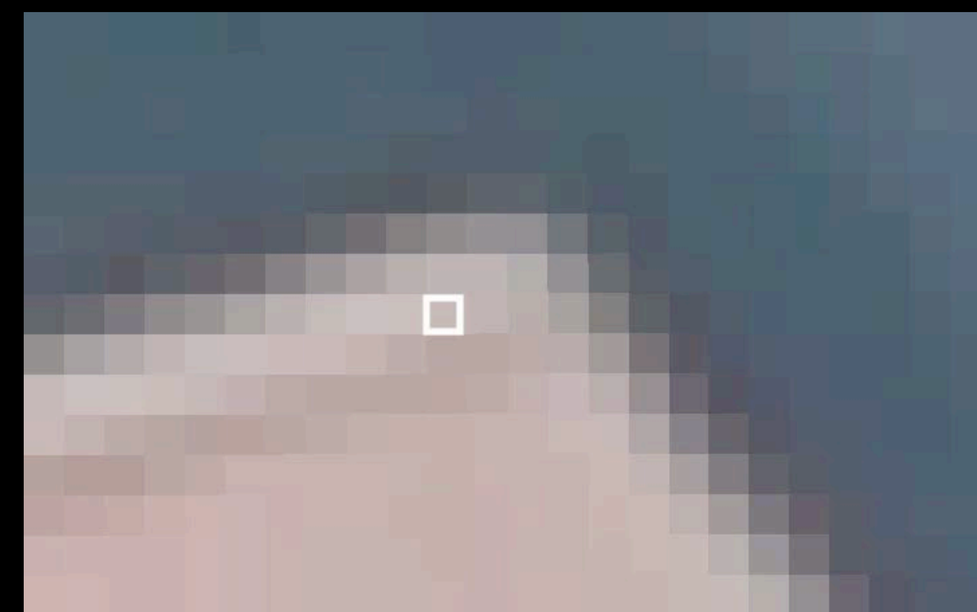


minV =

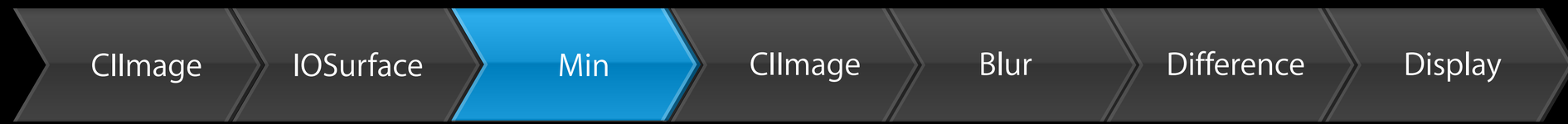


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

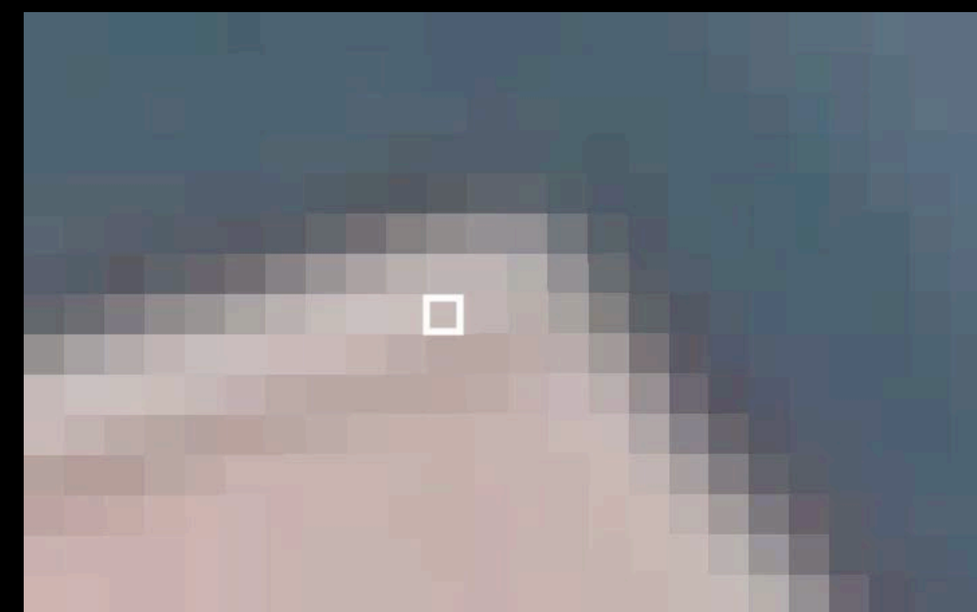


minV =

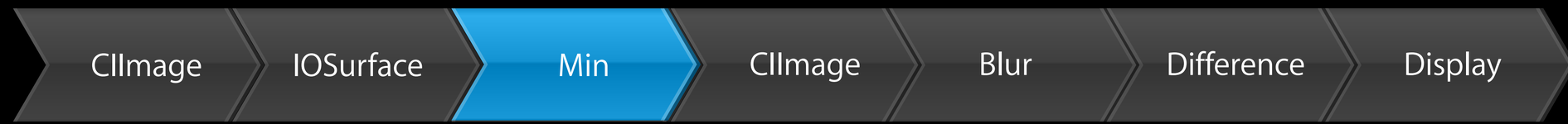


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

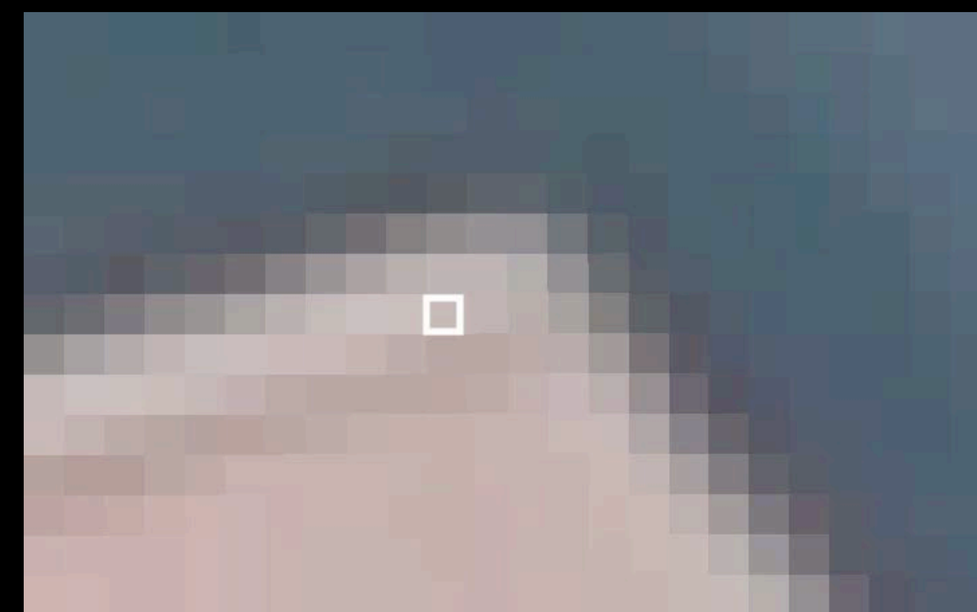


minV =

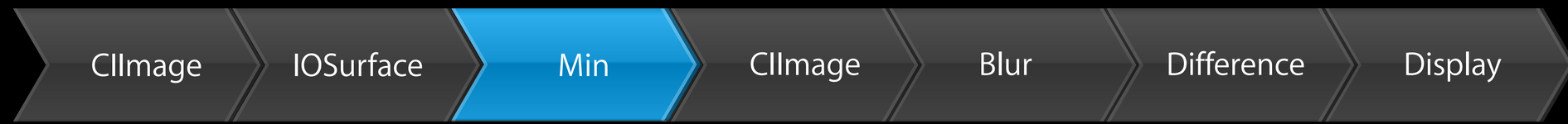


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

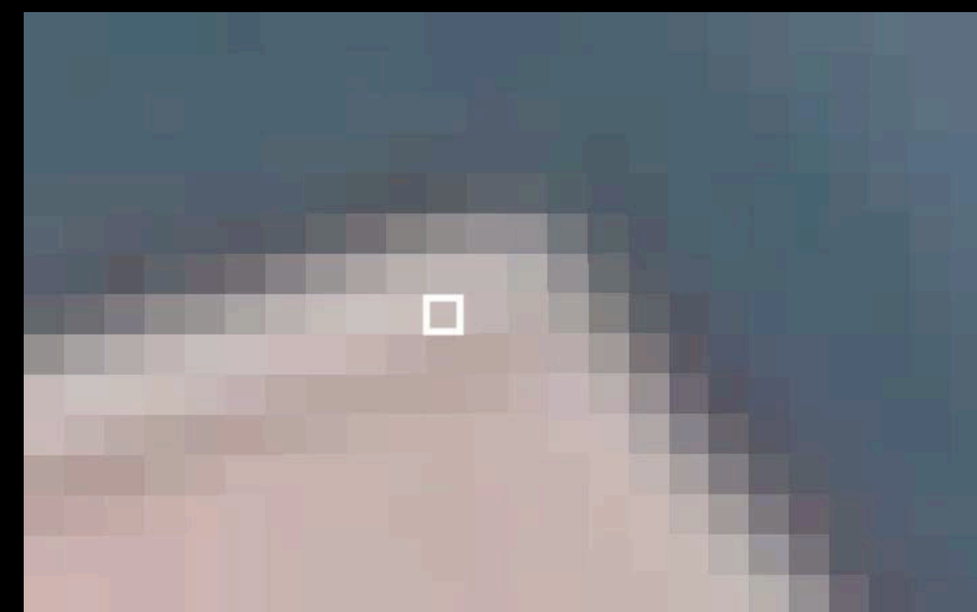


minV =

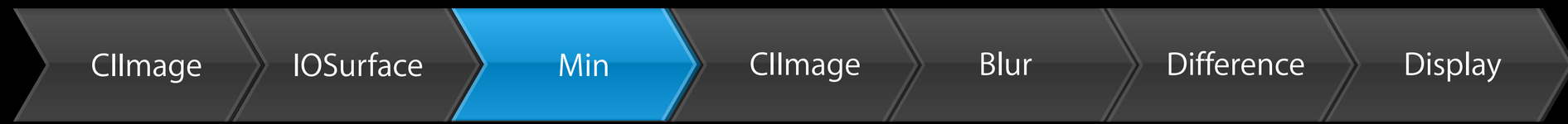


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

