#WWDC18

Core Image: Performance, Prototyping, and Python

Session 719

David Hayward, Core Image
Emmanuel Piuze-Phaneuf, Core Image

Core Image Performance APIs

Prototyping with Core Image

Core Image and Machine Learning

Core Image Performance APIs

Prototyping with Core Image

Core Image and Machine Learning

Core Image Performance APIs

Prototyping with Core Image

Core Image and Machine Learning

Core Image Performance APIs

Prototyping with Core Image

Core Image and Machine Learning

Core Image Performance APIs

Core Image Performance APIs

NEW

Core Image Performance APIs

NEW

Inserting intermediates

Core Image Performance APIs

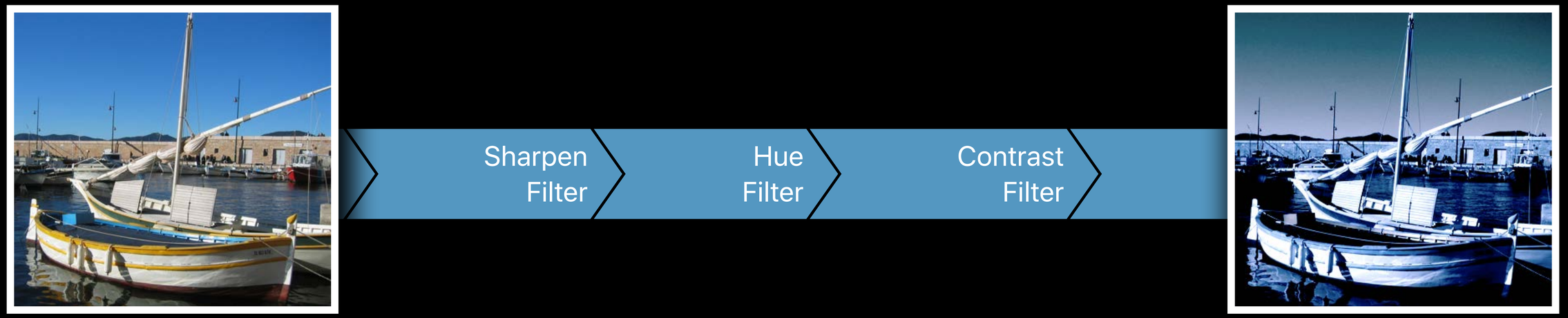
NEW

Inserting intermediates

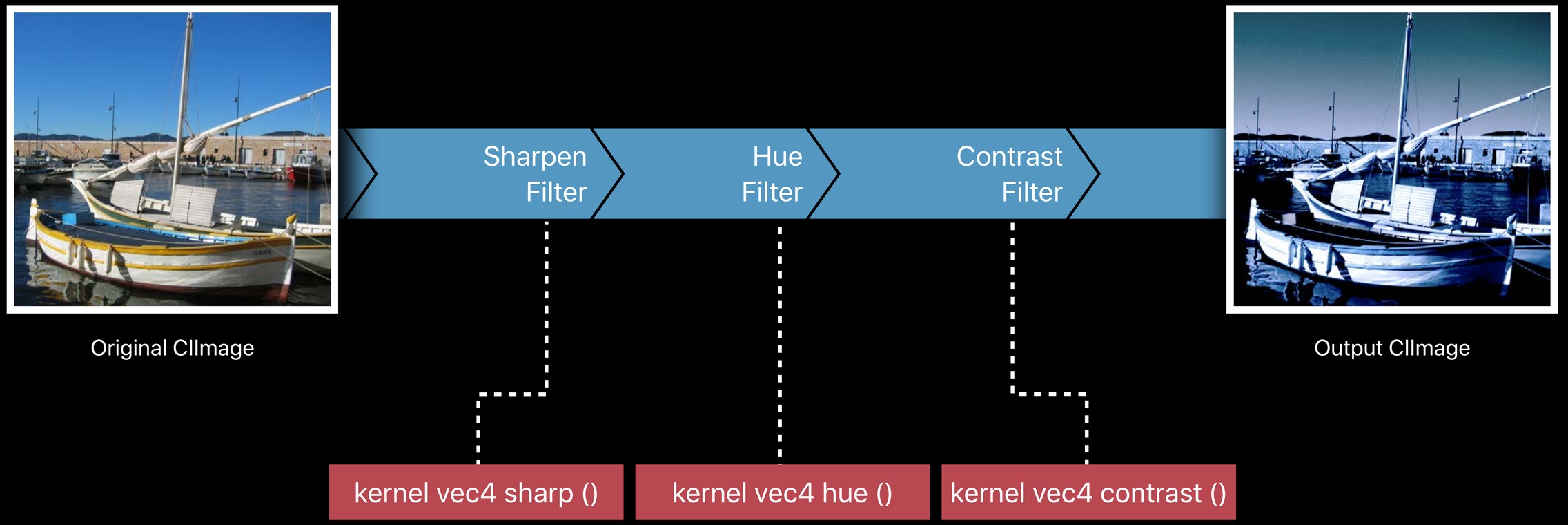
Kernel language features

Inserting Intermediates

Inserting Intermediate into Your Filter Graph



Each CIFilter Has One or More CIKernel Functions



CIKernels Are Concatenated for Performance

Original CIImage

Concatenated Program

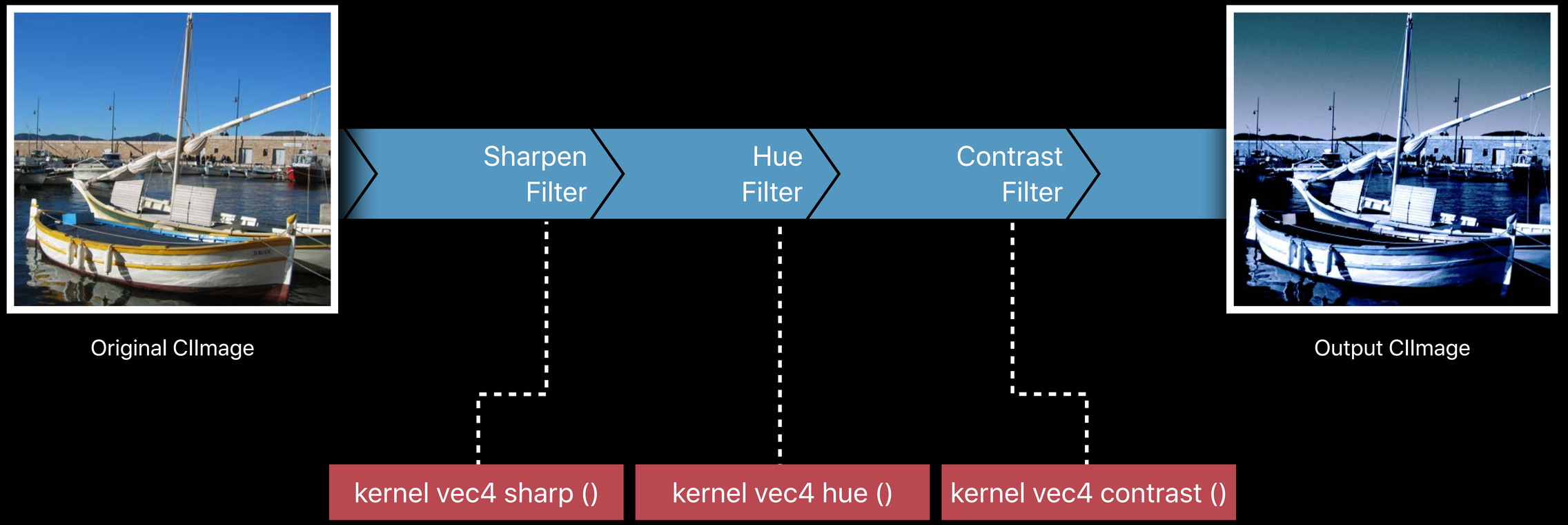
kernel vec4 sharp ()

kernel vec4 hue ()

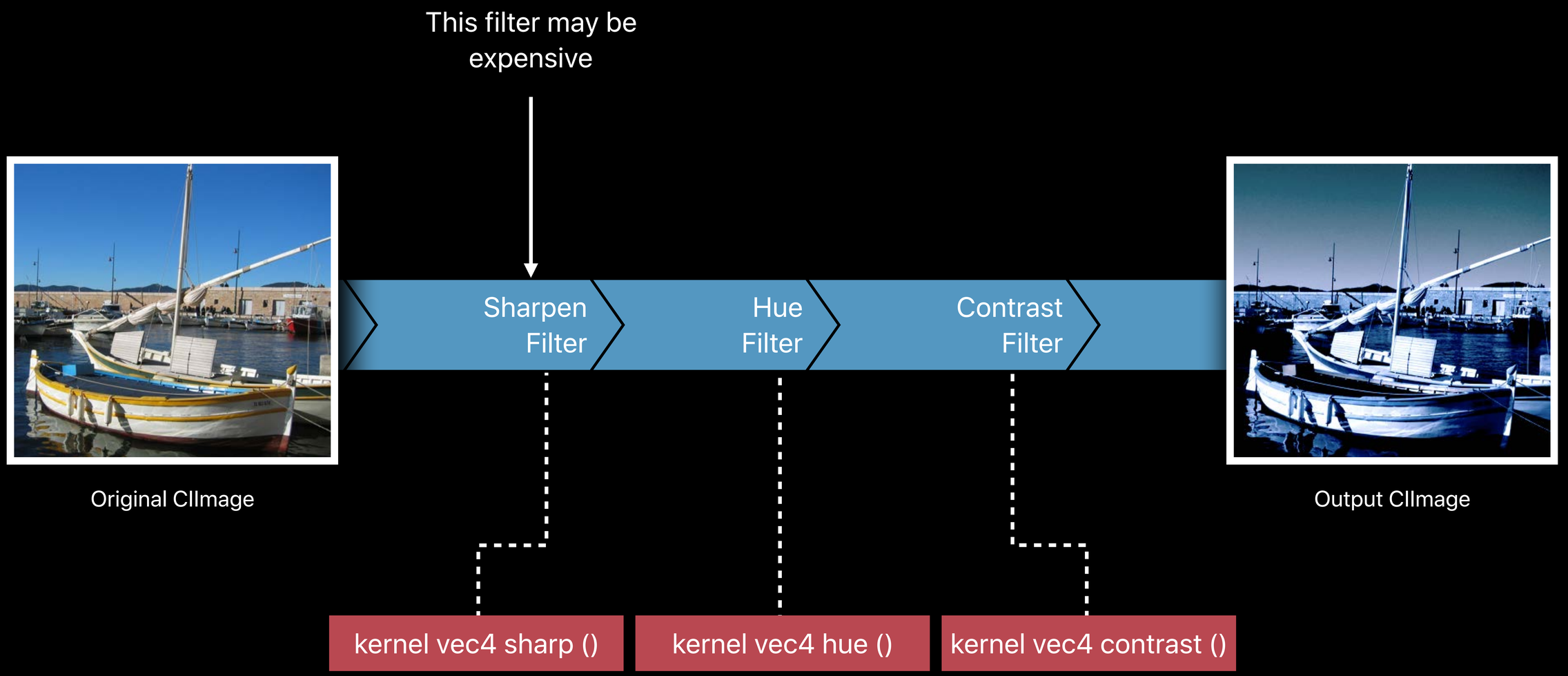
kernel vec4 contrast ()

Output CIImage

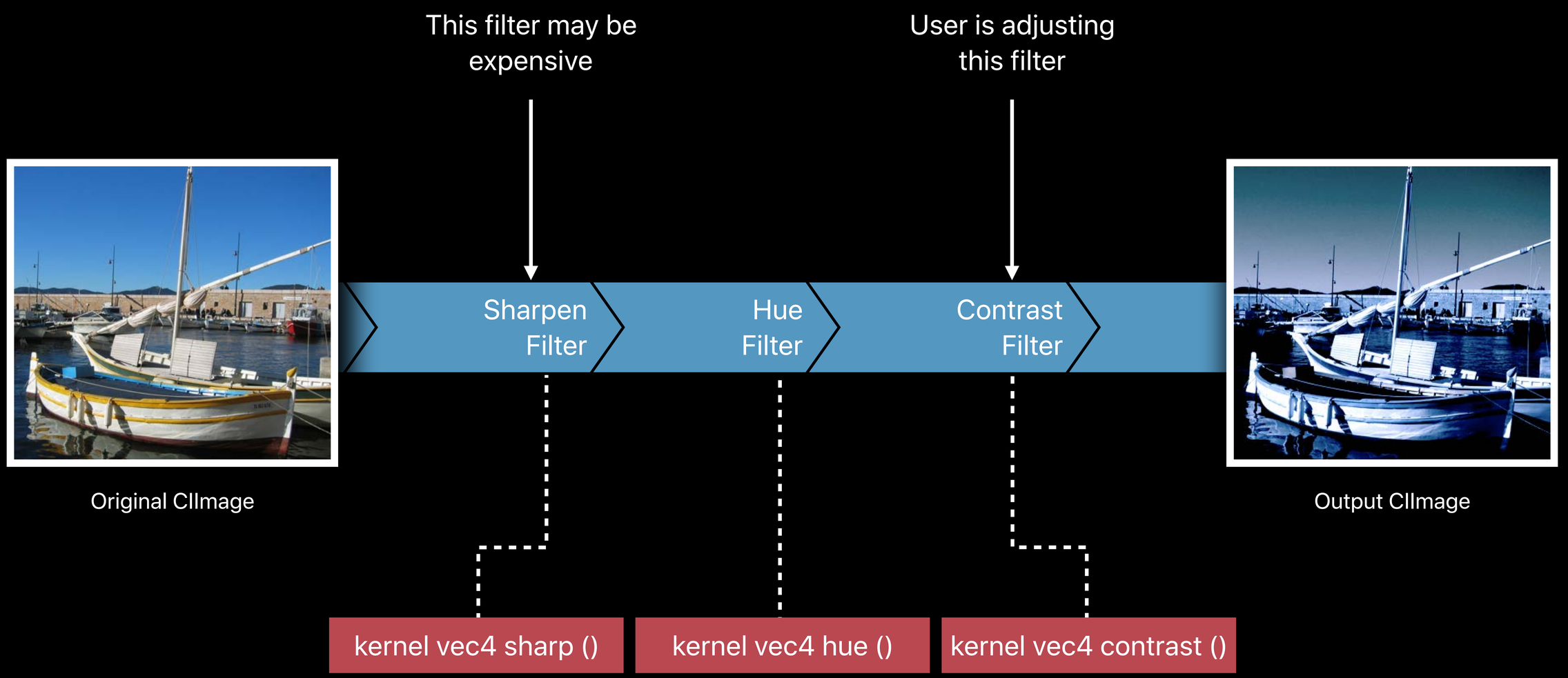
There Are Times When an App Knows Better



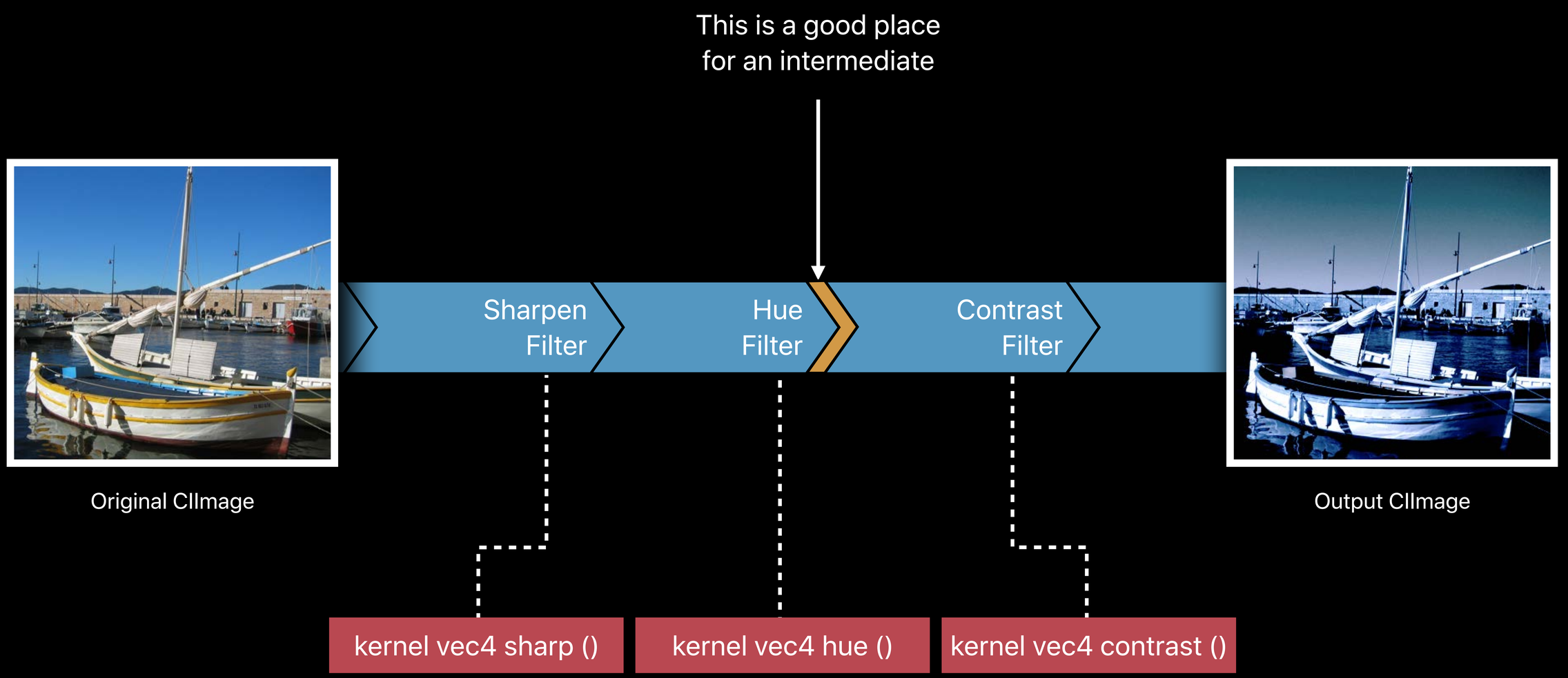
There Are Times When an App Knows Better



