IMAGE CONVOLUTION

LEARNING OBJECTIVES

- Learn about image convolutions and what makes them a good problem for solving on a GPU
- Learn what a naive image convolution may look like

IMAGE CONVOLUTION

Over the next few lectures we will be looking at some common GPU optimizations with an image convolution as the motivational example.

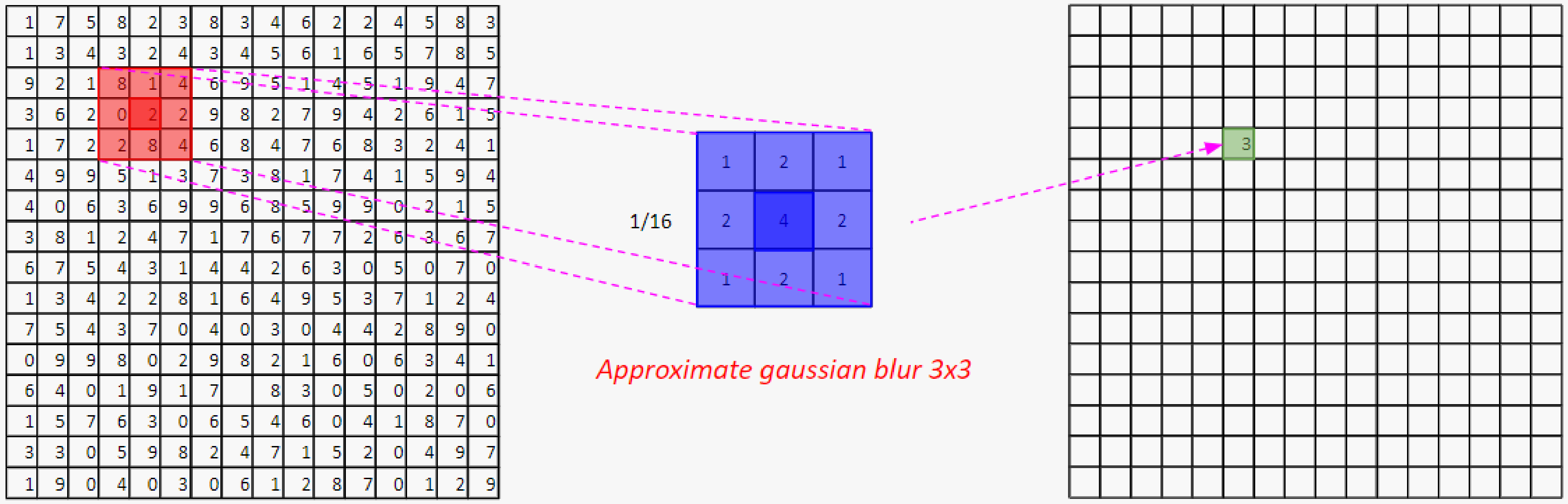
- A good problem to solve on a GPU.
- Can take advantage of a number of common optimizations.
- Convolution is a very powerful algorithm with many applications.
- Deep neural networks.
- Image processing.

WHY ARE IMAGE CONVOLUTIONS GOOD ON A GPU?

- The algorithm is **embarrassingly parallel**.
- Each work-item in the computation can be calculated entirely independently.
- The algorithm is computation heavy.
- A large number of operations are performed for each work-item in the computation, particularly when using large filters.
- The algorithm requires a large bandwidth.
- A lot of data must be passed through the GPU to process an image, particularly if the image is very high resolution.