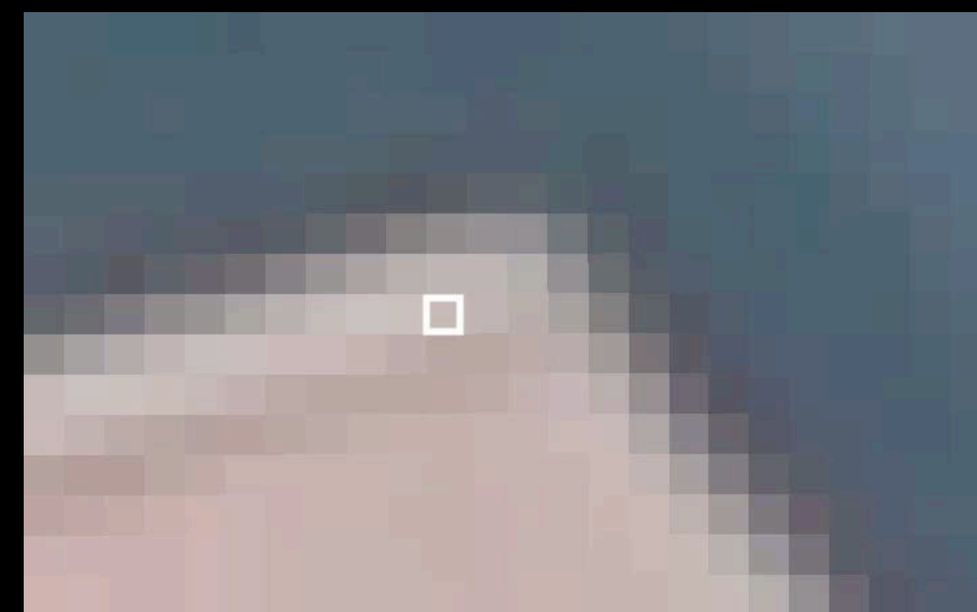


minV =

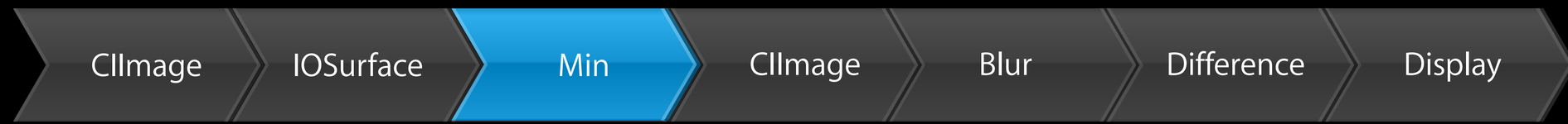


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

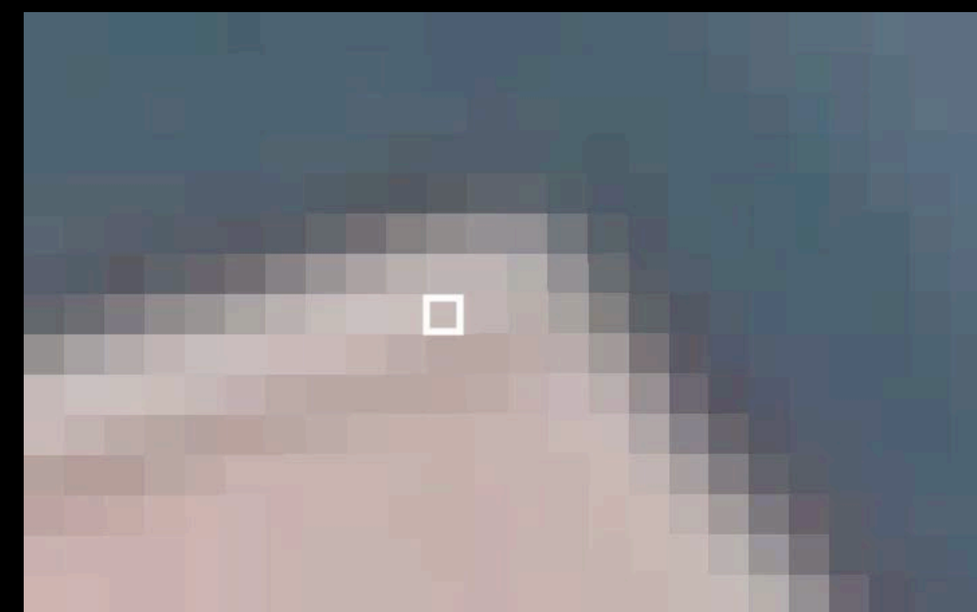


minV =

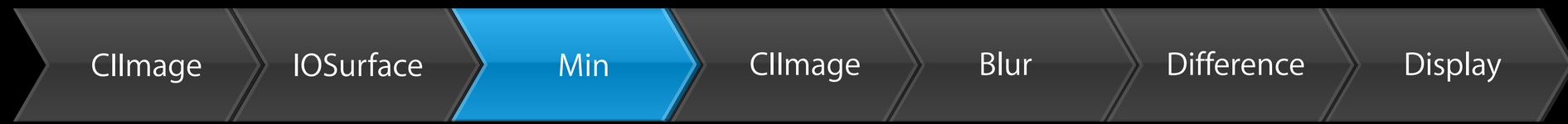


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

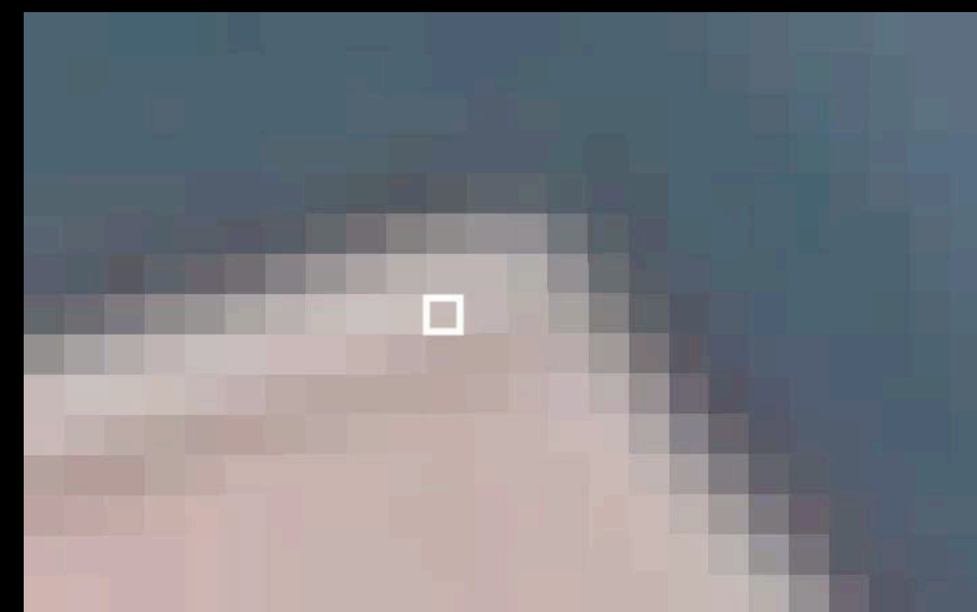


minV =

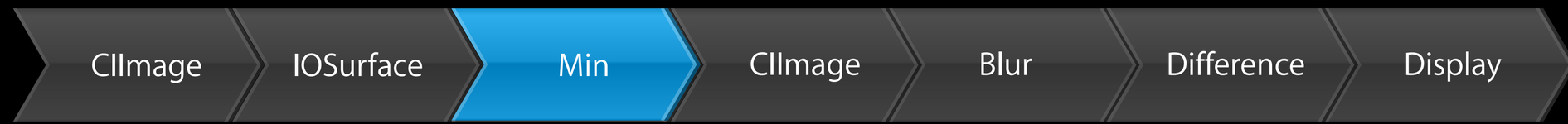


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

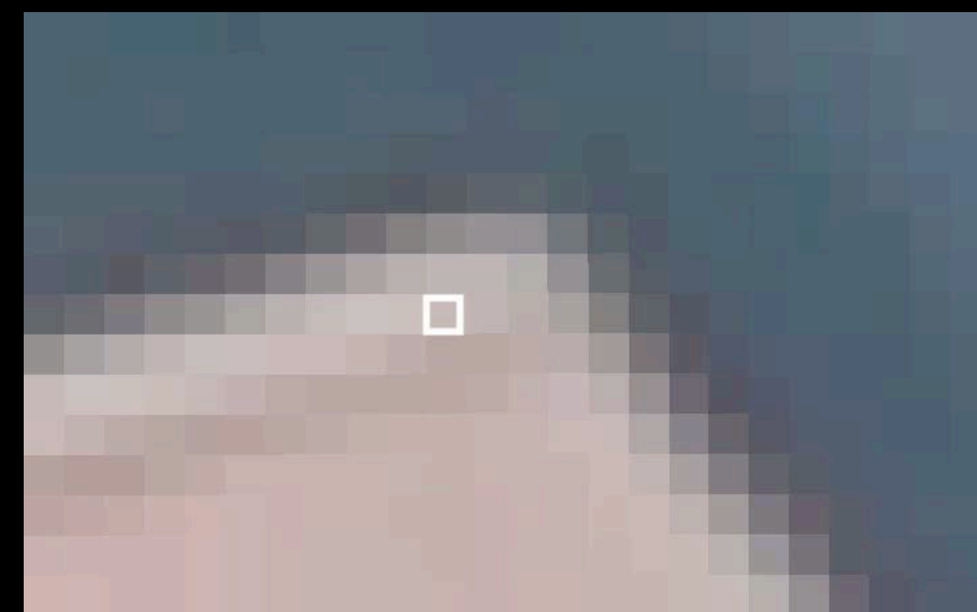


minV =

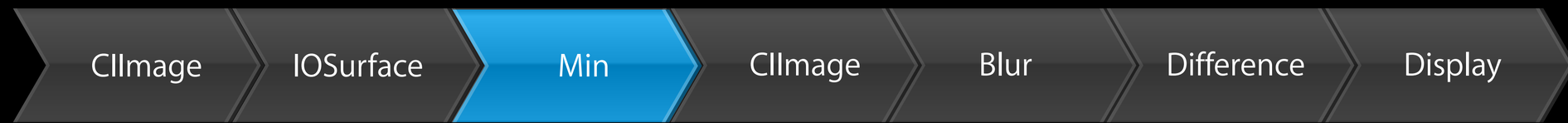


OpenCL: Kernel Code

```
__kernel void morphologicalMinX ( read_only image2d_t input , write_only image2d_t output , float span )
{
    const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
    CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;

    int2 loc = (int2)( get_global_id(0), get_global_id(1) );

    float4 minV = (float4)(1.0f);
    for ( int i=-(int)floor(span); i<=(int)ceil(span); i++ )
    {
      float2 readLoc = (float2)( loc.x + i + 0.5f, loc.y + 0.5f );
      float4 value = read_imagef ( input, sampler, readLoc );

      minV.xyz = min ( minV.xyz , value.xyz );
    }

    write_imagef ( output, loc, minV) ;
}
```