There Are Times When an App Knows Better



There Are Times When an App Knows Better



There Are Times When an App Knows Better

NEW

This is a good place
for an intermediate

```
image.insertingIntermediate()
```

Original CIImage

Sharpen Filter

Hue Filter

Contrast Filter

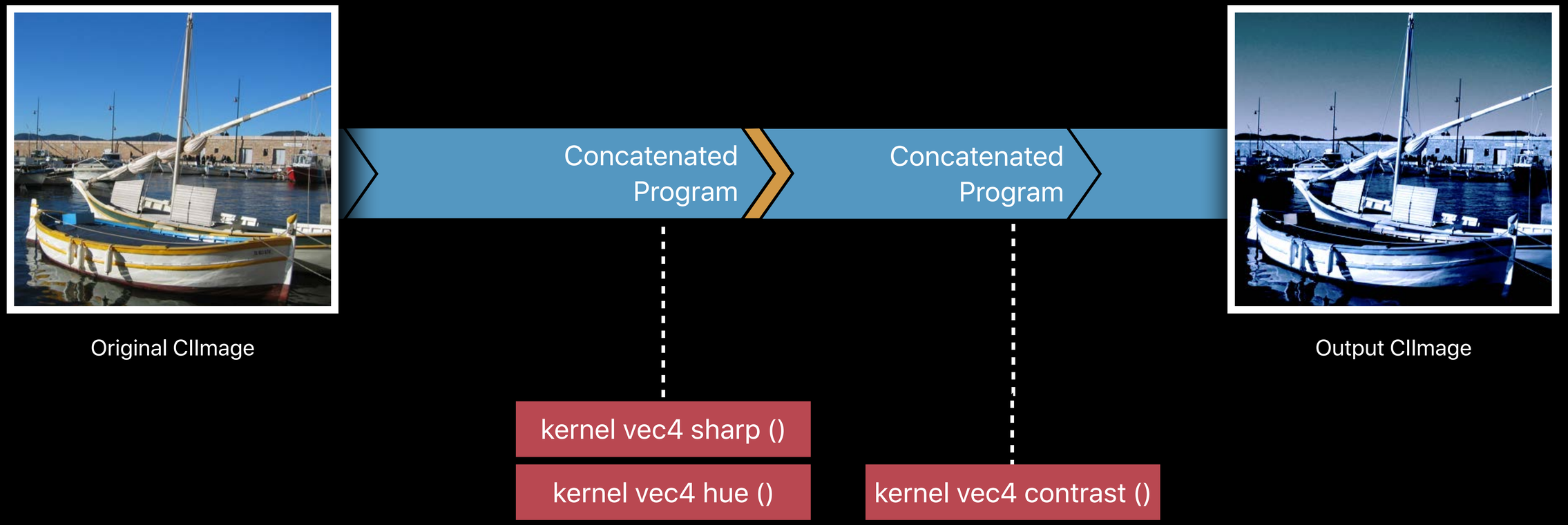
Output CIImage

kernel vec4 sharp ()

kernel vec4 hue ()

kernel vec4 contrast ()

There Are Times When an App Knows Better



Caching of Intermediates

Caching of Intermediates

Intermediates are cached by default

Caching of Intermediates

Intermediates are cached by default

Can turn off for all renders on a CIContext

```
let context = CIContext(options: [.cacheIntermediates: false] );
```

Caching of Intermediates

Intermediates are cached by default

Can turn off for all renders on a CIContext

```
let context = CIContext(options: [.cacheIntermediates: false] );
```

Or turn on for specific intermediates

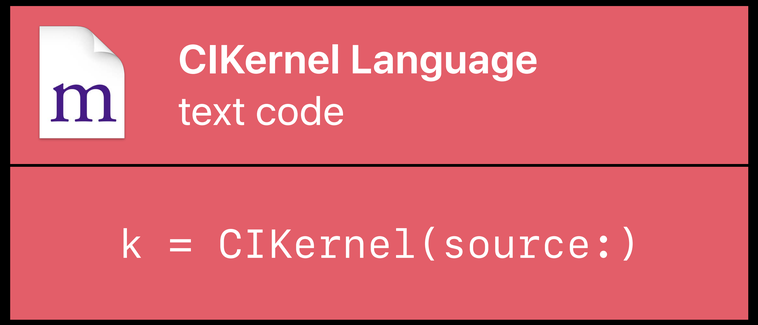
```
image.insertingIntermediate(cache: true);
```