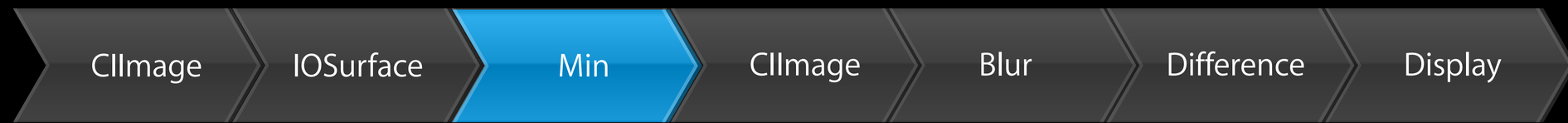


minV =



OpenCL: Create Program and Kernels

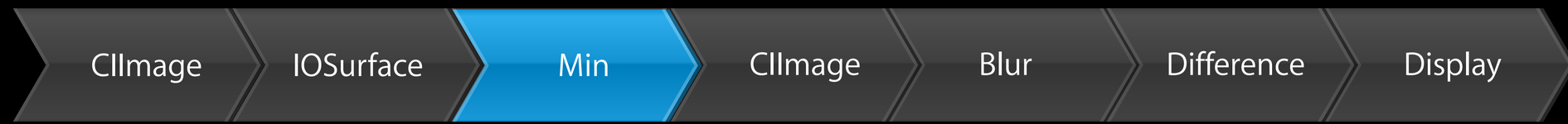
```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;

cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



OpenCL: Create Program and Kernels

```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;
```

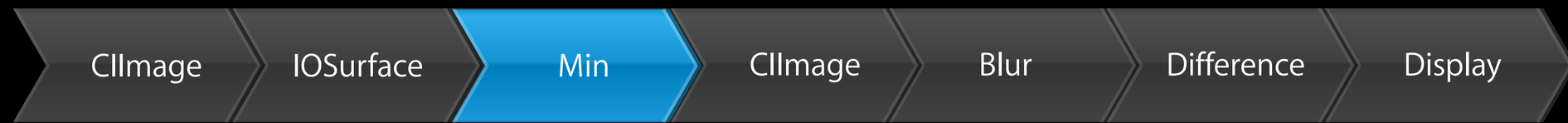
```
cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



OpenCL: Create Program and Kernels

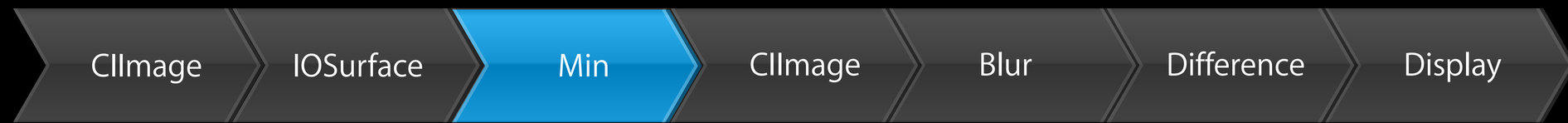
```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;
```

```
cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



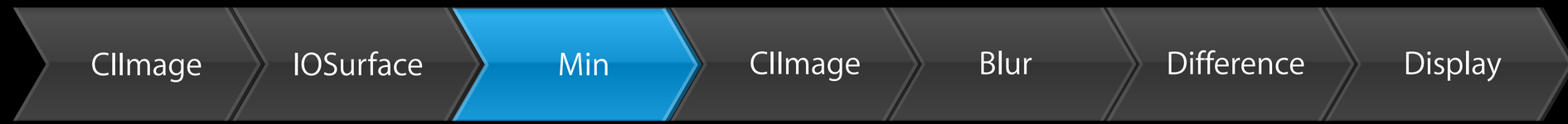
OpenCL: Create Program and Kernels

```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;

cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



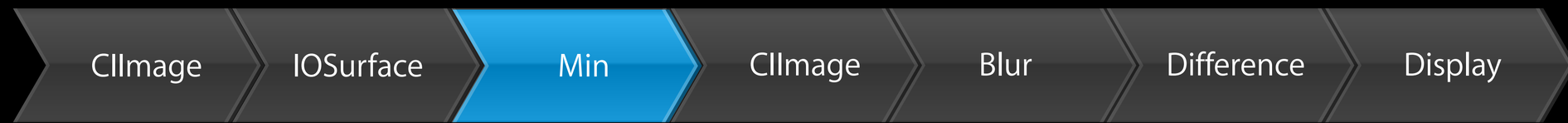
OpenCL: Create Program and Kernels

```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;

cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



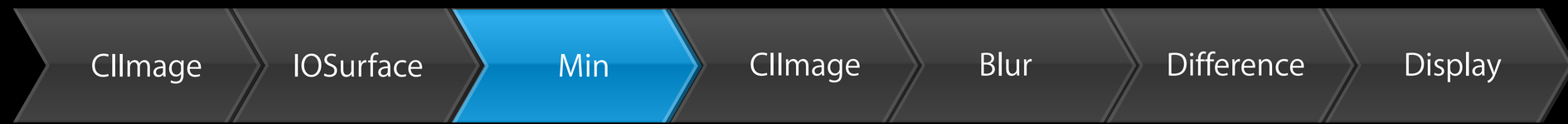
OpenCL: Create Program and Kernels

```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;

cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



OpenCL: Create Program and Kernels

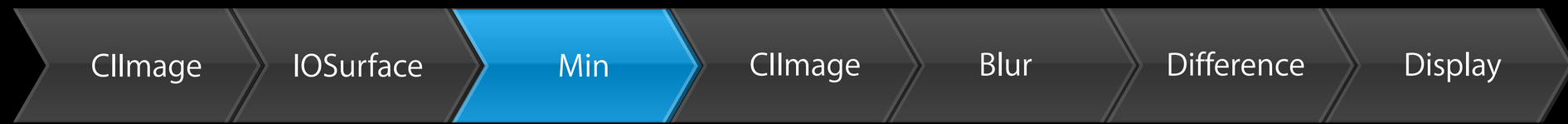
```
const char *code =
“kernel void morphologicalMinX( ... ) { ... code ... }\n”
“kernel void morphologicalMinY( ... ) { ... code ... }”;

cl_program program = clCreateProgramWithSource(clctx, 1, &code, lengths, &err);

clBuildProgram(program, n, devices, NULL, NULL, NULL);

cl_kernel minX_kernel = clCreateKernel ( program, “morphologicalMinX”, &err );

cl_kernel minY_kernel = clCreateKernel ( program, “morphologicalMinY”, &err );
```



OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

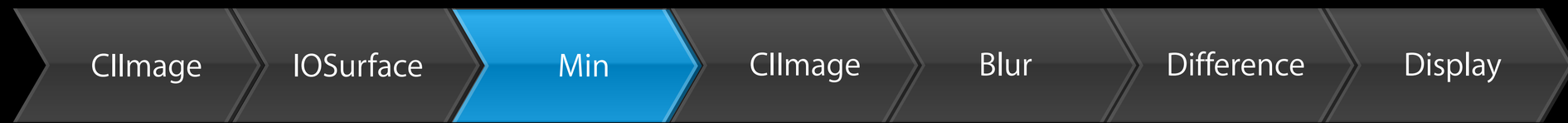
```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```



OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

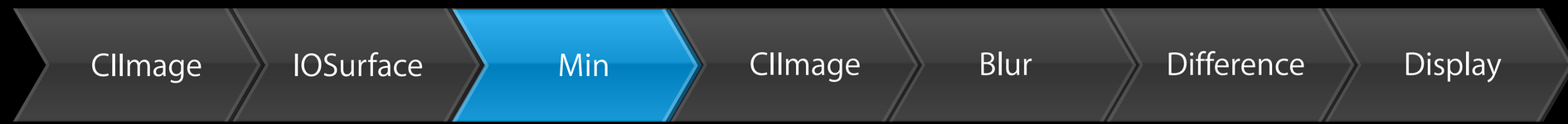
```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```



OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

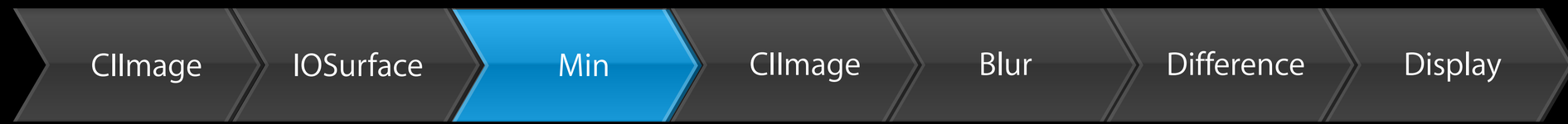
```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```



OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

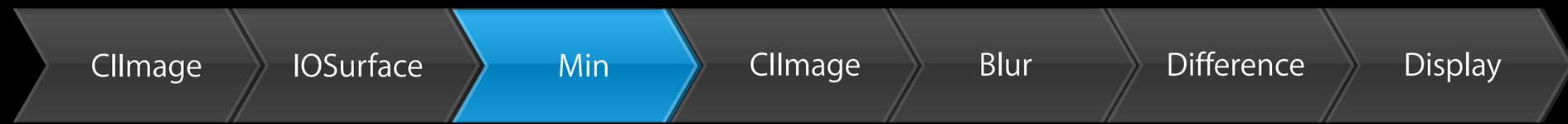
```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```



OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```



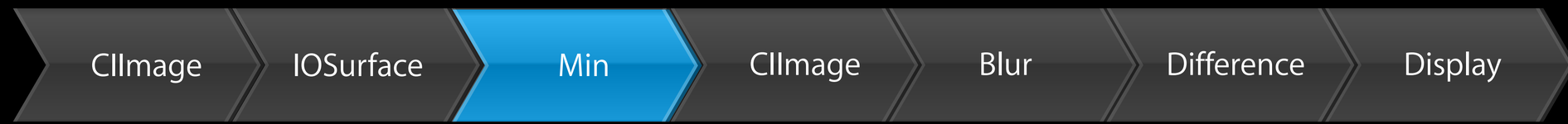
OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```

- Morphological Min Pass in Y direction

```
// set kernel args for Y pass then call EnqueueNDRange:
clEnqueueNDRangeKernel ( commandQueue, minY_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);

clFlush ( commandQueue );
```



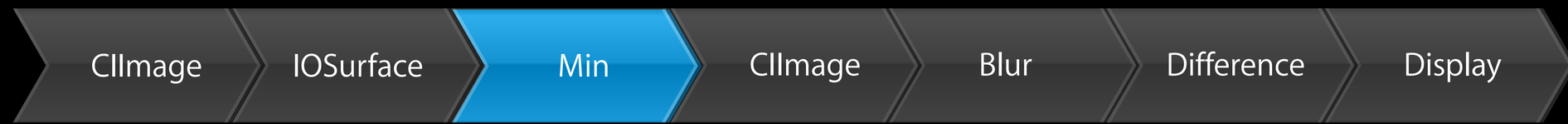
OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```

- Morphological Min Pass in Y direction

```
// set kernel args for Y pass then call EnqueueNDRange:
clEnqueueNDRangeKernel ( commandQueue, minY_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);

clFlush ( commandQueue );
```



OpenCL: Set Parameters and Run

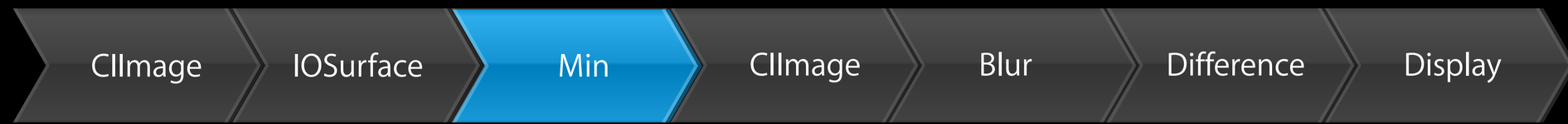
- Morphological Min Pass in X direction

```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```

- Morphological Min Pass in Y direction

```
// set kernel args for Y pass then call EnqueueNDRange:
clEnqueueNDRangeKernel ( commandQueue, minY_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```

```
clFlush ( commandQueue );
```



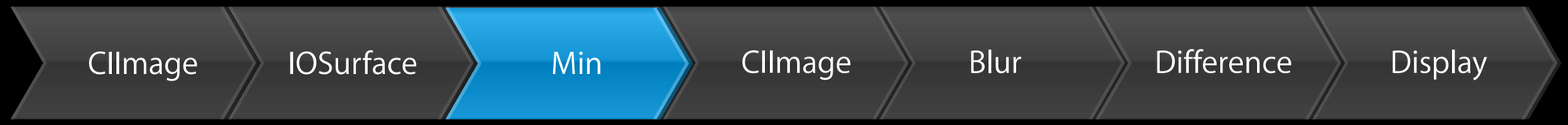
OpenCL: Set Parameters and Run

- Morphological Min Pass in X direction

```
clSetKernelArg ( minX_kernel, 0, sizeof(cl_mem), &input );
clSetKernelArg ( minX_kernel, 1, sizeof(cl_mem), &intermediateImage );
clSetKernelArg ( minX_kernel, 2, sizeof(float),  &spanX );
clEnqueueNDRangeKernel ( commandQueue, minX_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);
```

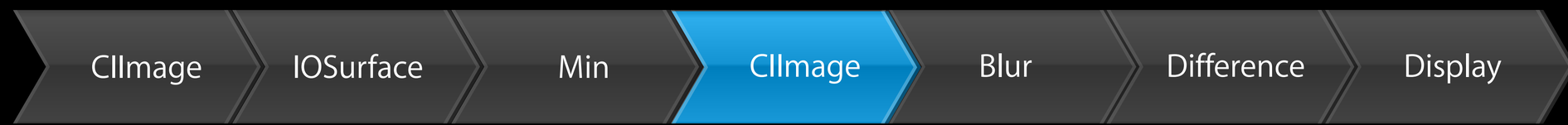
- Morphological Min Pass in Y direction

```
// set kernel args for Y pass then call EnqueueNDRange:
clEnqueueNDRangeKernel ( commandQueue, minY_kernel, 2, NULL,
                         execThreads, execLocal, 0, NULL, NULL);

clFlush ( commandQueue );
```



Create CIImage from Output IOSurface

```
CIImage *outputImage =
    [CIImage imageWithIOSurface : output
                        options : @{kCIImageColorSpace: (id)colorSpace} ];
```