New Kernel Language Features

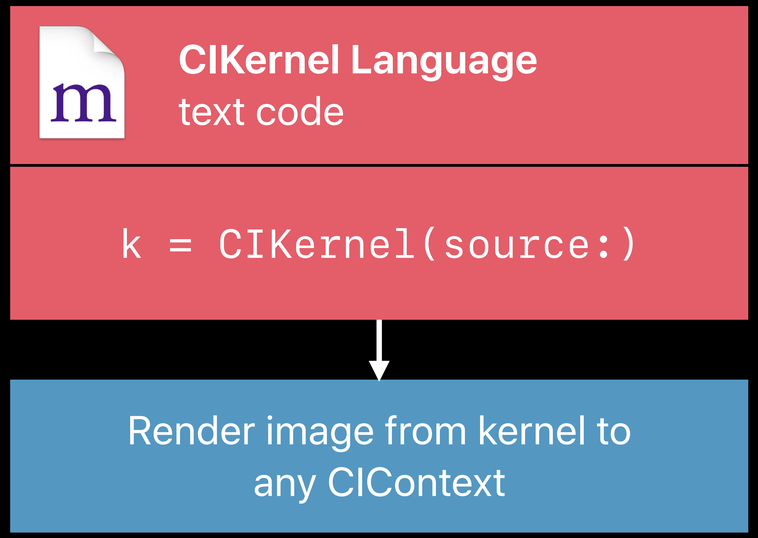
Two Ways To Write CIKernels

Two Ways To Write CIKernels

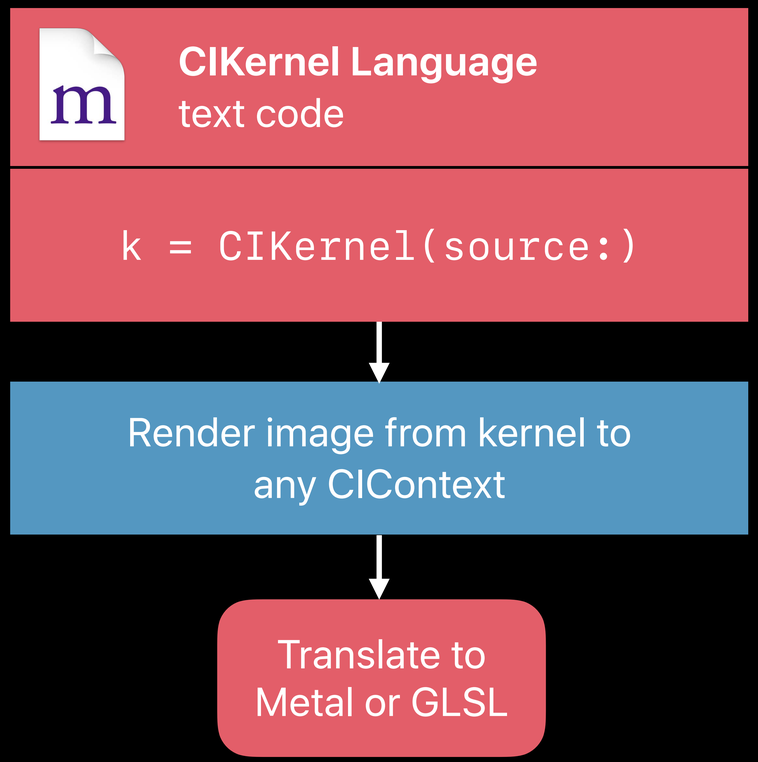
Two Ways To Write CIKernels



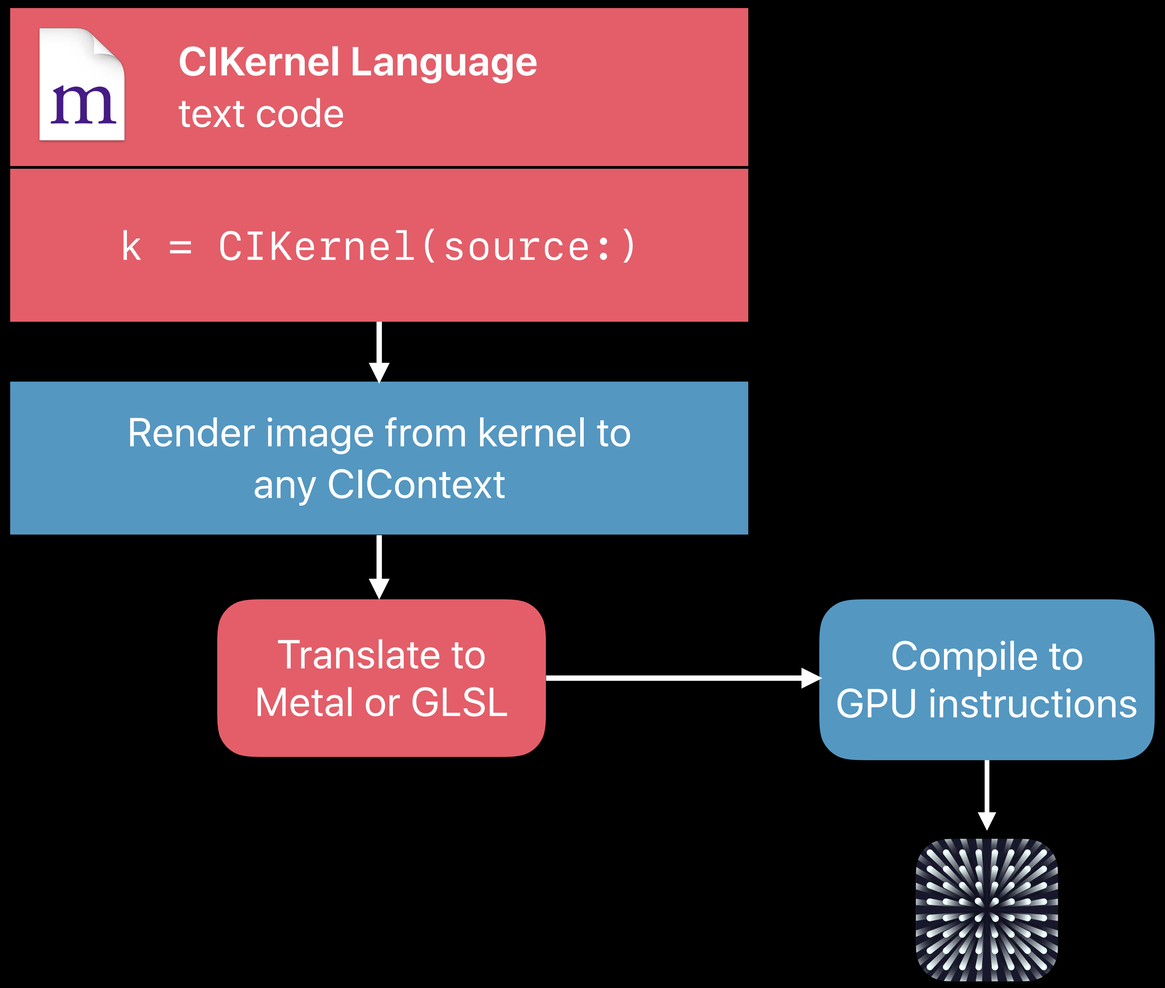
Two Ways To Write CIKernels



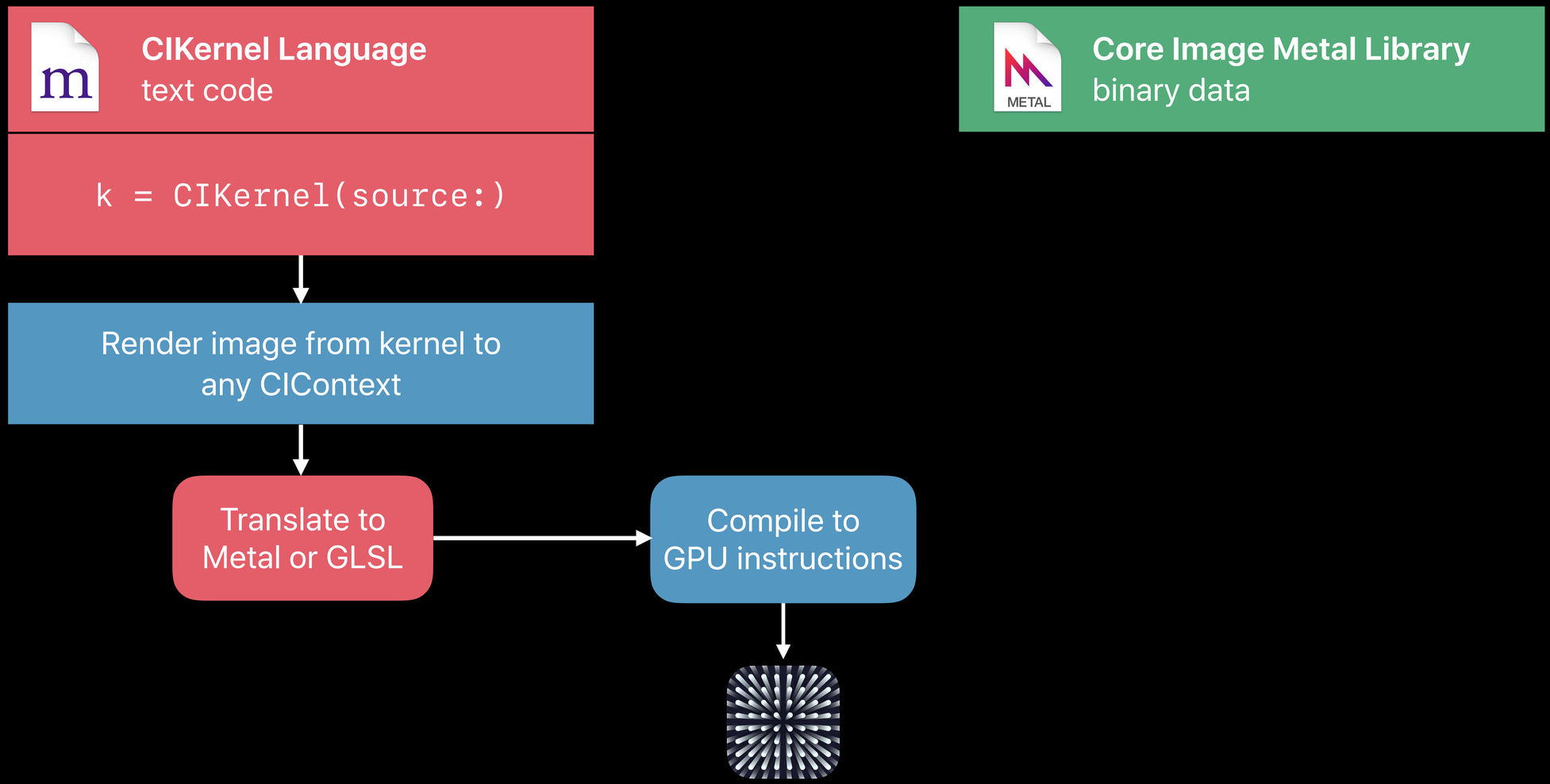
Two Ways To Write CIKernels



Two Ways To Write CIKernels



Two Ways To Write CIKernels



Two Ways To Write CIKernels

CIKernel Language
text code

```
k = CIKernel(source:)
```

Render image from kernel to any CIContext

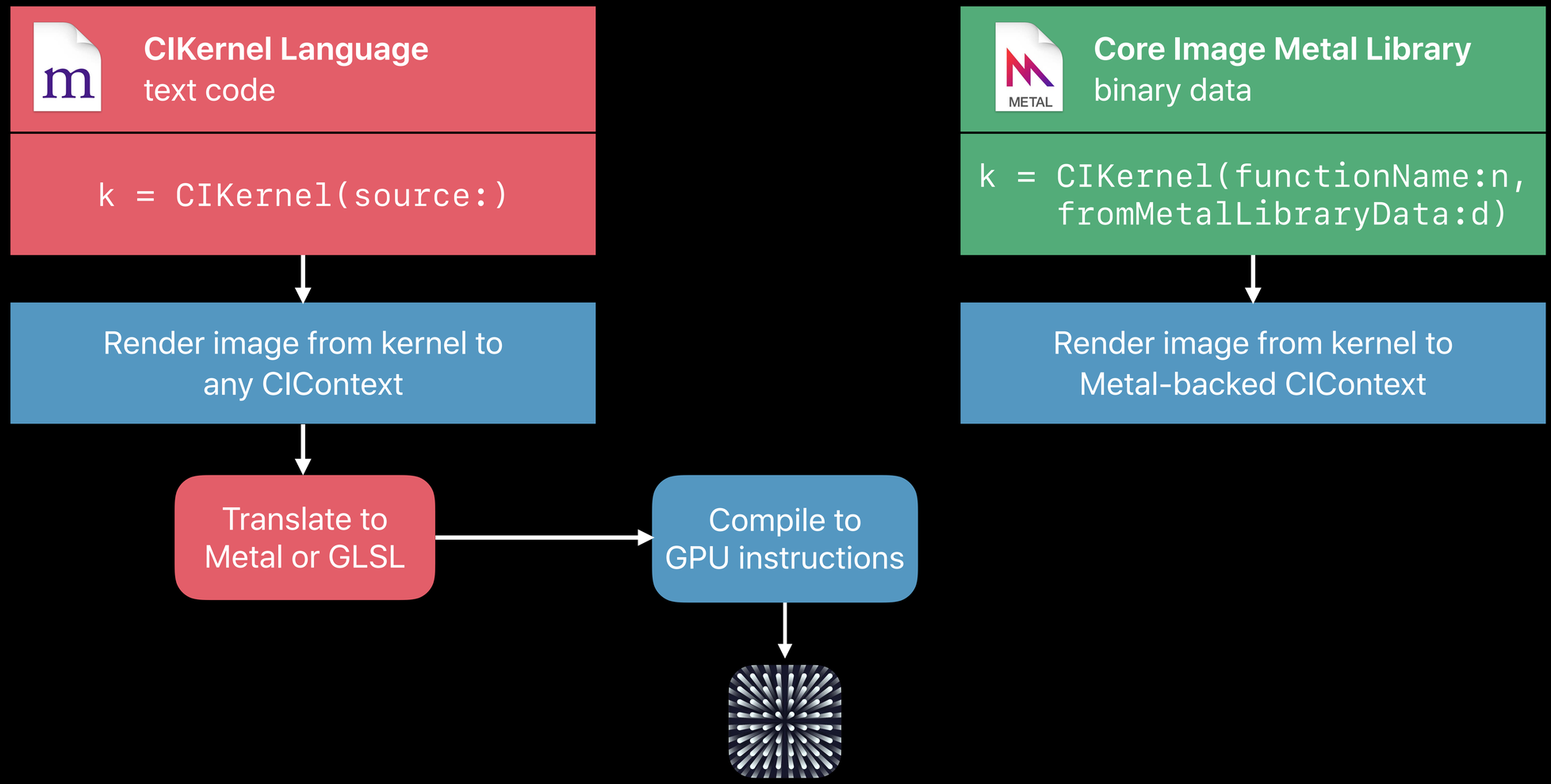
Translate to Metal or GLSL

Compile to GPU instructions

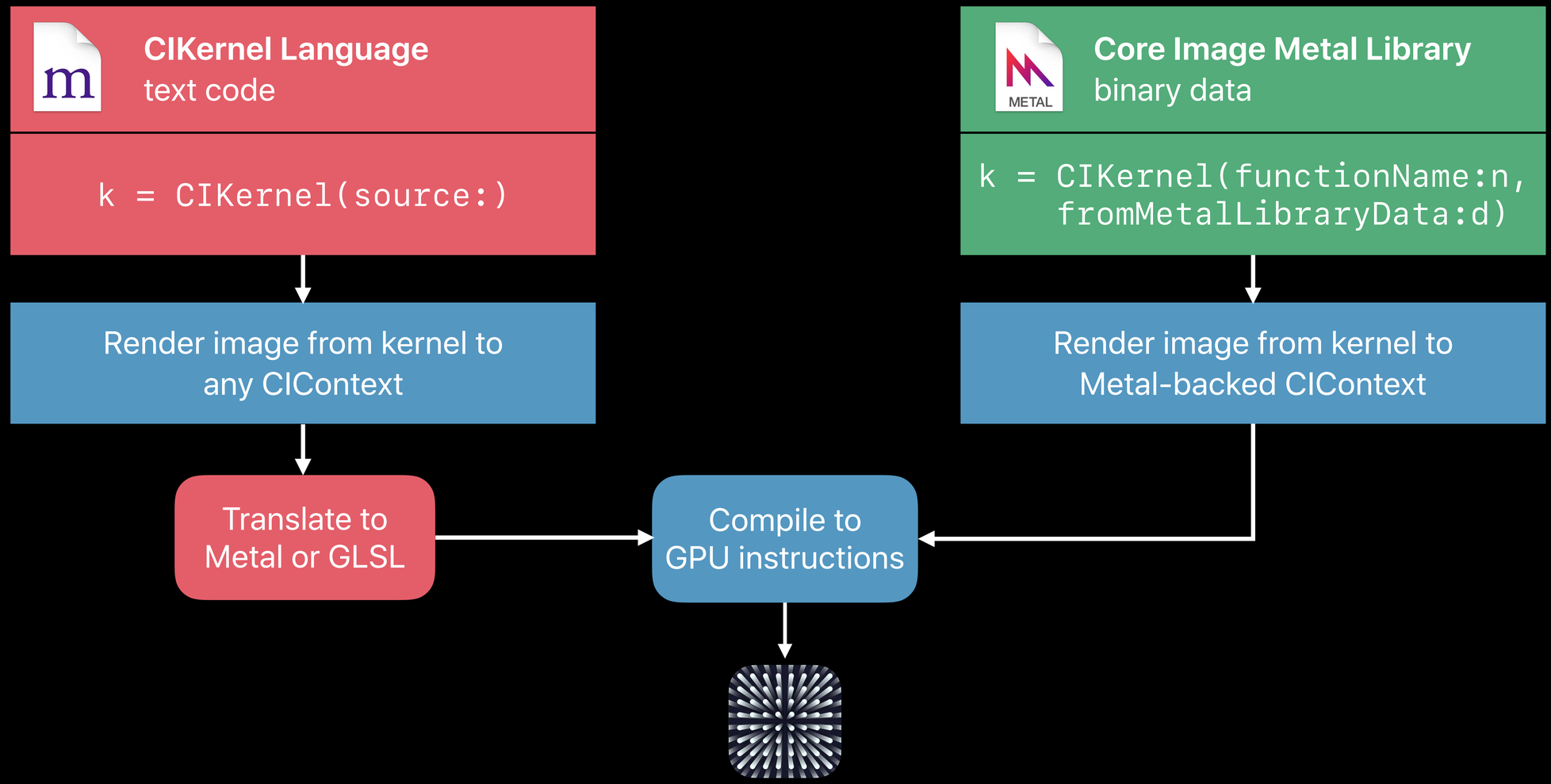
Core Image Metal Library
binary data

```
k = CIKernel(functionName:n,
    fromMetalLibraryData:d)
```

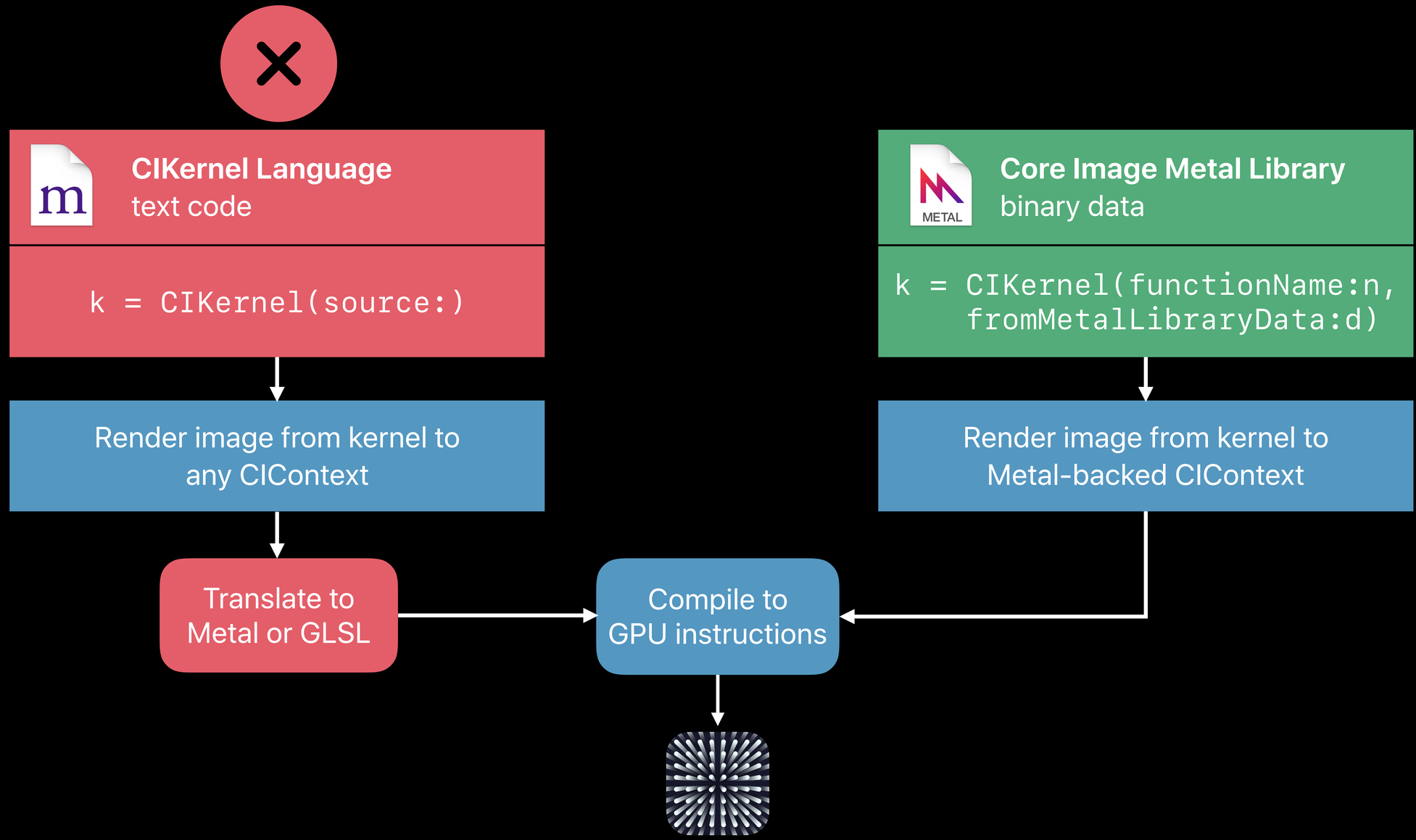
Two Ways To Write CIKernels



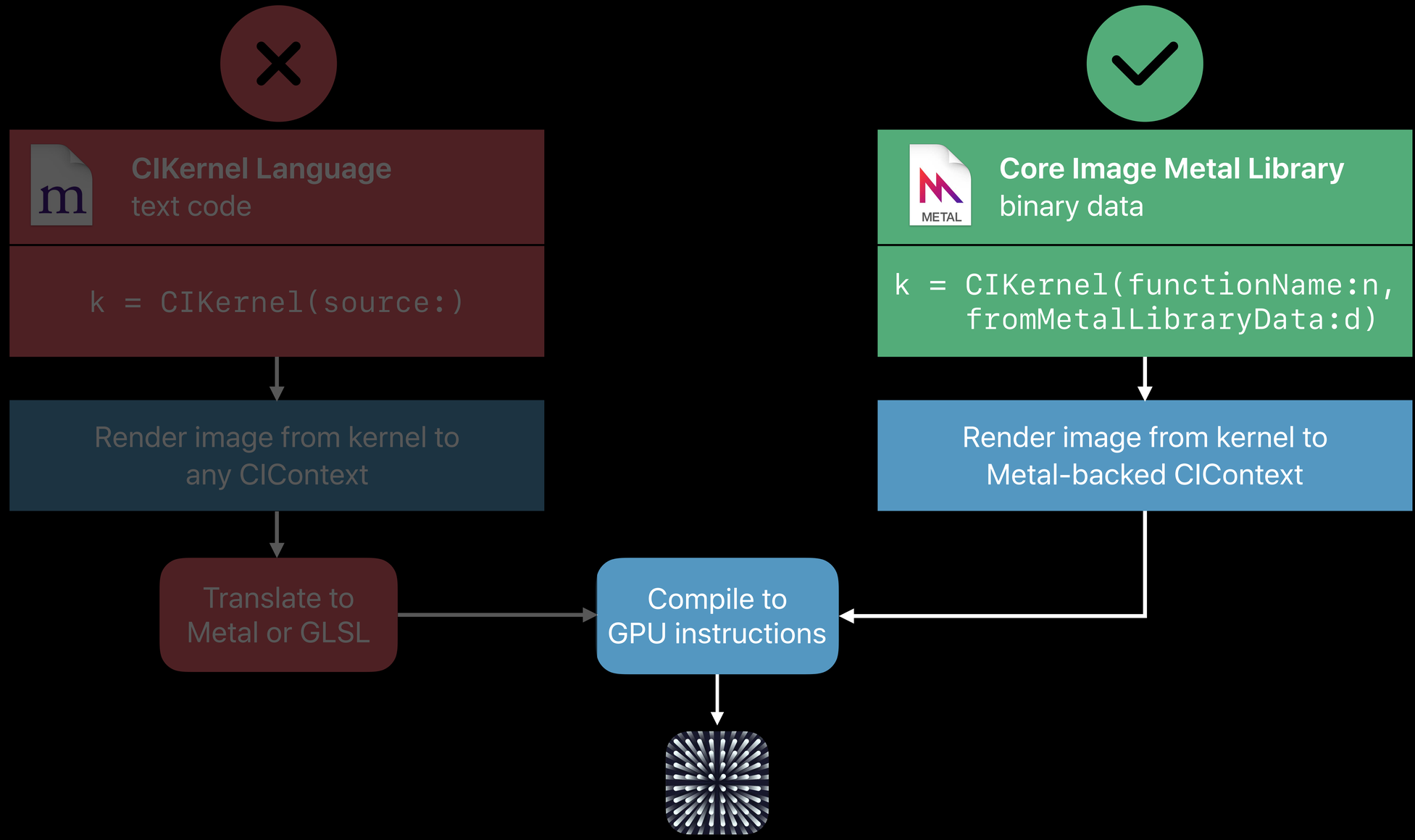
Two Ways To Write CIKernels



Two Ways To Write CIKernels



Two Ways To Write CIKernels



New Features for Better Performance

NEW

New Features for Better Performance

NEW

Half float support

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)
- Smaller registers increase utilization

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)
- Smaller registers increase utilization

Group reads

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)
- Smaller registers increase utilization

Group reads

- 4 single-channel pixels per read

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)
- Smaller registers increase utilization

Group reads

- 4 single-channel pixels per read

Group writes

New Features for Better Performance

NEW

Half float support

- Faster operations (on A11 devices)
- Smaller registers increase utilization

Group reads

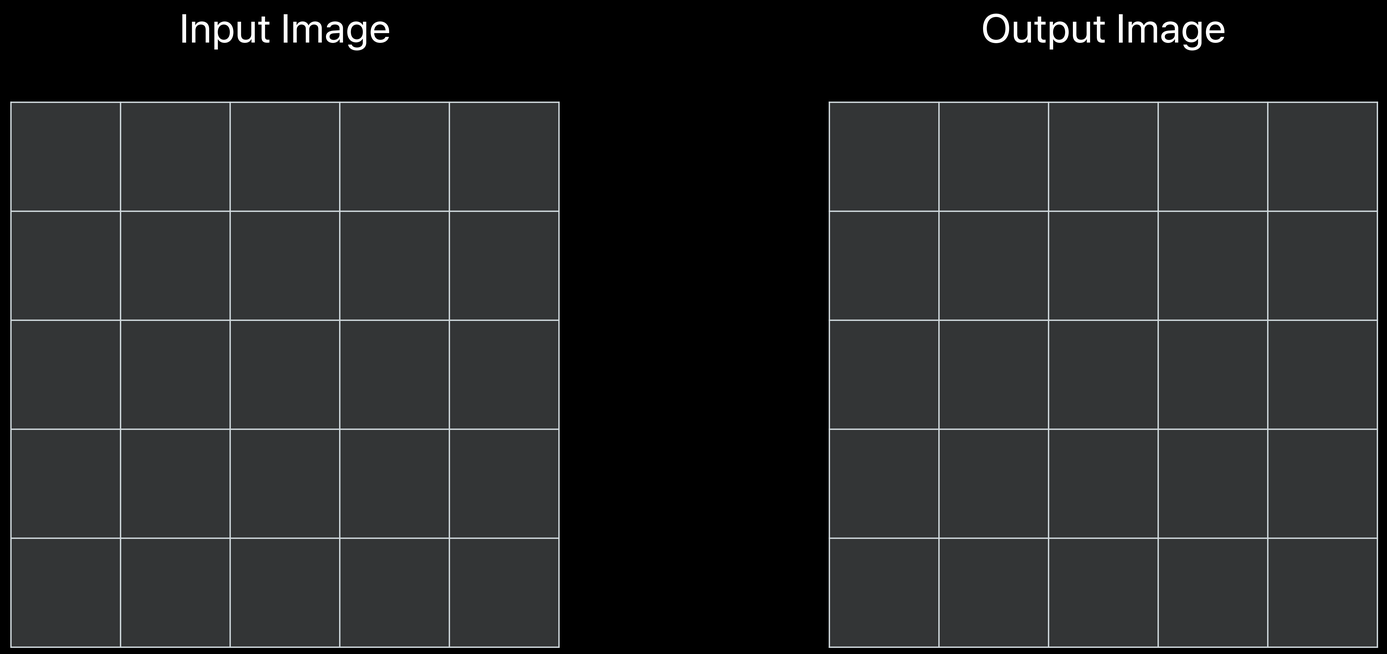
- 4 single-channel pixels per read

Group writes

- 4 pixels written per kernel execution

New Features for Better Performance

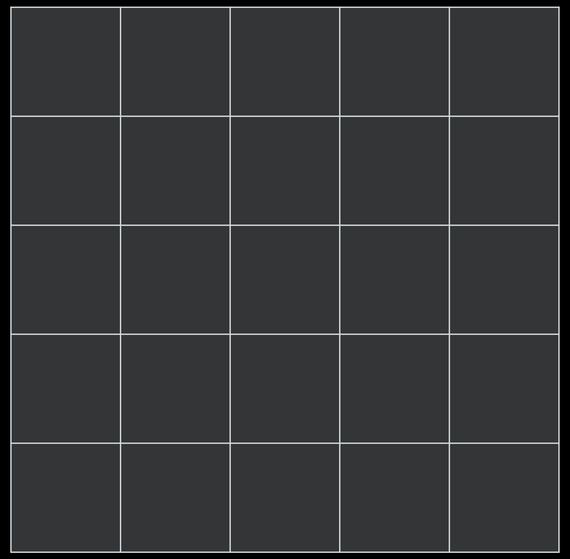
A simple 3x3 convolution example



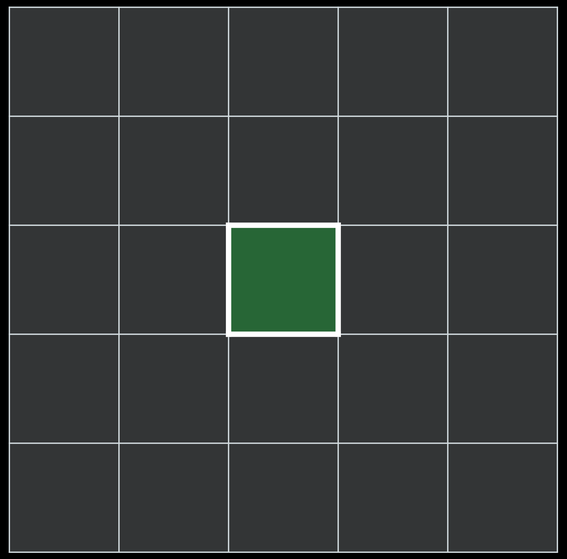
New Features for Better Performance

A simple 3x3 convolution example

Input Image



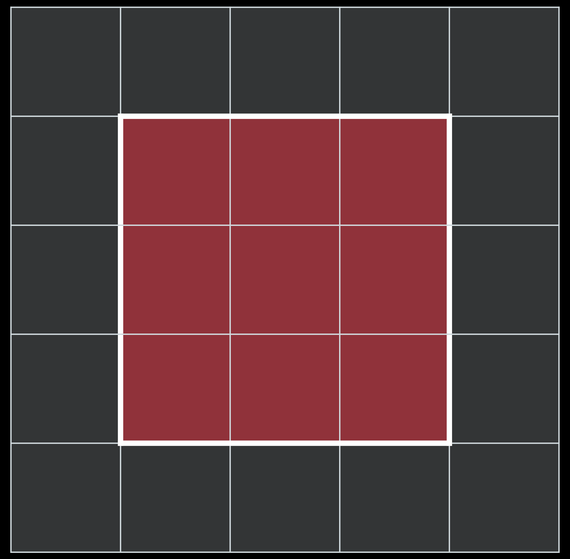
Output Image



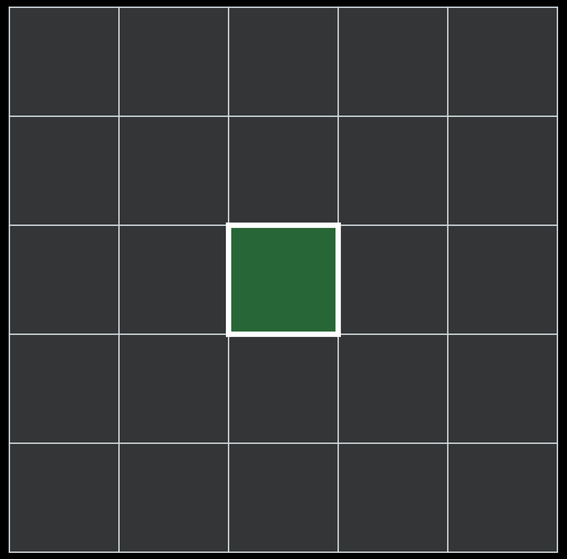
New Features for Better Performance

A simple 3x3 convolution example

Input Image



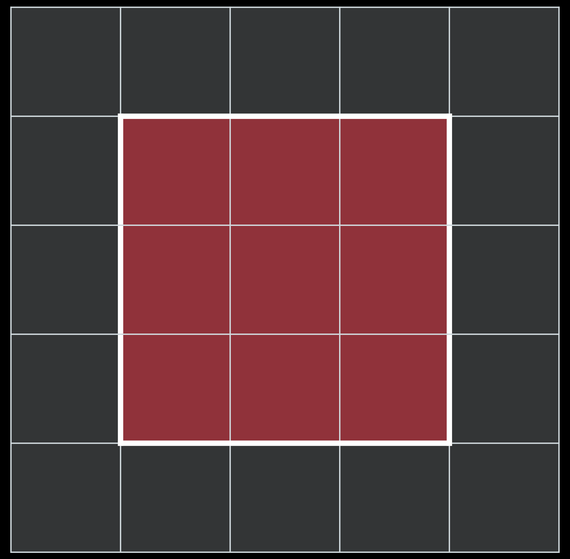
Output Image



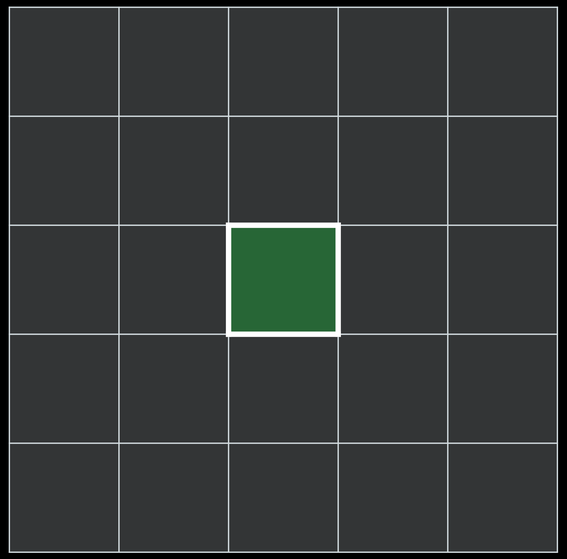
New Features for Better Performance

A simple 3x3 convolution example

Input Image



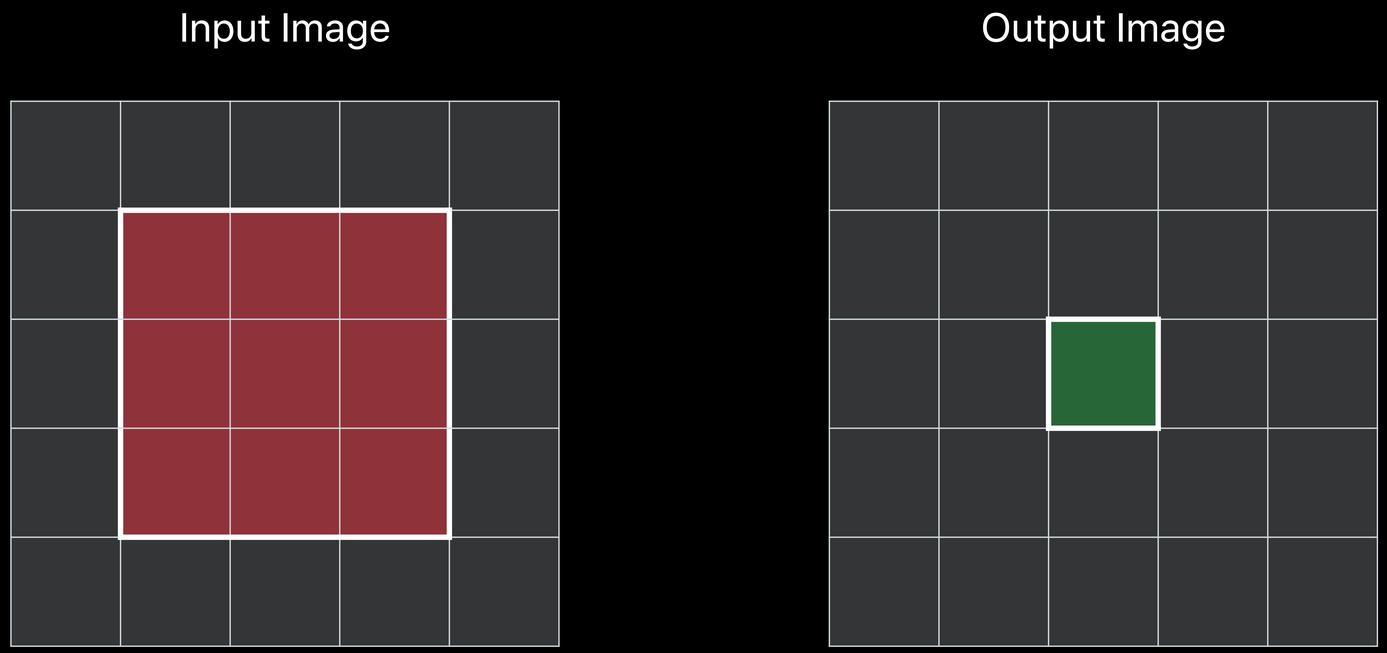
Output Image



9 pixels read for each 1 pixel written

New Features for Better Performance

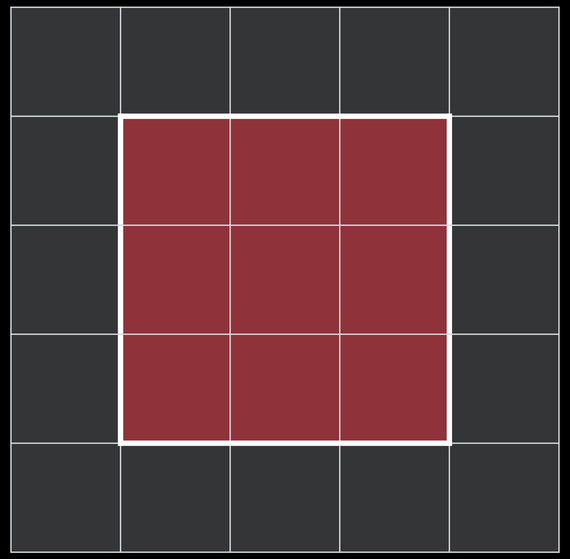
A simple 3x3 convolution example using group writes



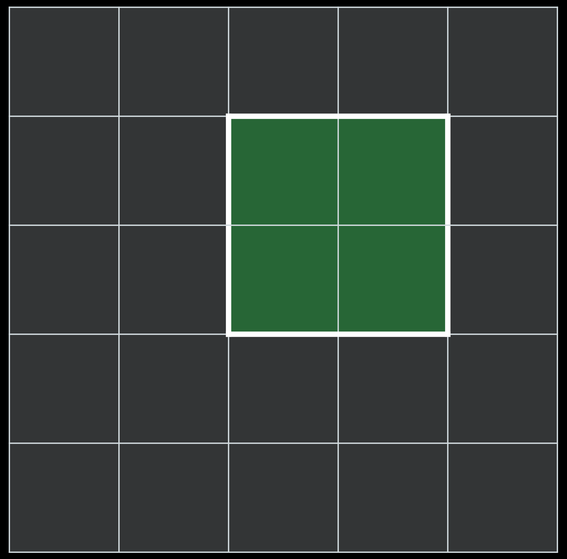
New Features for Better Performance

A simple 3x3 convolution example using group writes

Input Image



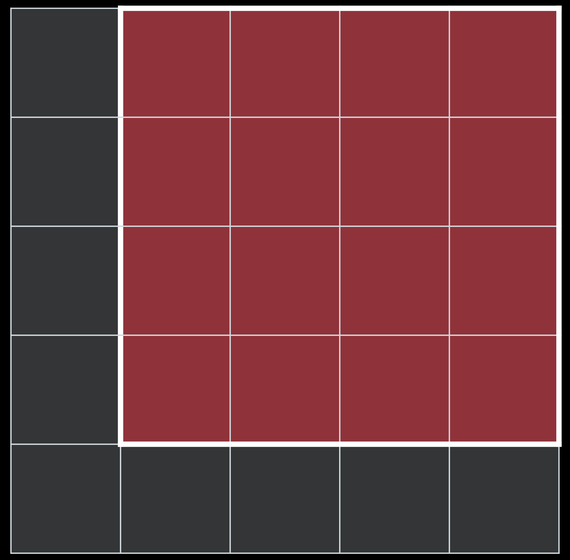
Output Image



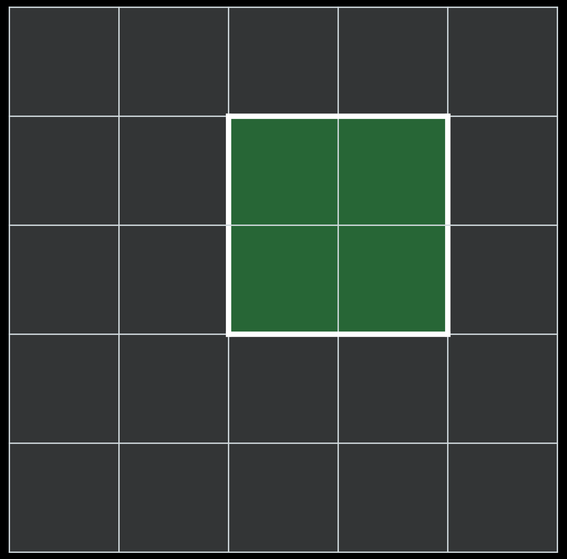
New Features for Better Performance

A simple 3x3 convolution example using group writes

Input Image

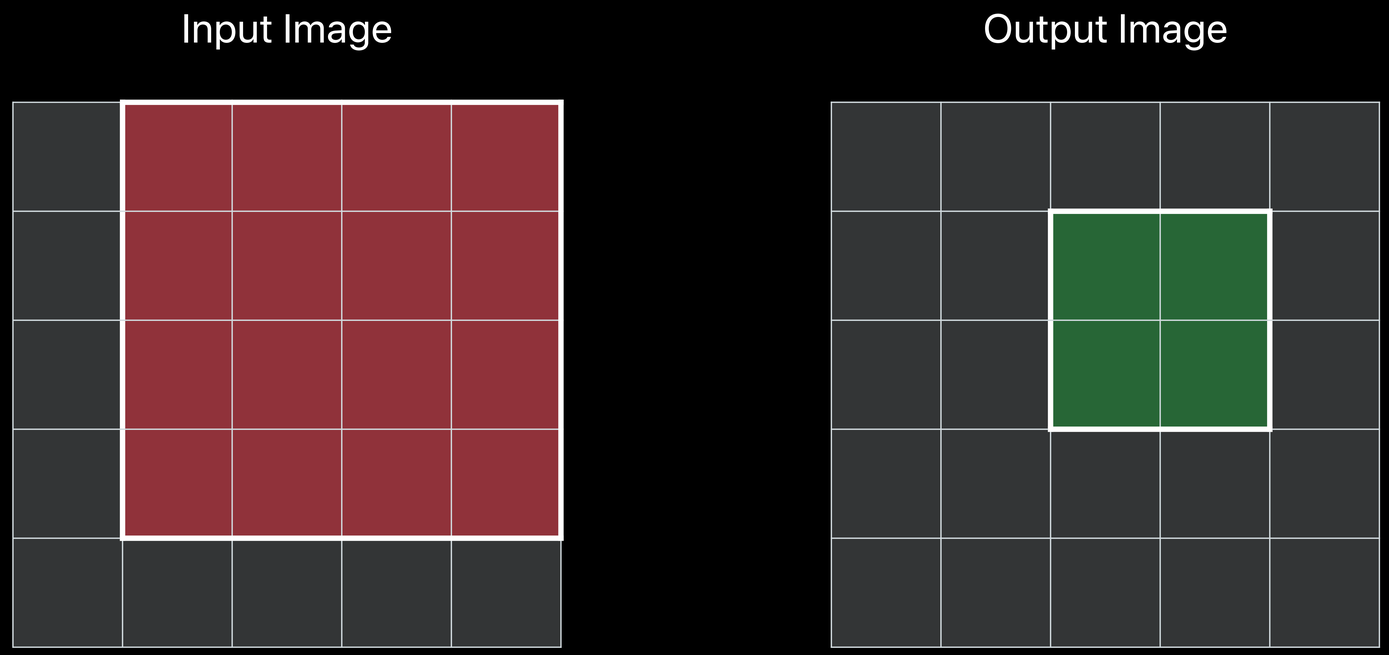


Output Image



New Features for Better Performance

A simple 3x3 convolution example using group writes

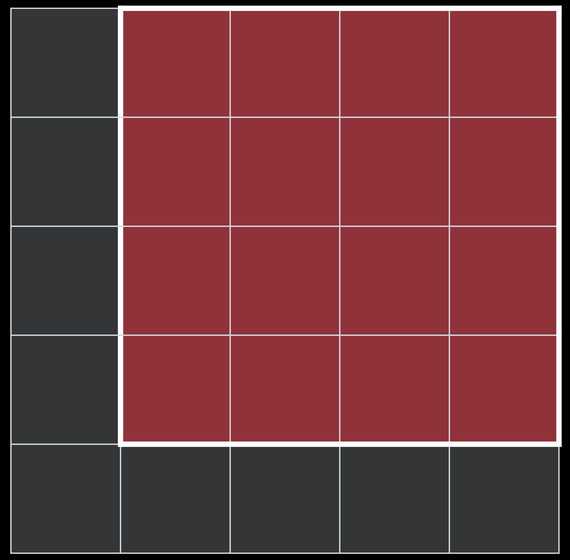


16 pixels read for each 4 pixels written

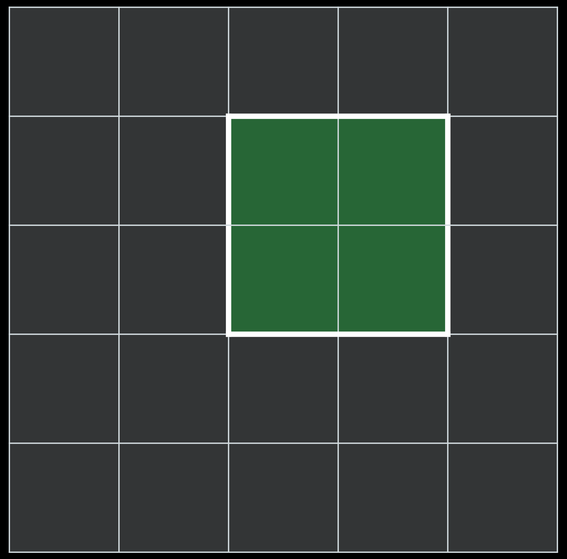
New Features for Better Performance

A simple 3x3 convolution example using group writes and gathers

Input Image

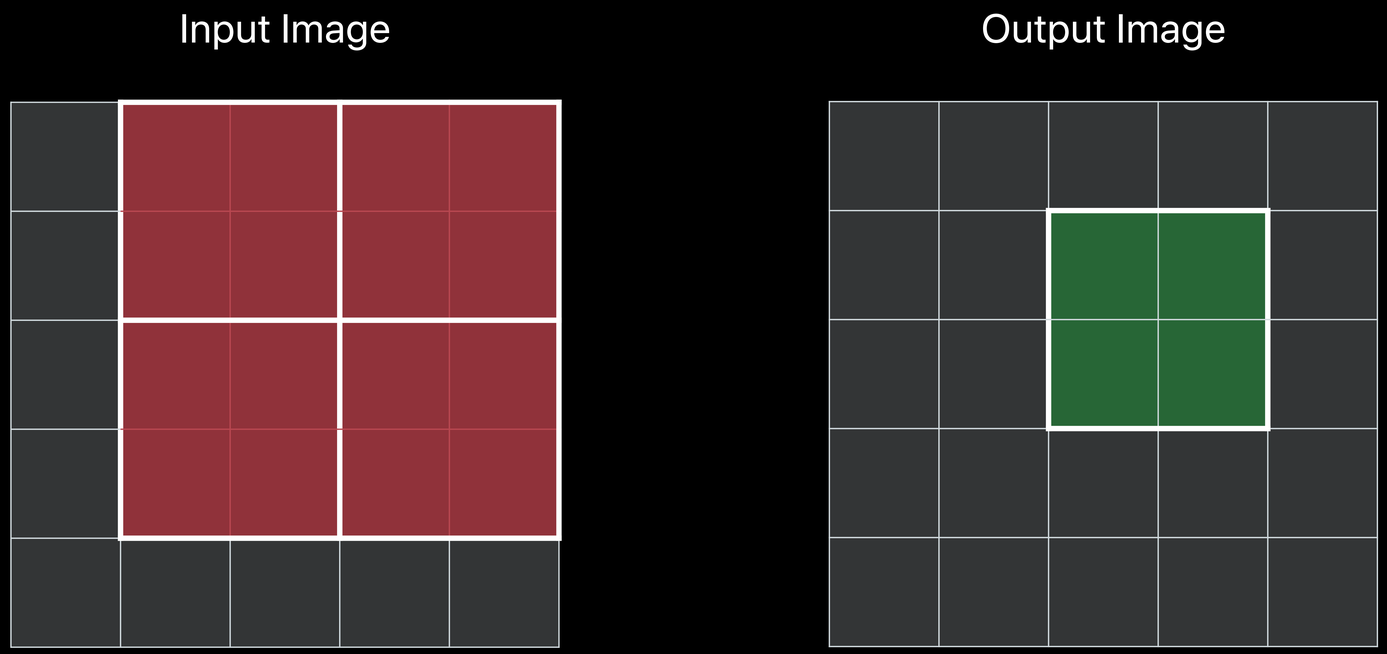


Output Image



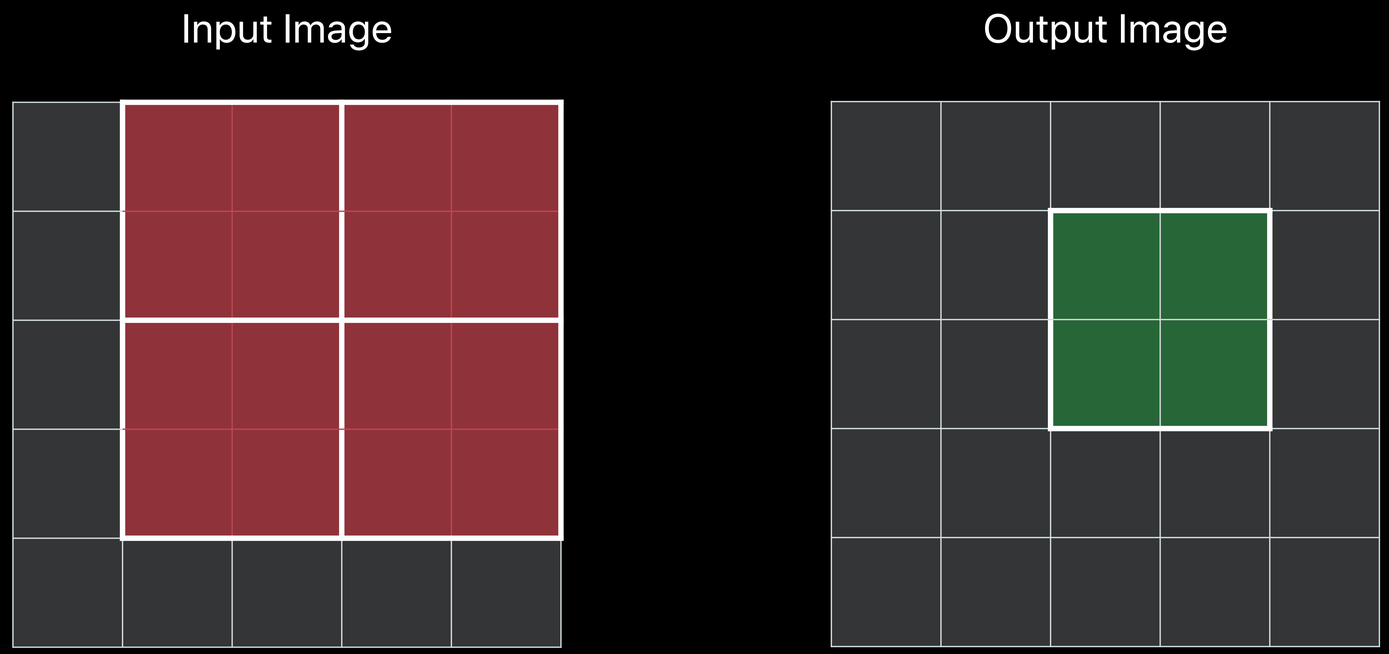
New Features for Better Performance

A simple 3x3 convolution example using group writes and group reads



New Features for Better Performance

A simple 3x3 convolution example using group writes and group reads



4 groups read for each 4 pixels written

```
// Red channel convolve

kernel
vec4 redConvolve (sampler s)
{
    vec2 dc = destCoord();
    float v1 =  sample(s, samplerTransform(s, dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 =  sample(s, samplerTransform(s, dc + vec2(0.0,0.0)).r;
    ...
    float v9 =  sample(s, samplerTransform(s, dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return vec4(result, 0.0,0.0,1.0);
}
```

```
// Red channel convolve

kernel
vec4 redConvolve (sampler s)
{
    vec2 dc = destCoord();
    float v1 =  sample(s, samplerTransform(s, dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 =  sample(s, samplerTransform(s, dc + vec2(0.0,0.0)).r;
    ...
    float v9 =  sample(s, samplerTransform(s, dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return vec4(result, 0.0,0.0,1.0);
}
```

```
// Red channel convolve

kernel
vec4 redConvolve (sampler s)
{
    vec2 dc = destCoord();
    float v1 =  sample(s, samplerTransform(s, dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 =  sample(s, samplerTransform(s, dc + vec2(0.0,0.0)).r;
    ...
    float v9 =  sample(s, samplerTransform(s, dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return vec4(result, 0.0,0.0,1.0);
}
```