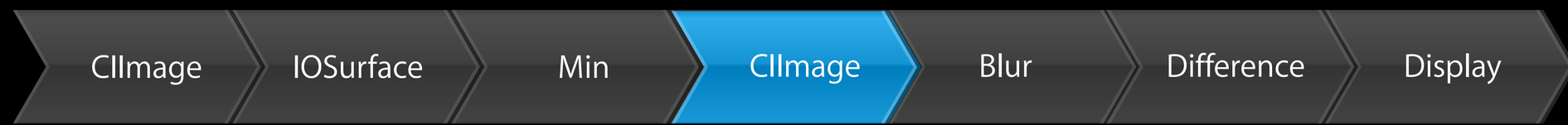


Create CIImage from Output IOSurface

```
CIImage *outputImage =
    [CIImage imageWithIOSurface : output
                        options : @{kCIImageColorSpace: (id)colorSpace} ];
```

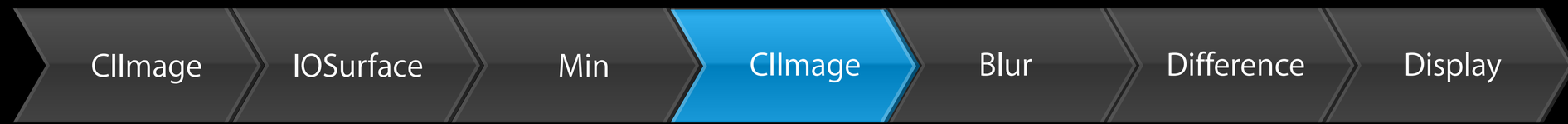
Create CIImage from Output IOSurface

```
CIImage *outputImage =
    [CIImage imageWithIOSurface : output
                        options : @{kCIImageColorSpace: (id)colorSpace} ];
```



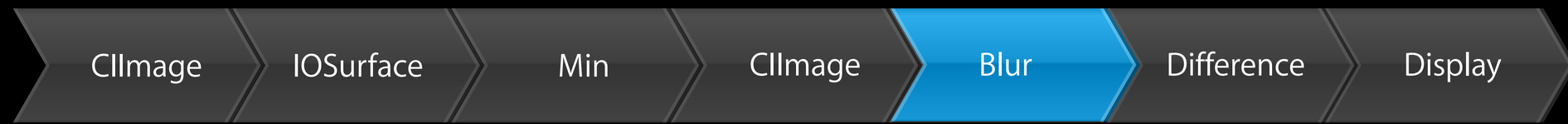
Create CIImage from Output IOSurface

```
CIImage *outputImage =
    [CIImage imageWithIOSurface : output
                        options : @{kCIImageColorSpace: (id)colorSpace} ];
```



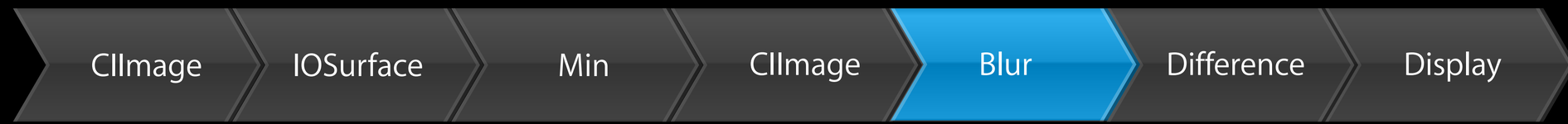
Blur Image

```
CIImage *clampedImage = [CIFilter filterWithName:@"CIAffineClamp", ...];

CIImage *blurredImage =
    [[CIFilter filterWithName:@"CIGaussianBlur"
        withKeysAndValues:@"inputImage", clampedImage, @"inputRadius",
        @(blurRadius), nil] valueForKey:@"outputImage"];

blurredImage = [blurredImage imageByCroppingToRect:[scaledImage extent]];
```



Blur Image

```
CIImage *clampedImage = [CIFilter filterWithName:@"CIAffineClamp", ...];

CIImage *blurredImage =
    [[CIFilter filterWithName:@"CIGaussianBlur"
        withKeysAndValues:@"inputImage", clampedImage, @"inputRadius",
        @(blurRadius), nil] valueForKey:@"outputImage"];

blurredImage = [blurredImage imageByCroppingToRect:[scaledImage extent]];
```

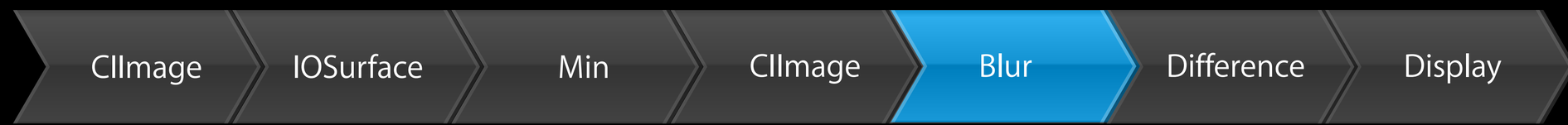


Blur Image

```
CIImage *clampedImage = [CIFilter filterWithName:@"CIAffineClamp", ...];
```

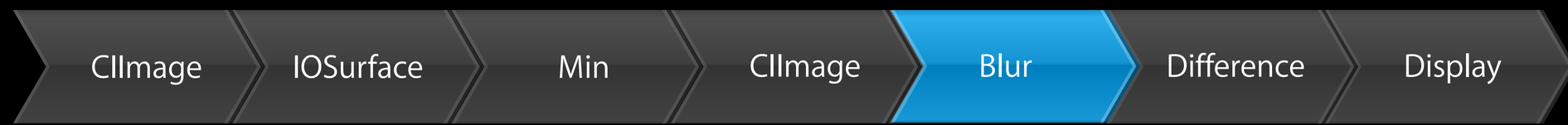
```
CIImage *blurredImage =
    [[CIFilter filterWithName:@"CIGaussianBlur"
        withKeysAndValues:@"inputImage", clampedImage, @"inputRadius",
        @(blurRadius), nil] valueForKey:@"outputImage"];
```

```
blurredImage = [blurredImage imageByCroppingToRect:[scaledImage extent]];
```



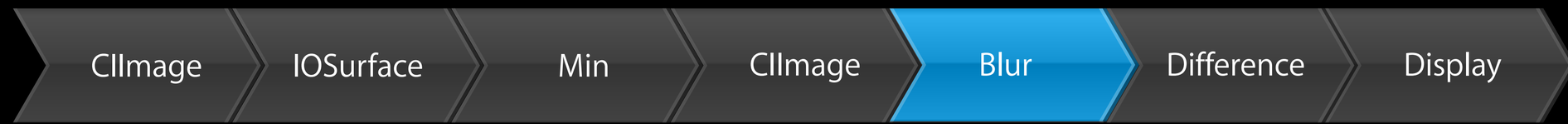
Blur Image

```
CIImage *clampedImage = [CIFilter filterWithName:@"CIAffineClamp", ...];

CIImage *blurredImage =
    [[CIFilter filterWithName:@"CIGaussianBlur"
        withKeysAndValues:@"inputImage", clampedImage, @"inputRadius",
        @(blurRadius), nil] valueForKey:@"outputImage"];

blurredImage = [blurredImage imageByCroppingToRect:[scaledImage extent]];
```