Step 1:

Convert to Metal

```
// Red channel convolve

kernel
vec4   redConvolve (sampler s)
{
    vec2 dc = destCoord();
    float v1 = sample(s, samplerTransform(s, dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 = sample(s, samplerTransform(s, dc + vec2(0.0,0.0)).r;
    ...
    float v9 = sample(s, samplerTransform(s, dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return   vec4(result, 0.0,0.0,1.0);
}
```

```
// Red channel convolve — convert to Metal


float4 redConvolve (sampler s, destination dest)
{
    vec2 dc = dest.coord();
    float v1 = s.sample(s.transform(dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 = s.sample(s.transform(dc + vec2(0.0,0.0)).r;
    ...
    float v9 = s.sample(s.transform(dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return float4(result, 0.0,0.0,1.0);
}
```

Step 2:

Use half-float

```
// Red channel convolve


float4 redConvolve (sampler s,   destination dest)
{
    float2 dc = dest.coord();
    float v1 =  s.sample(s.transform(dc + vec2(-1.0,-1.0)).r;
    ...
    float v5 =  s.sample(s.transform(dc + vec2(0.0,0.0)).r;
    ...
    float v9 =  s.sample(s.transform(dc + vec2(1.0,1.0)).r;

    float result = (v1 + … + v5 + … + v9) / 9.0;
    return float4(result, 0,0,1);
}
```

```
// Red channel convolve - using half math


half4  redConvolve (sampler_h s, destination_h dest)
{
    float2 dc = dest.coord();
    half  v1 =  s.sample(s.transform(dc + vec2(-1.0,-1.0)).r;
    ...
    half  v5 =  s.sample(s.transform(dc + vec2(0.0,0.0)).r;
    ...
    half  v9 =  s.sample(s.transform(dc + vec2(1.0,1.0)).r;

    half  result = (v1 + … + v5 + … + v9) / 9.0h;
    return  half4(result, 0,0,1);
}
```

Step 3:

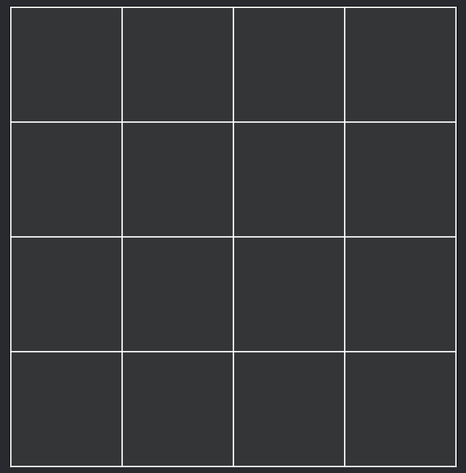
Use Group reads and writes

```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

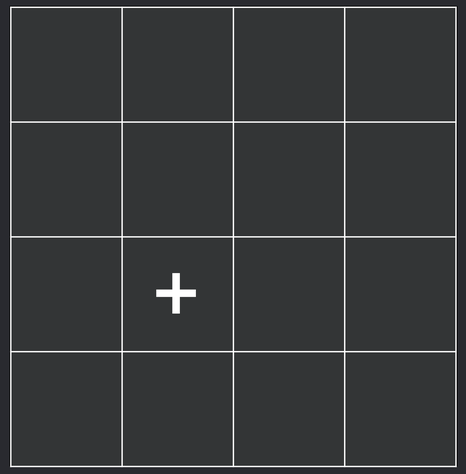


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

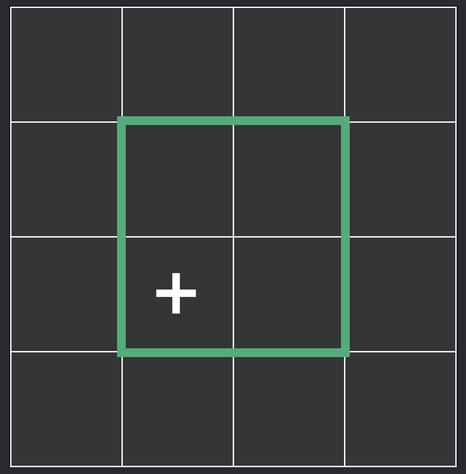


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

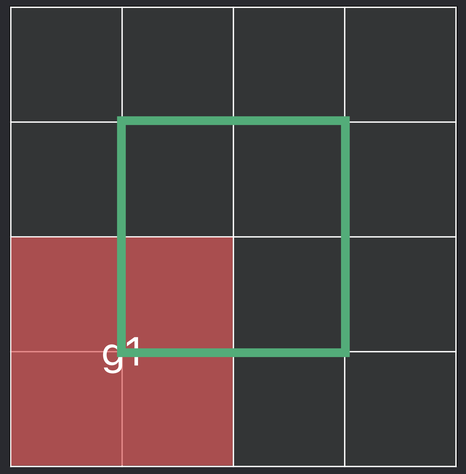


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

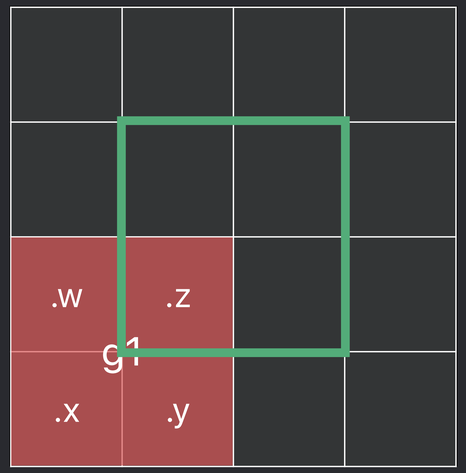


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

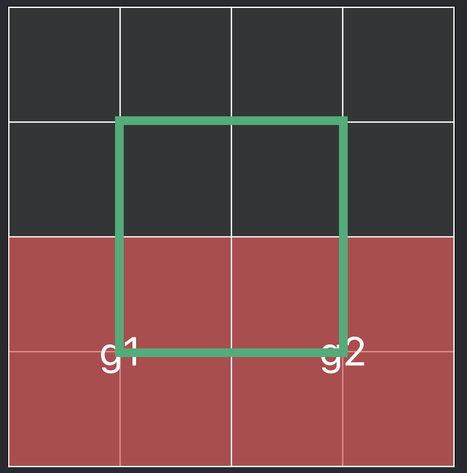


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

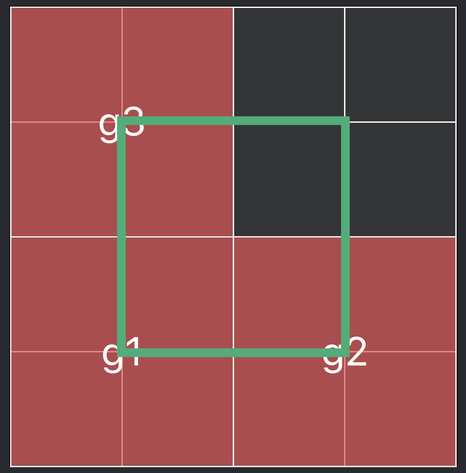


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

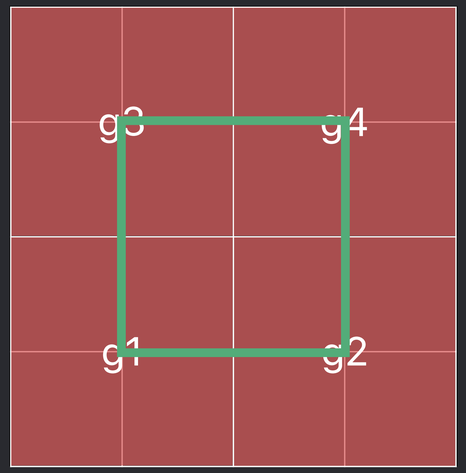


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

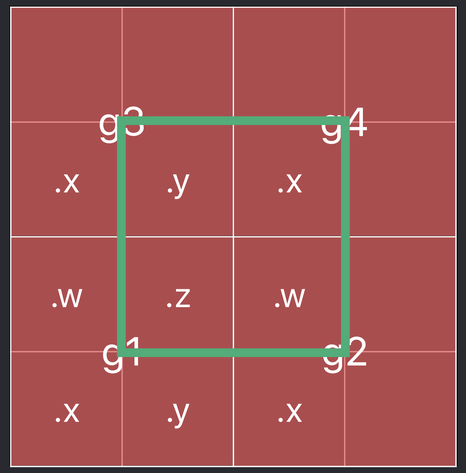


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

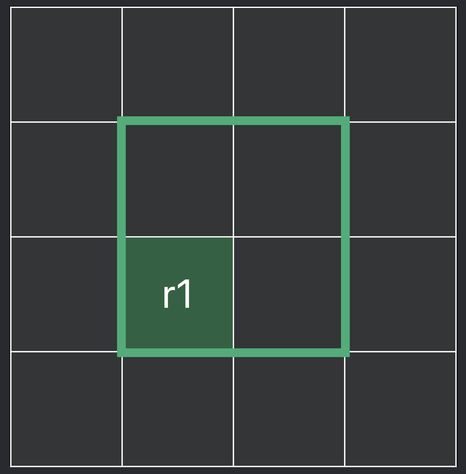


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

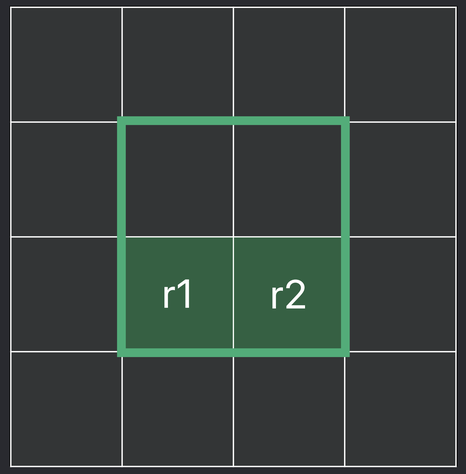


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

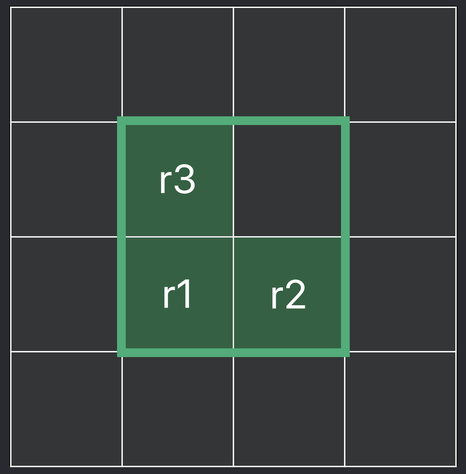


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

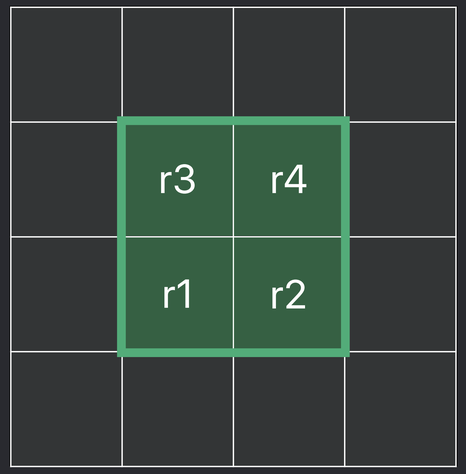


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```

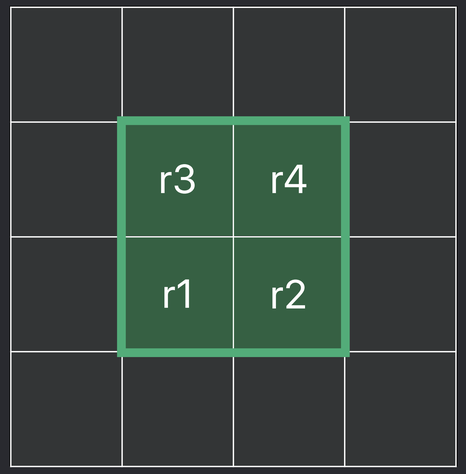


```
// Red channel convolve - using gather reads and group writes

void redConvolve (sampler_h s, group::destination_h dest)
{
    vec2 dc = dest.coord();
    half4 g1 = s.gatherX(s.transform(s, dc + float2(-0.5,-0.5));
    half4 g2 = s.gatherX(s.transform(s, dc + float2( 1.5,-0.5));
    half4 g3 = s.gatherX(s.transform(s, dc + float2(-0.5, 1.5));
    half4 g4 = s.gatherX(s.transform(s, dc + float2( 1.5, 1.5));

    half r1 = (g1.x + g1.y + g1.z + g1.w + g2.x + g2.w + g3.x + g3.y + g4.x) / 9.0h;
    half r2 = (g2.x + g2.y + g2.z + g2.w + g1.y + g1.z + g3.y + g4.x + g4.y) / 9.0h;
    half r3 = (g3.x + g3.y + g3.z + g3.w + g1.w + g1.z + g2.w + g4.x + g4.w) / 9.0h;
    half r4 = (g4.x + g4.y + g4.z + g4.w + g1.z + g2.w + g2.z + g3.y + g3.z) / 9.0h;