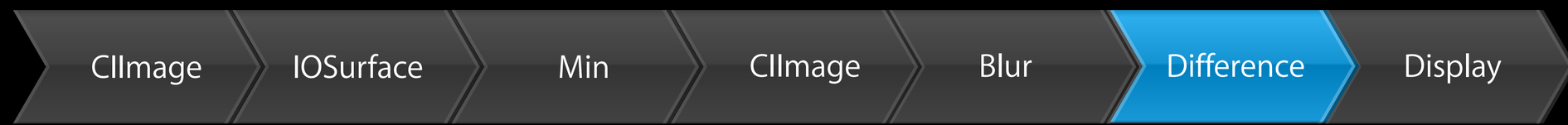
Blur Image

```
CIImage *clampedImage = [CIFilter filterWithName:@"CIAffineClamp", ...];

CIImage *blurredImage =
    [[CIFilter filterWithName:@"CIGaussianBlur"
        withKeysAndValues:@"inputImage", clampedImage, @"inputRadius",
        @(blurRadius), nil] valueForKey:@"outputImage"];

blurredImage = [blurredImage imageByCroppingToRect:[scaledImage extent]];
```



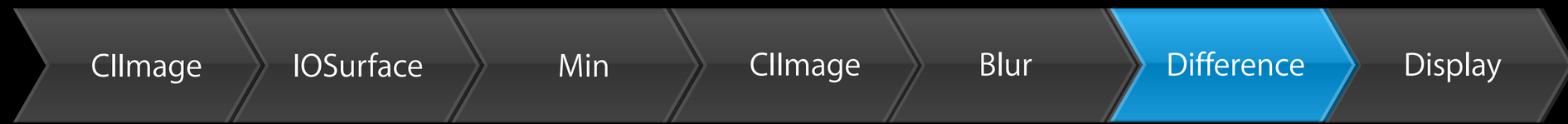
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



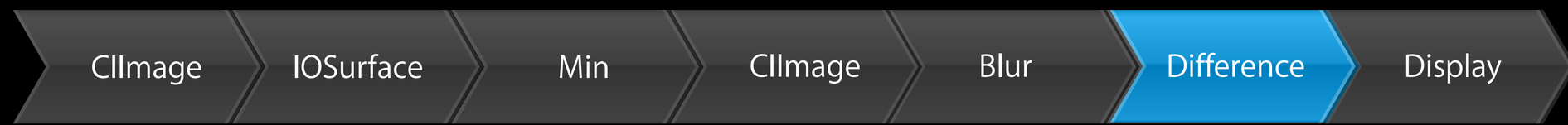
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



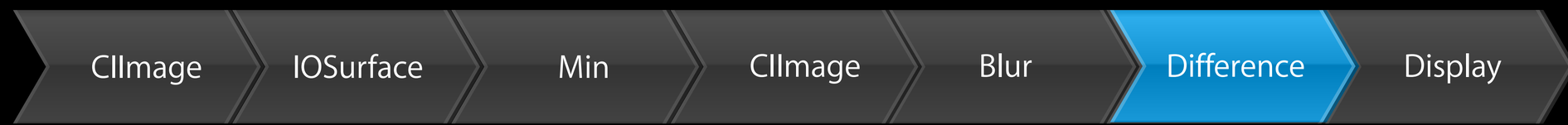
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



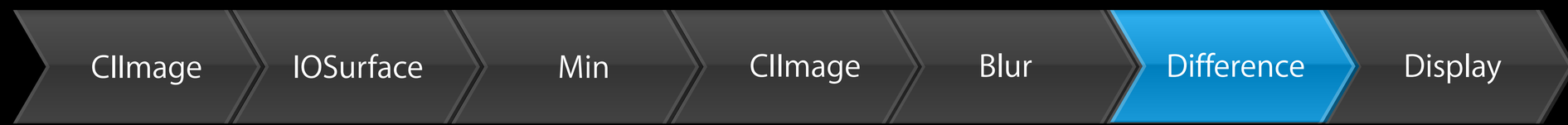
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



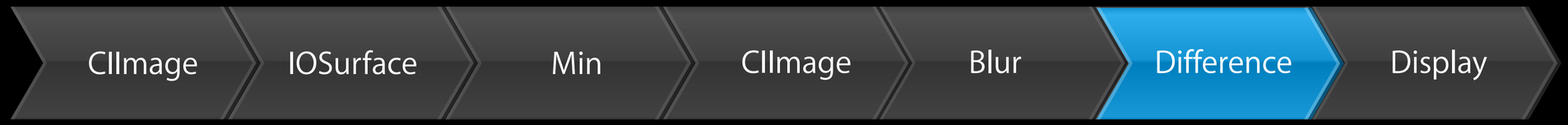
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



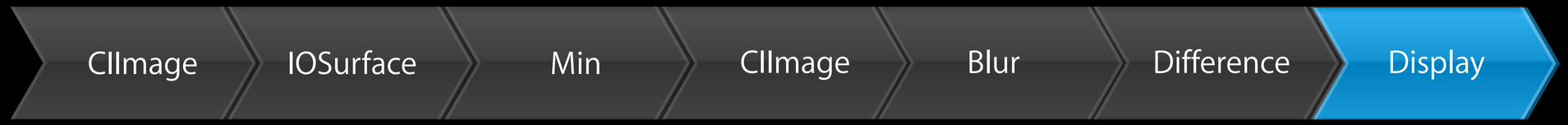
Apply Difference Blending

```
CIFilter *filter = [CIFilter filterWithName:@"CIDifferenceBlendMode"];

[filter setValue : scaledImage forKey : @"inputImage"];

[filter setValue : blurredImage forKey : @"inputBackgroundImage"];

CIImage *finalImage = [filter valueForKey : @"outputImage"];
```



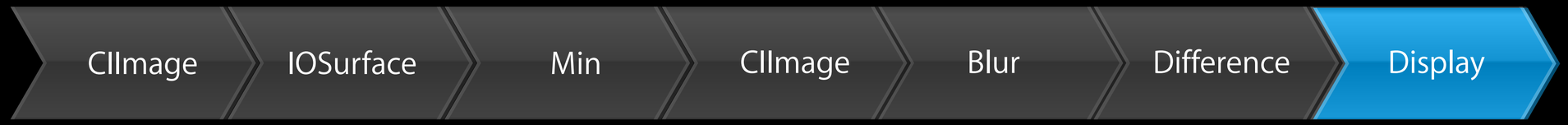
Display

```
[context drawImage : finalImage
            inRect : destRect // rendering destination rect
          fromRect : [finalImage extent]];
```



Display

```
[context drawImage : finalImage
            inRect : destRect // rendering destination rect
          fromRect : [finalImage extent]];
```



Display

```
[context drawImage : finalImage
            inRect : destRect // rendering destination rect
          fromRect : [finalImage extent]];
```

Debugging

```
140     bool allocatedColorSpace = false;
141     if ( ! colorSpace ) {
142         colorSpace = CGColorSpaceCreateDeviceRGB();
143         allocatedColorSpace = true;
144     }
145
146     [context render:inputImage toIOSurface:otherSurface bounds:[inputImage extent] colorSpace:colorSpace];
147
148     size_t width = IOSurfaceGetWidth(otherSurface);
149     size_t height = IOSurfaceGetHeight(otherSurface);
150
151     const float spanX = width / 15.0;
152     const float spanY = height / 256.0;
153     NSNumber *blurRadius = [NSNumber numberWithFloat:width / 20 ];
154
155     [(Dehaze *)cih removeHazeFromImage:otherSurface outputSurface:newOutputSurface spanX:spanX spanY:spanY colorSpace:colorSpace];
156
157
158     if ( otherSurface && newOutputSurface  ) {
159         CIImage *newImage = [CIImage im                                  : (id)colorSpace}];
160
161         ▶ (CIImage *) 0x1001226f0                                        ", newImage, @"inputTransform",
162         CIImage *blurredImage = [[[CIFi                                  ius", blurRadius, @"inputImage", clampe
            kCIOutputImageKey] imageBy
163         CIImage *outputImage = [[CIFilter filterWithName:@"CIDifferenceBlendMode" keysAndValues:@"inputImage", inputImage, @"inputBackgroun
            valueForKey:@"outputImage"];
164
165         [context drawImage:outputImage inRect:rect fromRect:[outputImage extent]];
166     }
167
168     if ( allocatedColorSpace )
169         CGColorSpaceRelease (colorSpace);
170
171     if ( newOutputSurface )
172         CFRelease(newOutputSurface);
```

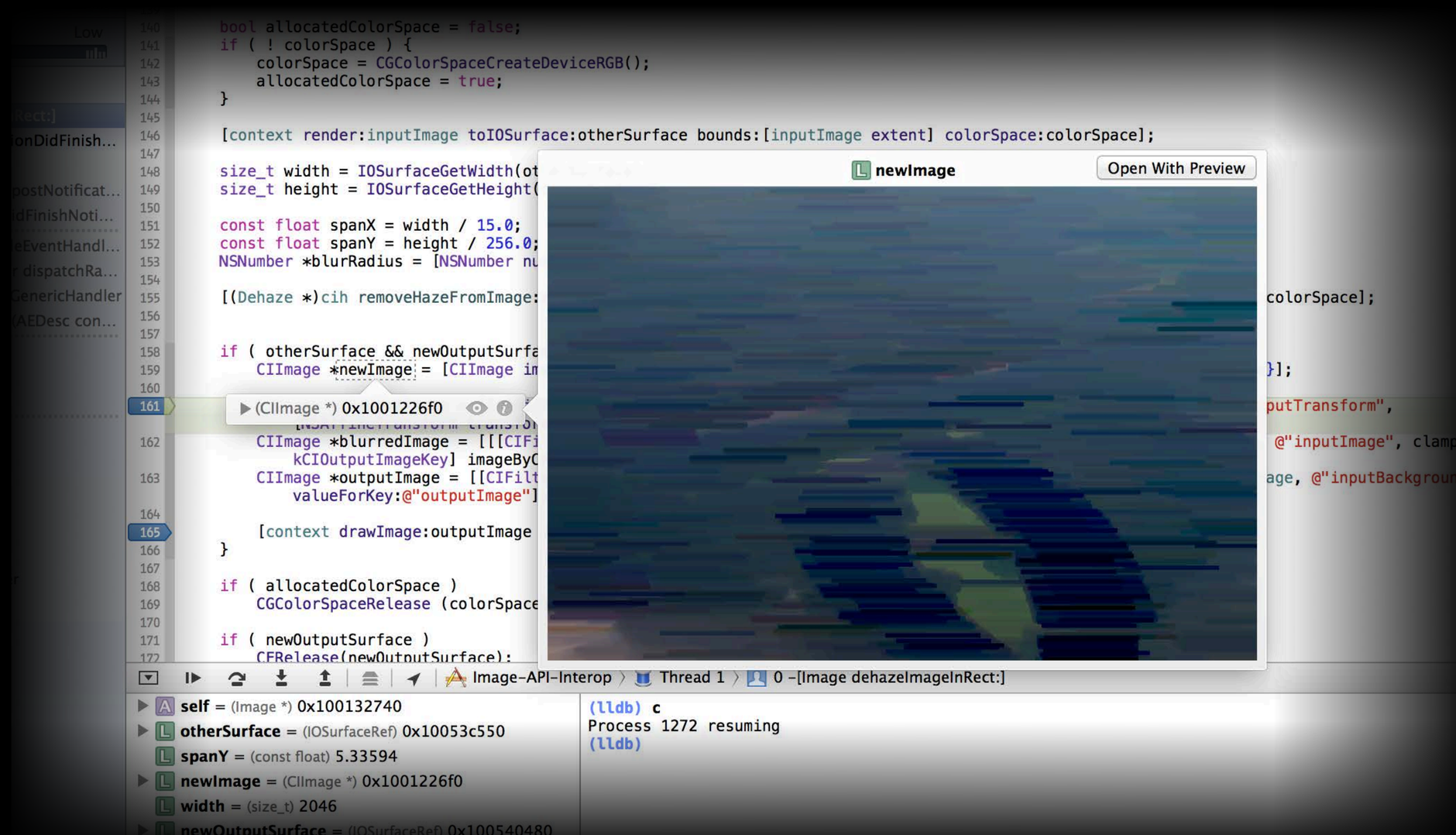
```
<CIImage: 0x1001226f0> {
    FEIOSurfaceImage: 0x100142b20 extent: [0 0 2046 1366];
format: BGRA_8; uid 2
}
```

Image-API-Interop 〉 Thread 1 〉 0 -[Image dehazeImageInRect:]

```
self = (Image *) 0x100132740
otherSurface = (IOSurfaceRef) 0x10053c550
spanY = (const float) 5.33594
newImage = (CIImage *) 0x1001226f0
width = (size_t) 2046
newOutputSurface = (IOSurfaceRef) 0x100540480
```

```
(lldb) c
Process 1272 resuming
Printing description of newImage:
<CIImage: 0x1001226f0> {
    FEIOSurfaceImage: 0x100142b20 extent: [0 0 2046 1366]; format: BGRA_8; uid 2
}
(lldb)
```

Debugging


bool allocatedColorSpace = false;
if (! colorSpace) {
colorSpace = CGColorSpaceCreateDeviceRGB();
allocatedColorSpace = true;
}
[context render:inputImage toIOSurface:otherSurface bounds:[inputImage extent] colorSpace:colorSpace];
size_t width = IOSurfaceGetWidth(ot
size_t height = IOSurfaceGetHeight(
const float spanX = width / 15.0;
const float spanY = height / 256.0;
NSNumber *blurRadius = [NSNumber nu
[(Dehaze *)cih removeHazeFromImage:
if (otherSurface && newOutputSurfa
CIImage *newImage = [CIImage im
(CIImage *) 0x1001226f0
CIImage *blurredImage = [[[CIFi
kCIOutputImageKey] imageByC
CIImage *outputImage = [[CIFilt
valueForKey:@"outputImage"]
[context drawImage:outputImage
}
if (allocatedColorSpace)
CGColorSpaceRelease (colorSpace
if (newOutputSurface)
newImage
Open With Preview
Image-API-Interop
Thread 1
0 -[Image dehazeImageInRect:]
self = (Image *) 0x100132740
otherSurface = (IOSurfaceRef) 0x10053c550
spanY = (const float) 5.33594
newImage = (CIImage *) 0x1001226f0
width = (size_t) 2046
newOutputSurface = (IOSurfaceRef) 0x100540480
(lldb) c
Process 1272 resuming
(lldb)

Once More in Action

Once More in Action

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Documentation
Core Image Programming Guide
http://developer.apple.com/library/ios/#documentation/GraphicsImaging/Conceptual/CoreImaging

Apple Developer Forums
http://devforums.apple.com

Related Labs and Sessions

Core Image Lab	Graphics and Games Lab B Friday 11:30AM
Introduction to Sprite Kit	Presidio Wednesday 11:30AM
Advanced Editing with AV Foundation	Marina Thursday 9:00AM
Working with OpenCL	Marina Thursday 3:15PM

WWDC2013