    dest.write(half4(r1, 0,0,1), half4(r2, 0,0,1),
               half4(r3, 0,0,1), half4(r4, 0,0,1));
}
```



2x

Faster kernel execution time

2x

Faster kernel execution time

---

https://developer.apple.com/documentation/coreimage/writing_custom_kernels

---

Hello, Python.

Prototyping with Core Image

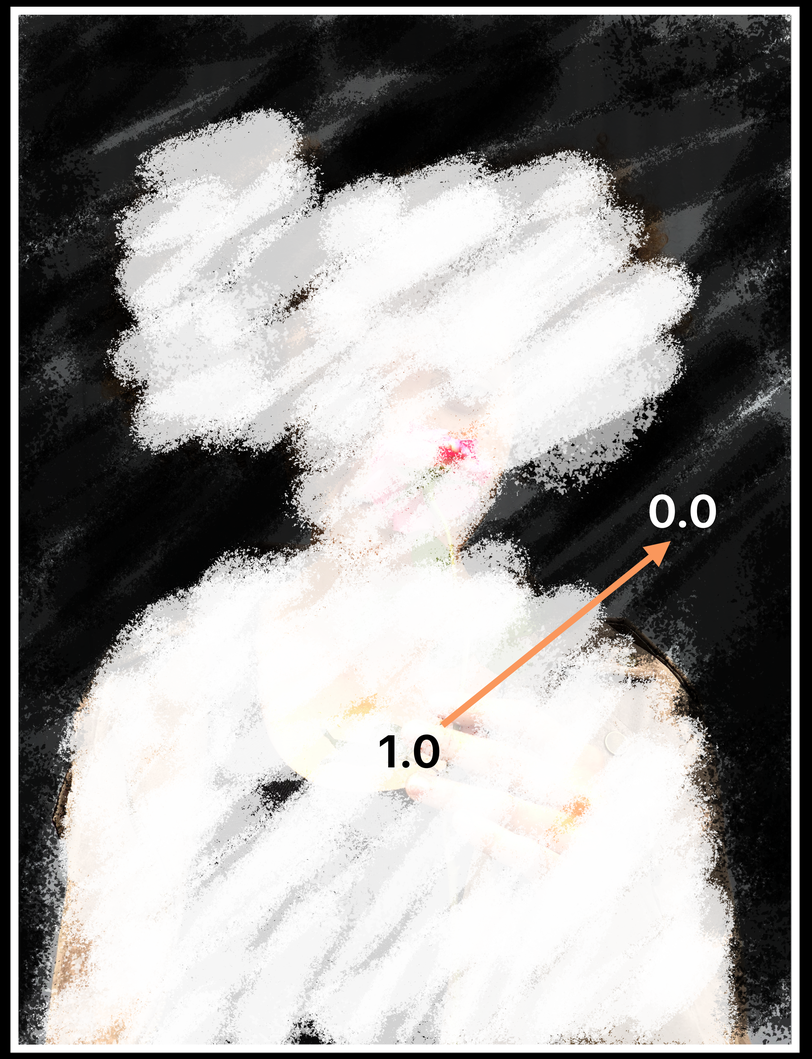
Emmanuel Piuze-Phaneuf, Core Image

Filter Concept

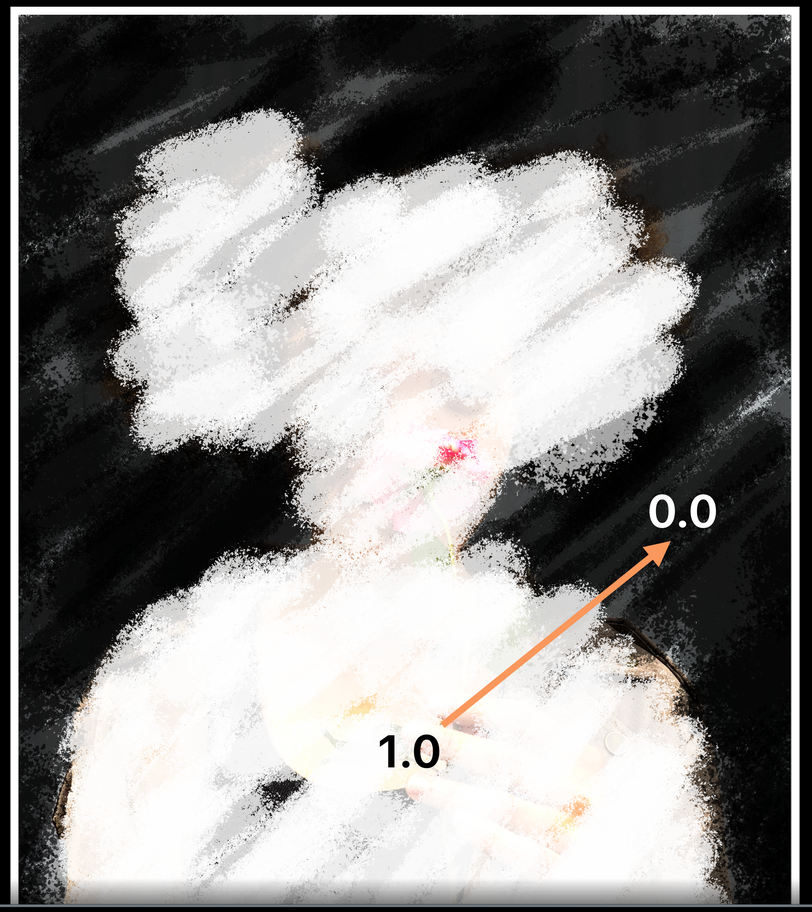
Filter Concept

Filter Concept

Filter Concept



Filter Concept



The Great Divide

The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

TensorFlow

MATLAB

Python

Swift Playground

…

The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

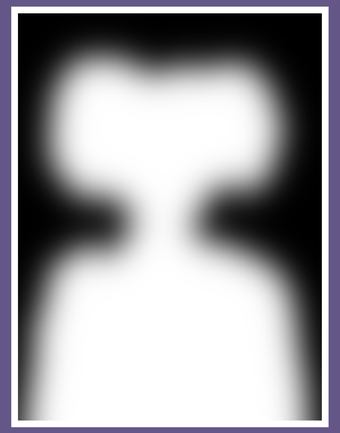
TensorFlow

MATLAB

Python

Swift Playground

…



The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

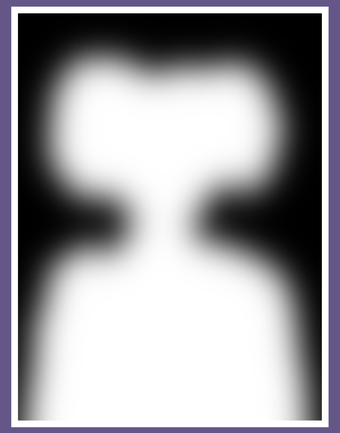
TensorFlow

MATLAB

Python

Swift Playground

…



Product

Core Image

Metal

MPS

vImage

…

The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

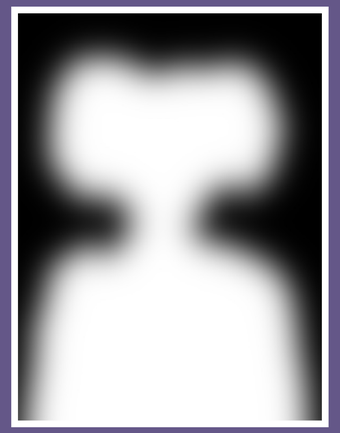
TensorFlow

MATLAB

Python

Swift Playground

…



Product

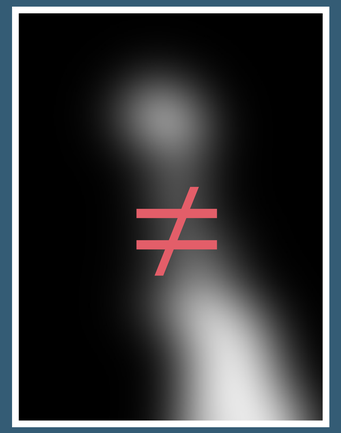
Core Image

Metal

MPS

vImage

…



The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

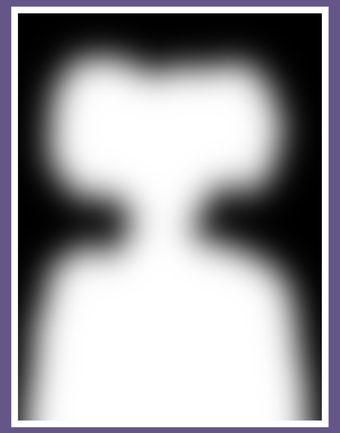
TensorFlow

MATLAB

Python

Swift Playground

…



Product

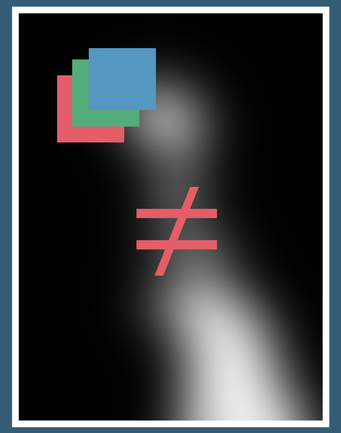
Core Image

Metal

MPS

vImage

…



The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

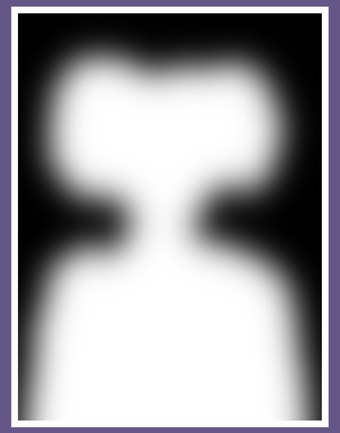
TensorFlow

MATLAB

Python

Swift Playground

…



Product

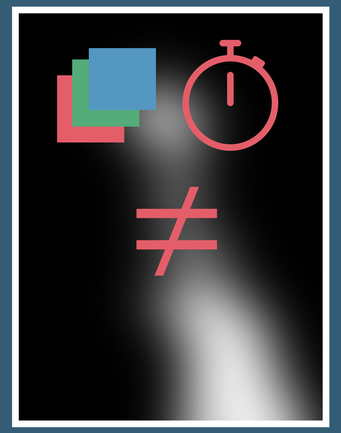
Core Image

Metal

MPS

vImage

…



The Great Divide

Prototype

Turi Create

NumPy

SciPy

OpenCV

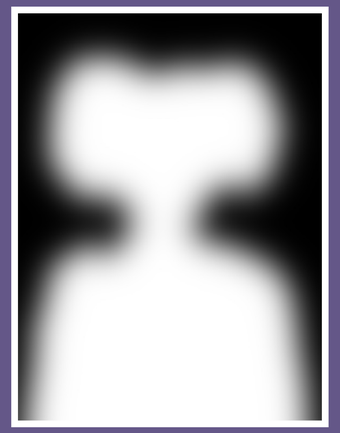
TensorFlow

MATLAB

Python

Swift Playground

…



Product

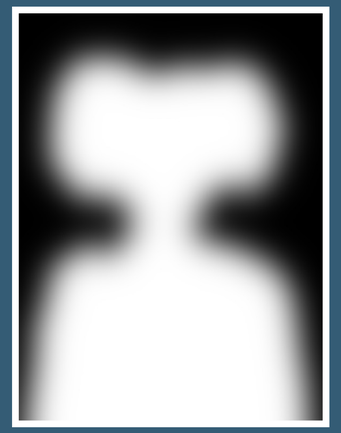
Core Image

Metal

MPS

vImage

…



NEW

PyCoreImage

PyCoreImage

Prototype

Turi Create

NumPy

SciPy

OpenCV

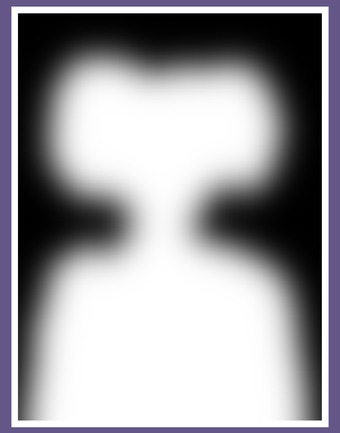
TensorFlow

MATLAB

Python

Swift Playground

...



Product

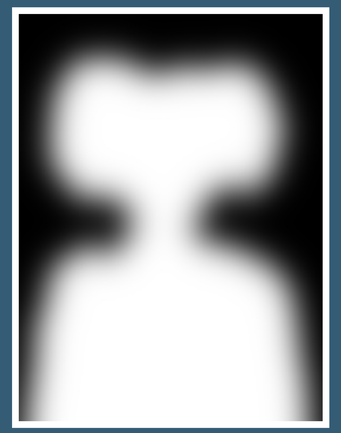
Core Image

Metal

MPS

vImage

...



PyCoreImage

Prototype

Turi Create

NumPy

SciPy

OpenCV

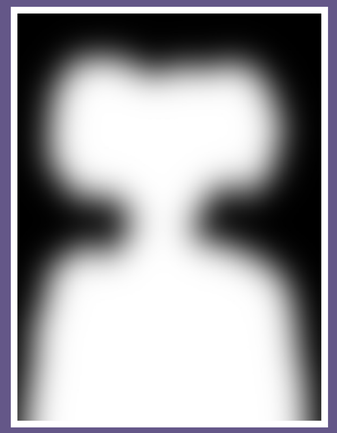
TensorFlow

MATLAB

Python

Swift Playground

...



Product

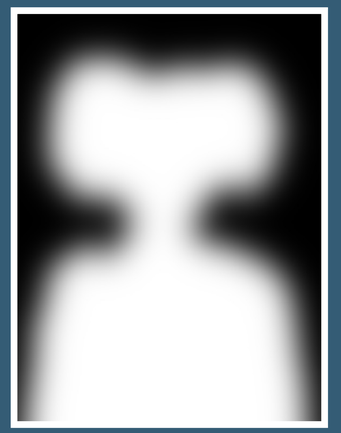
Core Image

Metal

MPS

vImage

...



Prototype

Turi Create

NumPy

SciPy

OpenCV

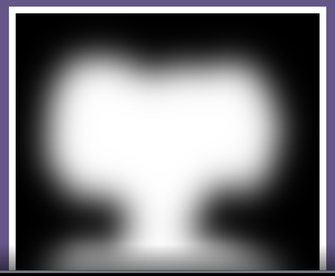
TensorFlow

MATLAB

Python

Swift Playground

...



Product

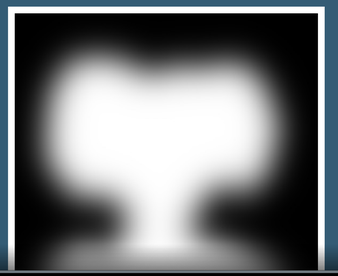
Core Image

Metal

MPS

vImage

...



Create Your Own Swift Playgrounds Subscription

Executive Ballroom

Friday 9:00AM

PyObjC

PyObjC

Since Mac OS X 10.5 Leopard

PyObjC

Since Mac OS X 10.5 Leopard

Python and ObjC bridge

PyObjC

Since Mac OS X 10.5 Leopard

Python and ObjC bridge

Colon → underscore

PyObjC

Since Mac OS X 10.5 Leopard

Python and ObjC bridge

Colon → underscore

ObjC

```
#import <CoreImage/CoreImage.h>
CIVector *v = [CIVector vectorWithX:0 Y:1 Z:2 W:3];
```

PyObjC

Since Mac OS X 10.5 Leopard

Python and ObjC bridge

Colon → underscore

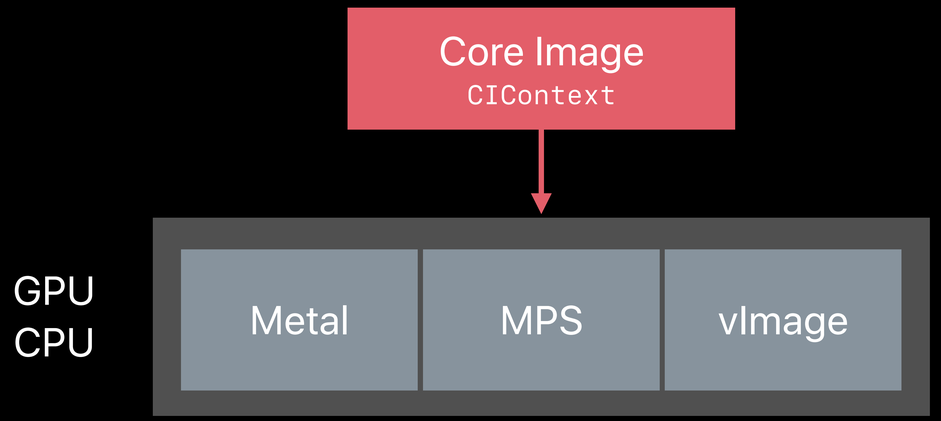
ObjC

```
#import <CoreImage/CoreImage.h>
CIVector *v = [CIVector vectorWithX:0 Y:1 Z:2 W:3];
```

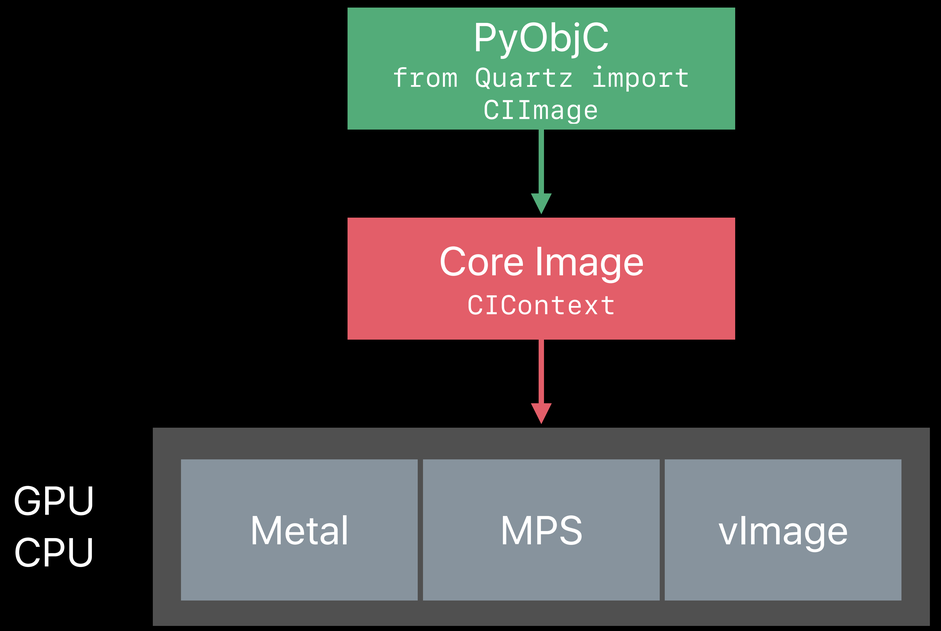
Python

```
from Quartz import CIVector
v = CIVector.vectorWithX_Y_Z_W_(0, 1, 2, 3)
```

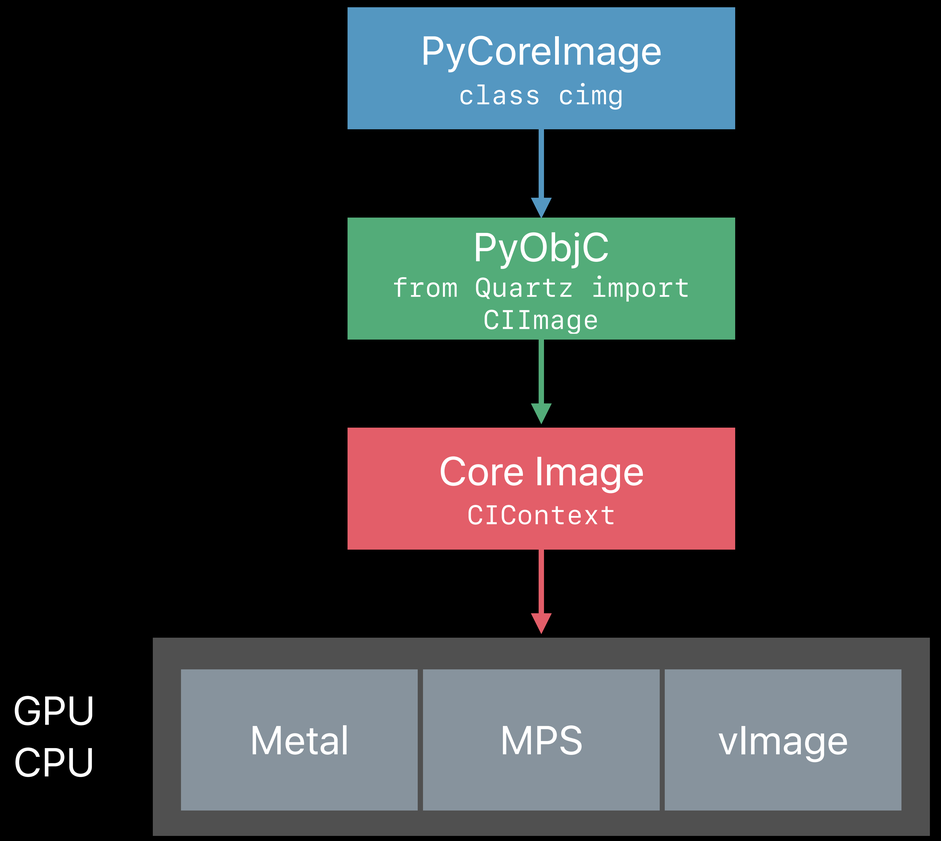
Python Bindings



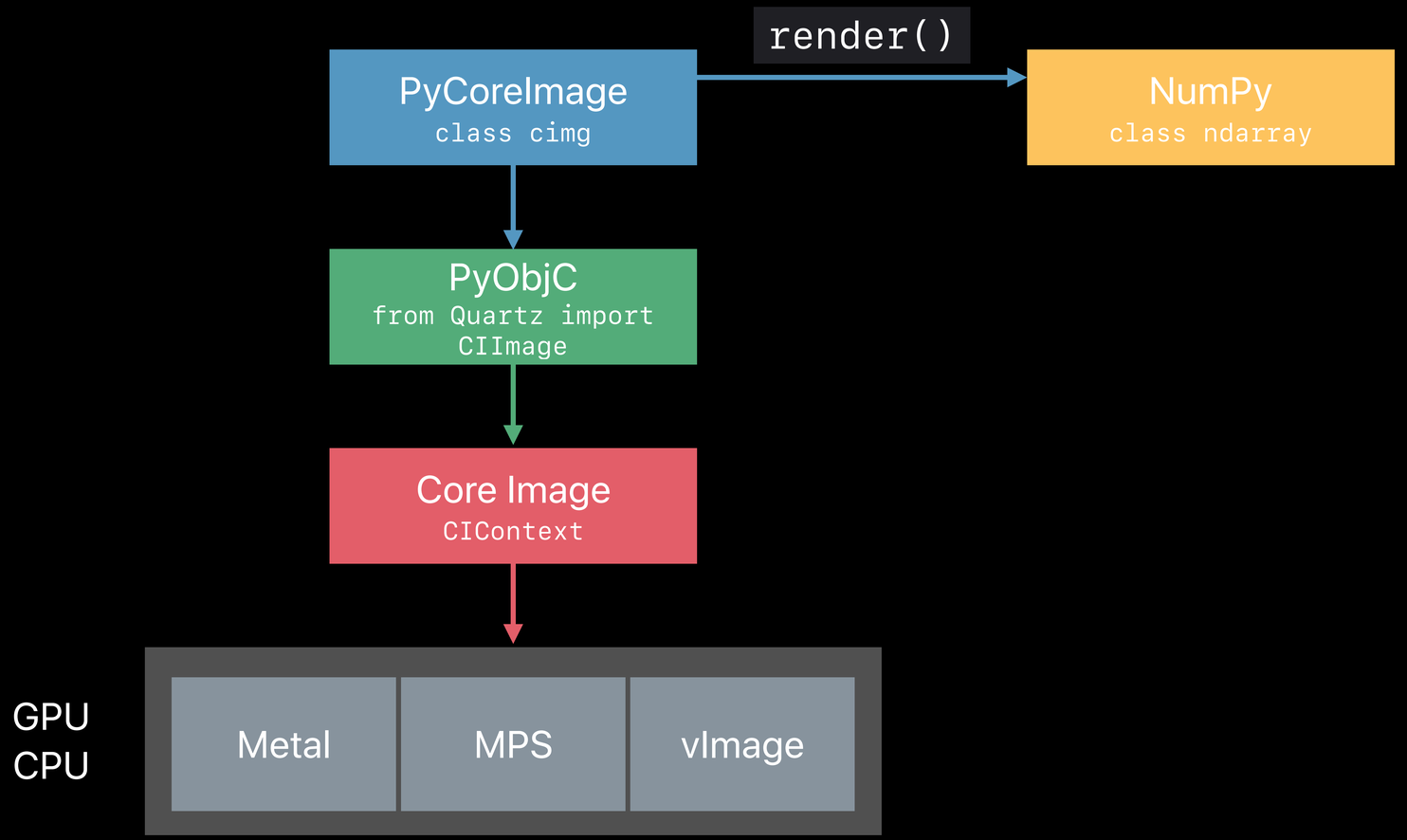
Python Bindings



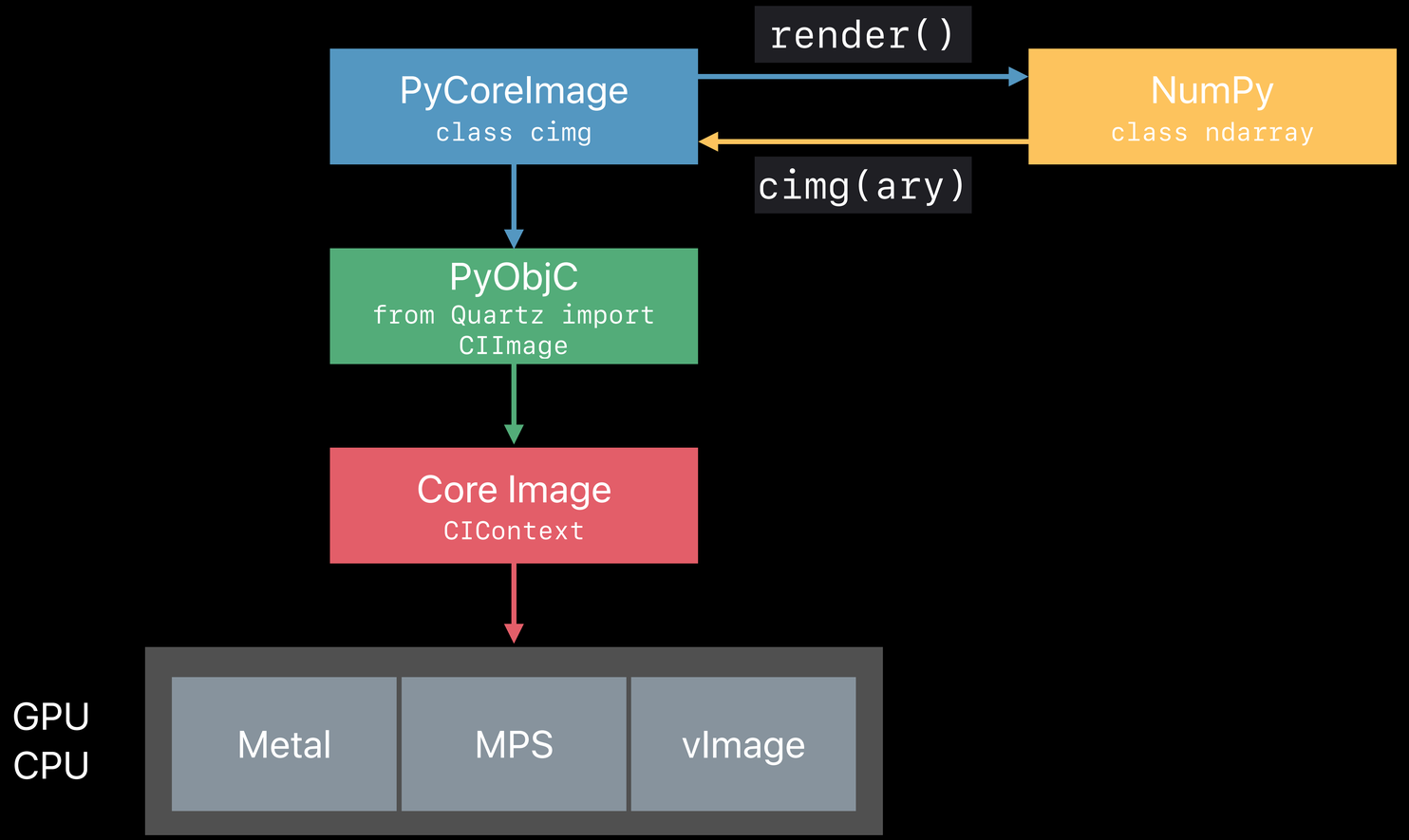
Python Bindings



Python Bindings



Python Bindings



PyCoreImage

Python

PyCoreImage

```python
from pycoreimage.pyci import cimg
```

PyCoreImage

CIImage

Python

```python
from pycoreimage.pyci import cimg
img = cimg.fromFile('img.heif')
```

```
CGImageRef 0x7fb712ec8210(1) RGBX8 alpha_one edge_clamp
rois=[0 0 756 1008]
extent=[infinite][0 0 756 1008]
opaque shallow
digest=8B277BE8791C6E0F
```

PyCoreImage

CIImage

Python

```python
from pycoreimage.pyci import cimg
img = cimg.fromFile('img.heif')
img = img.gaussianBlur(radius=40)
```

```
program graph
get_bitmap
(metal-DG context 1 frame 1 tile 1)
format=RGBAf roi=[0 0 756 1008]
workingFormat=RGBAf
```

```
program
affine [1 0 0 -1 0 1008]
rois=[0 0 756 1008]
extent=[infinite] deep
digest=2617A94E5DCFC1E3
```

```
program format=RGBAf
processor CIBlurProcessor digest=DB491FDF8760CCD4 outputFormat=RGBAf
rois=[0 0 756 1008]
extent=[infinite] deep
digest=9515033E3DC7AC70
```

```
program format=RGBAf
affine [1 0 0 -1 0 1008]
  clampRect [0 0 756 1008]
    swizzle_bgr1
rois=[-75 -75 906 1158]
extent=[infinite][0 0 756 1008]
opaque
digest=A9CAC61E01E36D0A
```

```
CGImageRef 0x7fb712ec8210(1) RGBX8 alpha_one edge_clamp
rois=[0 0 756 1008]
extent=[infinite][0 0 756 1008]
opaque shallow
digest=8B277BE8791C6E0F
```

PyCoreImage

CIImage

Python

```
from pycoreimage.pyci import cimg
img = cimg.fromFile('img.heif')
img = img.gaussianBlur(radius=40)
```

```
program graph
get_bitmap
(metal-DG context 1 frame 1 tile 1)
format=RGBAf roi=[0 0 756 1008]
workingFormat=RGBAf
```

```
program
affine [1 0 0 -1 0 1008]
rois=[0 0 756 1008]
extent=[infinite] deep
```

```
program format=RGBAf
processor CIBlurProcessor digest=DB491FDF8760CCD4 outputFormat=RGBAf
rois=[0 0 756 1008]
extent=[infinite] deep
digest=9515033E3DC7AC70
```

```
affine [1 0 0 -1 0 1008]
  clampRect [0 0 756 1008]
    swizzle_bgr1
rois=[-75 -75 906 1158]
extent=[infinite][0 0 756 1008]
opaque
digest=A9CAC61E01E36D0A
```

```
CGImageRef 0x7fb712ec8210(1) RGBX8 alpha_one edge_clamp
rois=[0 0 756 1008]
extent=[infinite][0 0 756 1008]
opaque shallow
digest=8B277BE8791C6E0F
```

PyCoreImage

CIImage

NumPy

Python

```python
from pycoreimage.pyci import cimg
img = cimg.fromFile('img.heif')
img = img.gaussianBlur(radius=40)
bar = img.render()
```

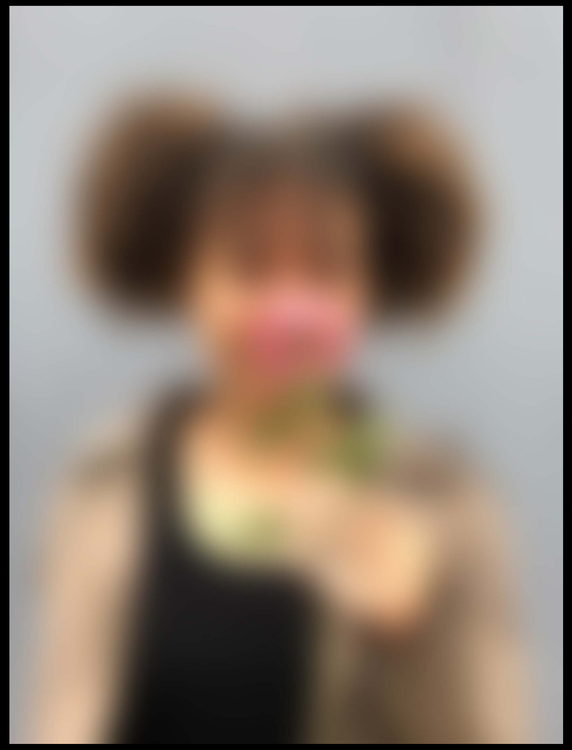
```
program graph
get_bitmap
(metal-DG context 1 frame 1 tile 1)
format=RGBAf roi=[0 0 756 1008]
workingFormat=RGBAf
```

```
program
affine [1 0 0 -1 0 1008]
rois=[0 0 756 1008]
extent=[infinite] deep
digest=2617A94E5DCFC1E3
```

```
program format=RGBAf
processor CIBlurProcessor digest=DB491FDF8760CCD4 outputFormat=RGBAf
rois=[0 0 756 1008]
extent=[infinite] deep
digest=9515033E3DC7AC70
```

```
program format=RGBAf
affine [1 0 0 -1 0 1008]
  clampRect [0 0 756 1008]
    swizzle_bgr1
rois=[-75 -75 906 1158]
extent=[infinite][0 0 756 1008]
opaque
digest=A9CAC61E01E36D0A
```

```
CGImageRef 0x7fb712ec8210(1) RGBX8 alpha_one edge_clamp
rois=[0 0 756 1008]
extent=[infinite][0 0 756 1008]
opaque shallow
digest=8B277BE8791C6E0F
```



Core Image

Core Image

High performance GPU processing

Core Image

High performance GPU processing

Most pixel formats

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

200+ filters

Core Image

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

200+ filters

That’s a lot!

Core Image for Python

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

200+ filters

Core Image for Python

High performance GPU processing

Most pixel formats

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

200+ filters

Core Image for Python

High performance GPU processing

Most pixel formats → RGB float

Most file formats

Image metadata, depth, matte

Color management

Boundary conditions

Infinite images

200+ filters

Core Image for Python

High performance GPU processing

Most pixel formats → RGB float

Most file formats

Image metadata, depth, matte

Color management → sRGB

Boundary conditions

Infinite images

200+ filters

Core Image for Python

High performance GPU processing

Most pixel formats → RGB float

Most file formats

Image metadata, depth, matte

Color management → sRGB

Boundary conditions → clamped with crop

Infinite images

200+ filters

Core Image for Python

High performance GPU processing

Most pixel formats → RGB float

Most file formats

Image metadata, depth, matte

Color management → sRGB

Boundary conditions → clamped with crop

Infinite images → finite only

200+ filters

Cheat Sheet

Cheat Sheet

Import #

```python
from pycoreimage.pyci import cimg
```

Cheat Sheet

		Python
Import	#	`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`

Cheat Sheet

		Python
Import	#	`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`

Swift

```
var img : CIImage! = CIImage(contentsOfFile:“img.heif")
```

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`
Matte		`cimg.fromFile('img.heif', useMatte=True)`

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`
Matte		`cimg.fromFile('img.heif', useMatte=True)`
NumPy		`cimg(ary), img.render()`

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`
Matte		`cimg.fromFile('img.heif', useMatte=True)`
NumPy		`cimg(ary), img.render()`

Swift

```
let dest = CIRenderDestination(bitmapData: &bytes, width: w, height: h,
                               bytesPerRow: bpr, format: .RGBAf)
try CIContext().startTask(toRender: img, to: dest)
```

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`
Matte		`cimg.fromFile('img.heif', useMatte=True)`
NumPy		`cimg(ary), img.render()`
Color		`cimg.fromColor(0.14, 0.15, 0.92)`

Cheat Sheet

		Python
Import		`from pycoreimage.pyci import cimg`
File		`cimg.fromFile('img.heif')`
Depth		`cimg.fromFile('img.heif', useDepth=True)`
Matte		`cimg.fromFile('img.heif', useMatte=True)`
NumPy		`cimg(ary), img.render()`
Color		`cimg.fromColor(0.14, 0.15, 0.92)`
Generator		`cimg.fromGenerator('QRCodeGenerator', message='42')`

Cheat Sheet

Cheat Sheet

Filter	f	`img.gaussianBlur(radius=25)` Python

Cheat Sheet

Filter	f	Python `img.gaussianBlur(radius=25)`
		Swift `img = img.applyingFilter("CIGaussianBlur", parameters: ["inputRadius" : 25])`

Cheat Sheet

Filter	f	Python `img.gaussianBlur(radius=25)`
Kernel	k	`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`

Cheat Sheet

		Python
Filter	f	`img.gaussianBlur(radius=25)`
Kernel	k	`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`
Composite		`img.over(img2)`

Cheat Sheet

		Python
Filter		`img.gaussianBlur(radius=25)`
Kernel		`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`
Composite		`img.over(img2)`
Translate		`img.translate(tx, ty)`

Cheat Sheet

		Python
Filter		`img.gaussianBlur(radius=25)`
Kernel		`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`
Composite		`img.over(img2)`
Translate		`img.translate(tx, ty)`
Scale		`img.scale(sx, sy)`

Cheat Sheet

		Python
Filter		`img.gaussianBlur(radius=25)`
Kernel		`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`
Composite		`img.over(img2)`
Translate		`img.translate(tx, ty)`
Scale		`img.scale(sx, sy)`
Rotate		`img.rotate(rads)`

Cheat Sheet

		Python
Filter		`img.gaussianBlur(radius=25)`
Kernel		`img.applyKernel(source, mask, 0.5,` `  extent=img.extent, roi=lambda index, r: r)`
Composite		`img.over(img2)`
Translate		`img.translate(tx, ty)`
Scale		`img.scale(sx, sy)`
Rotate		`img.rotate(rads)`
Crop		`img.crop(0, 0, 1024, 768)`

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """
```

```
img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

```
src = """ kernel vec4 sharpen(sampler s, float amount) {
 // 5 tap Laplacian
 //  0 -1  0
 // -1  4 -1
 //  0 -1  0
 vec2 p = destCoord();
 vec4 a = sample(s, samplerTransform(s, p + vec2(-1.0, 0.0)));
 vec4 b = sample(s, samplerTransform(s, p + vec2( 0.0, 1.0)));
 vec4 c = sample(s, samplerTransform(s, p));
 vec4 d = sample(s, samplerTransform(s, p + vec2( 1.0, 0.0)));
 vec4 e = sample(s, samplerTransform(s, p + vec2( 0.0,-1.0)));
 vec4 L = 4.0*c - a - b - d - e;
 return vec4(c.rgb + amount * L.rgb, c.a);
} """

img = img.applyKernel(src, 2,
                      extent=(250, 500, 256, 256),
                      roi=lambda index, r: inset(r, -1, -1))
```

Demo

Source Available for Download

Core Image and Core ML

Core ML in Core Image

NEW

CICoreMLModelFilter

Core ML in Core Image

NEW

Input

CICoreMLModelFilter

Core ML in Core Image

NEW

Input

CICoreMLModelFilter

Model

Core ML in Core Image

NEW

CICoreMLModelFilter

MLMODEL

Input

Model

Output

Core ML in Core Image

NEW

```
let result = image.applyingFilter("CICoreMLModelFilter", parameters: ["inputModel": model])!
```

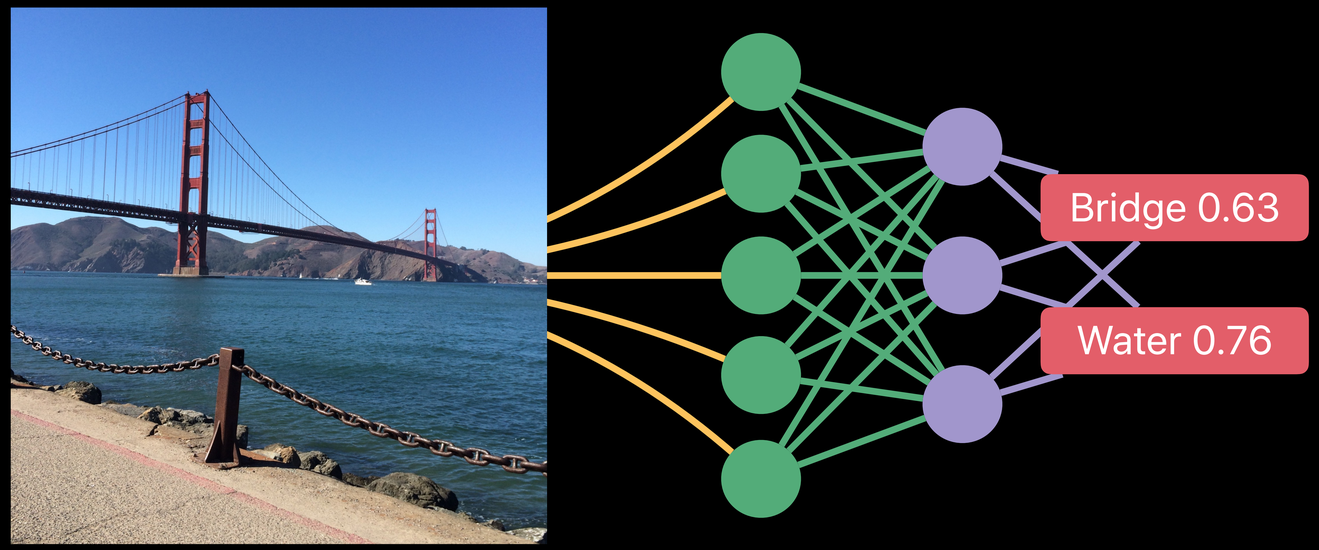
Core ML in Core Image

NEW

```
let result = image.applyingFilter("CICoreMLModelFilter", parameters: ["inputModel": model])!
```

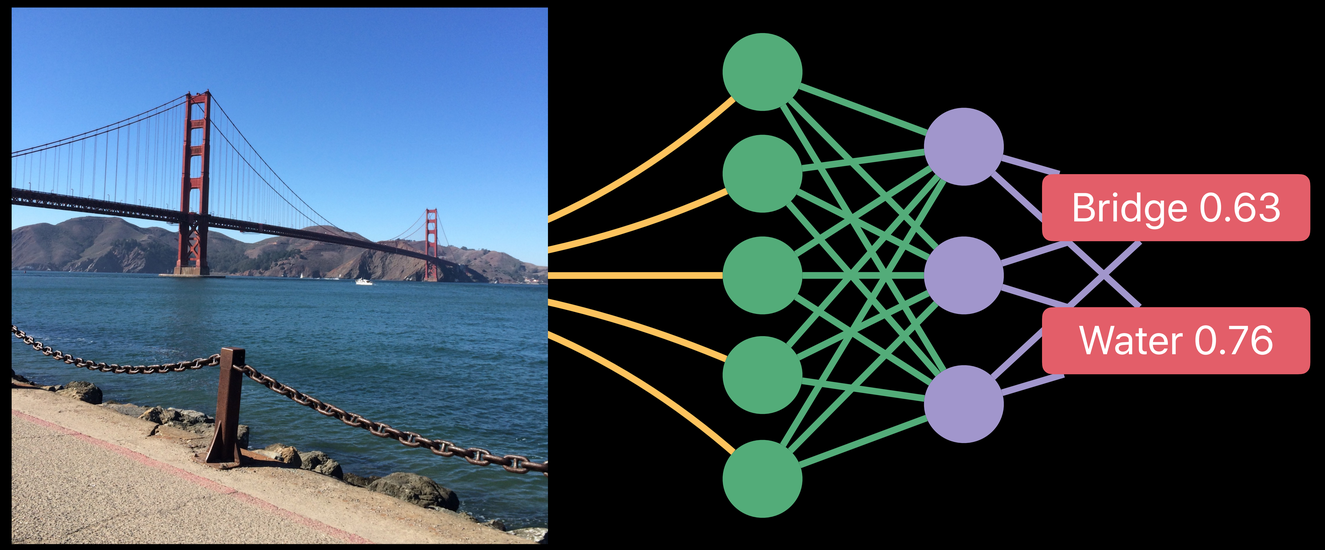
A Guide to Turi Create	WWDC 2018
Vision with Core ML	WWDC 2018

Data Augmentation



Data Augmentation

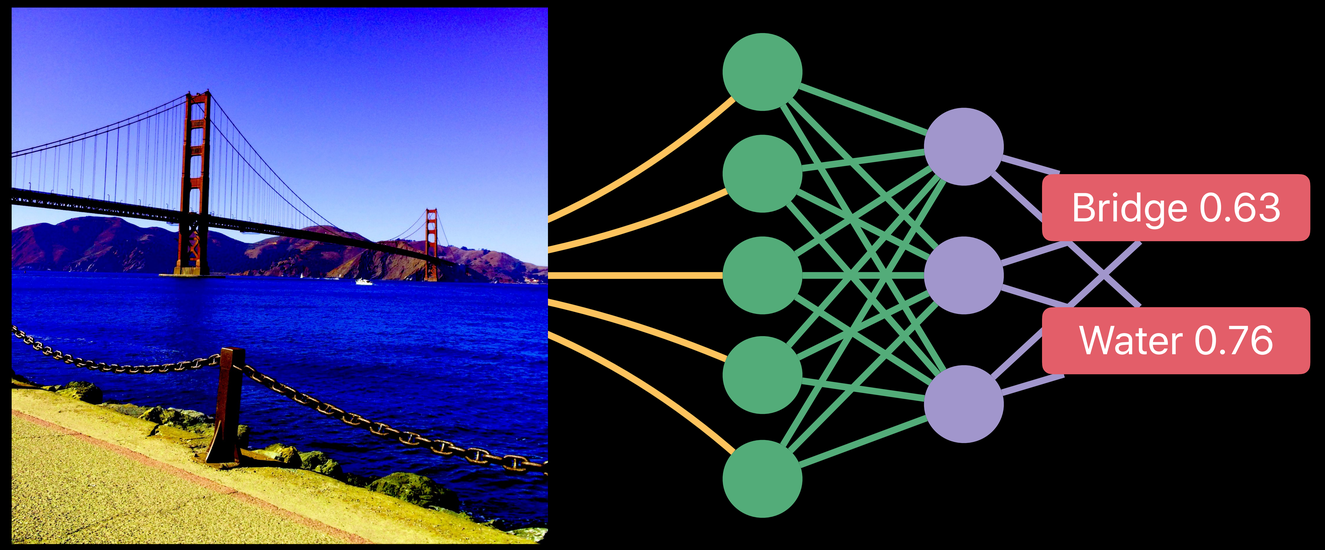
Robustness to image variations



Data Augmentation

Robustness to image variations

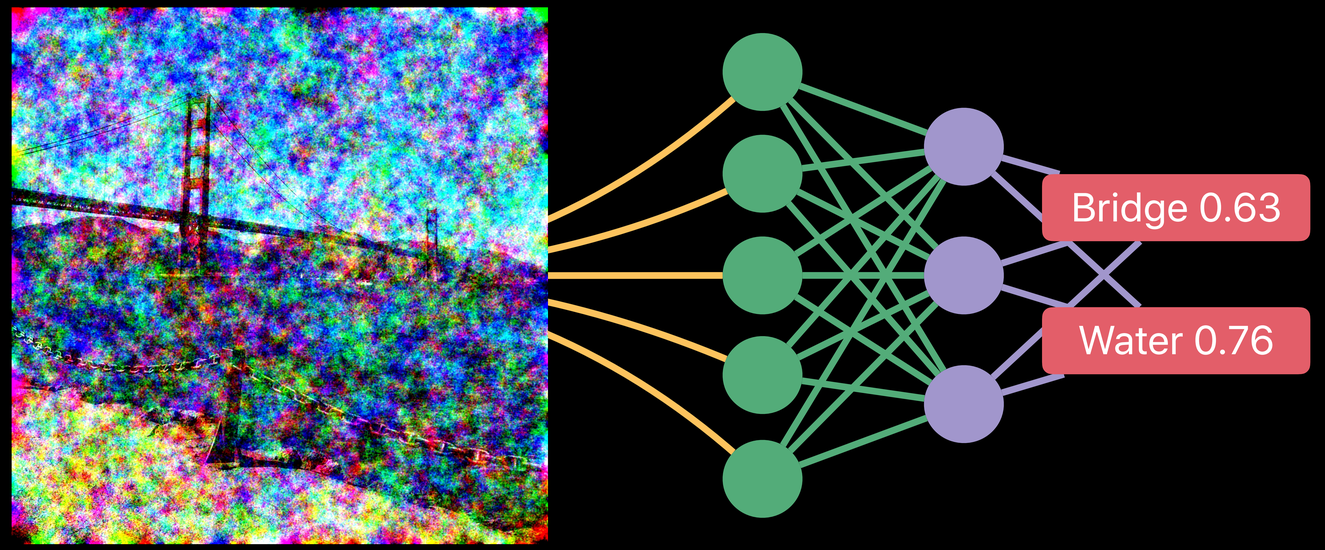
- Appearance



Data Augmentation

Robustness to image variations

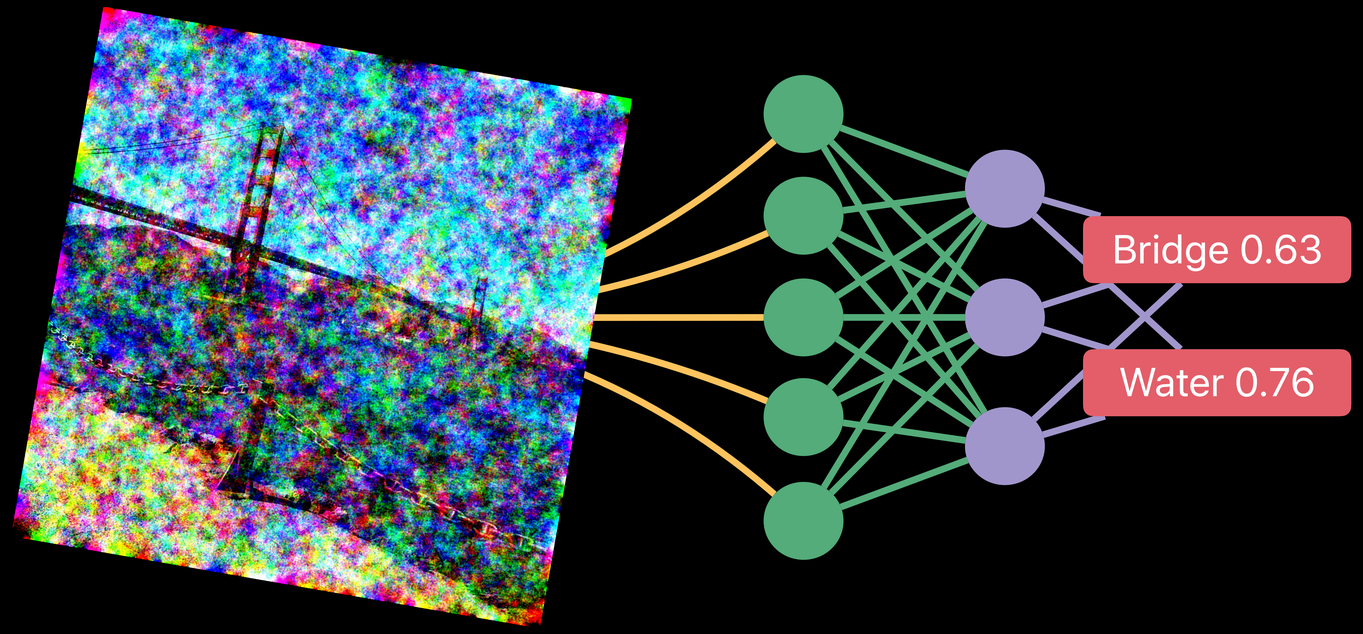
- Appearance
- Noise



Data Augmentation

Robustness to image variations

- Appearance
- Noise
- Geometry

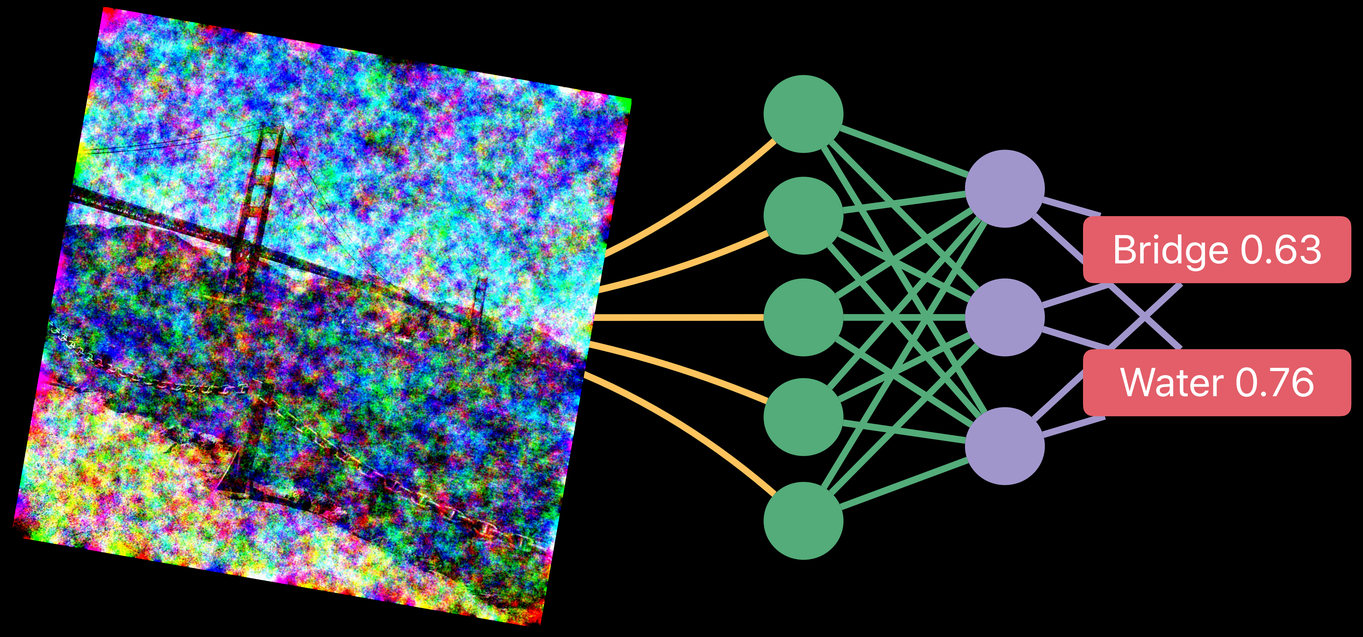


Data Augmentation

Robustness to image variations

- Appearance
- Noise
- Geometry

Trivial with Core Image

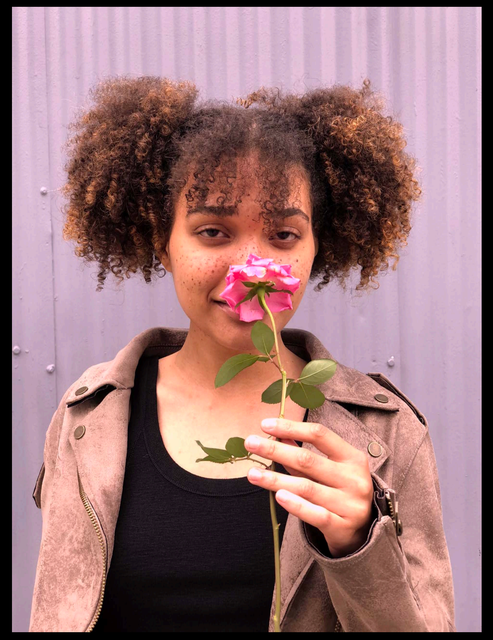


Data Augmentation

Input

Data Augmentation

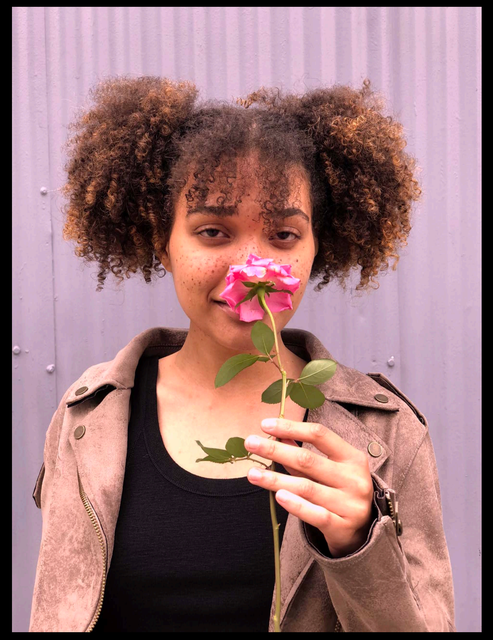
Input



CITemperatureAndTint

Data Augmentation

Input

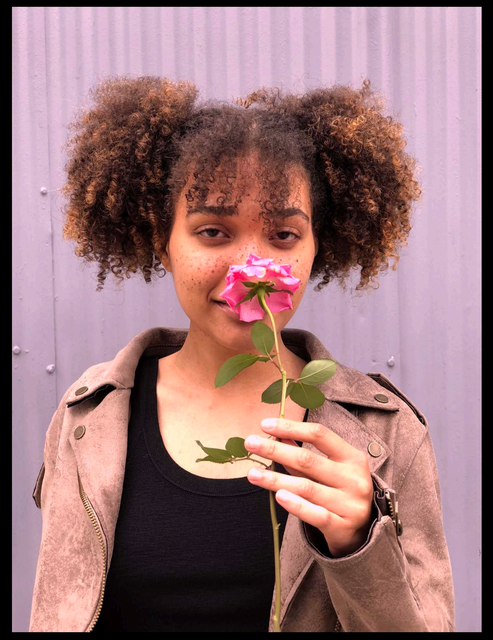


CITemperatureAndTint

CIColorControls

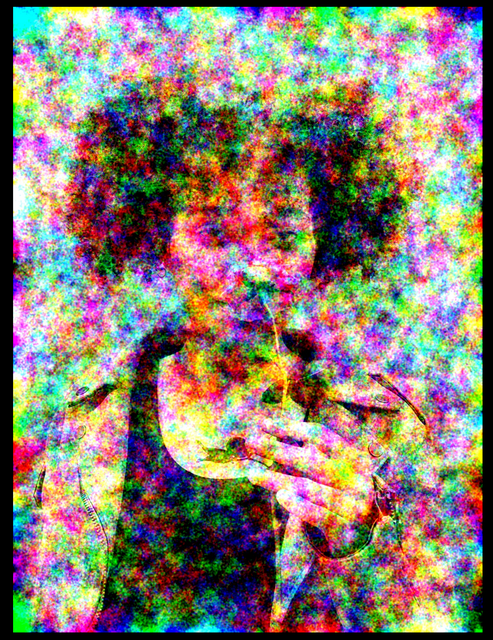
Data Augmentation

Input



CITemperatureAndTint

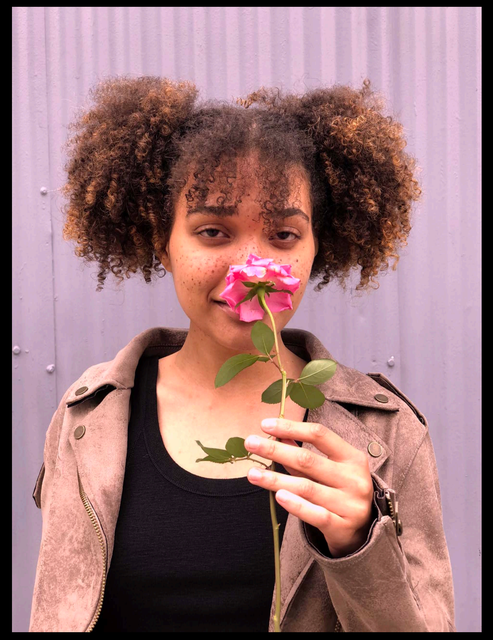
CIColorControls



CIDither

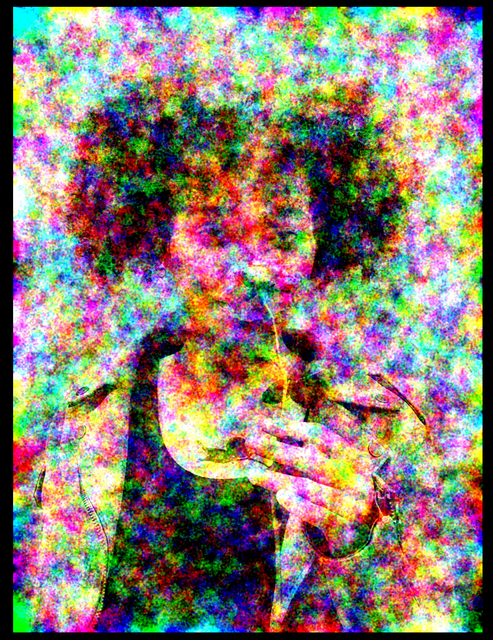
Data Augmentation

Input



CITemperatureAndTint

CIColorControls



CIDither

Affine

Demo

More Information

https://developer.apple.com/wwdc18/719

Core Image Lab	Technology Lab 2	Friday 3:00PM


WWDC18