IMAGE CONVOLUTION DEFINITION

$$G = h \otimes F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$



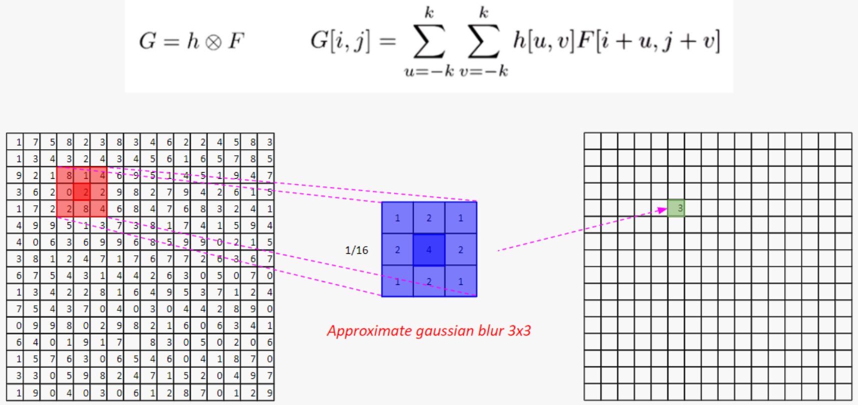
1	7	5	8	2	3	8	3	4	6	2	2	4	5	8	3
1	3	4	3	2	4	3	4	5	6	1	6	5	7	8	5
9	2	1	8	1	4	6	9	5	1	4	5	1	9	4	7
3	6	2	0	2	2	9	8	2	7	9	4	2	6	1	5
1	7	2	2	8	4	6	8	4	7	6	8	3	2	4	1
4	9	9	5	1	3	7	3	8	1	7	4	1	5	9	4
4	0	6	3	6	9	9	6	8	5	9	9	0	2	1	5
3	8	1	2	4	7	1	7	6	7	7	2	6	3	6	7
6	7	5	4	3	1	4	4	2	6	3	0	5	0	7	0
1	3	4	2	2	8	1	6	4	9	5	3	7	1	2	4
7	5	4	3	7	0	4	0	3	0	4	4	2	8	9	0
0	9	9	8	0	2	9	8	2	1	6	0	6	3	4	1
6	4	0	1	9	1	7		8	3	0	5	0	2	0	6
1	5	7	6	3	0	6	5	4	6	0	4	1	8	7	0
3	3	0	5	9	8	2	4	7	1	5	2	0	4	9	7
1	9	0	4	0	3	0	6	1	2	8	7	0	1	2	9

1/16

1	2	1
2	4	2
1	2	1

Approximate gaussian blur 3x3

IMAGE CONVOLUTION DEFINITION



$$G = h \otimes F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$

1	7	5	8	2	3	8	3	4	6	2	2	4	5	8	3
1	3	4	3	2	4	3	4	5	6	1	6	5	7	8	5
9	2	1	8	1	4	6	9	5	1	4	5	1	9	4	7
3	6	2	0	2	2	9	8	2	7	9	4	2	6	1	5
1	7	2	2	8	4	6	8	4	7	6	8	3	2	4	1
4	9	9	5	1	3	7	3	8	1	7	4	1	5	9	4
4	0	6	3	6	9	9	6	8	5	9	9	0	2	1	5
3	8	1	2	4	7	1	7	6	7	7	2	6	3	6	7
6	7	5	4	3	1	4	4	2	6	3	0	5	0	7	0
1	3	4	2	2	8	1	6	4	9	5	3	7	1	2	4
7	5	4	3	7	0	4	0	3	0	4	4	2	8	9	0
0	9	9	8	0	2	9	8	2	1	6	0	6	3	4	1
6	4	0	1	9	1	7		8	3	0	5	0	2	0	6
1	5	7	6	3	0	6	5	4	6	0	4	1	8	7	0
3	3	0	5	9	8	2	4	7	1	5	2	0	4	9	7
1	9	0	4	0	3	0	6	1	2	8	7	0	1	2	9

1/16

1	2	1
2	4	2
1	2	1

Approximate gaussian blur 3x3

- A filter of a given size is applied as a stencil to the position of each pixel in the input image.
- Each pixel covered by the filter is then multiples with the corresponding element in the filter.
- The result of these multiplications is then summed to give the resulting output pixel.
- Here we have a 3x3 gaussian blur approximation as an example.

IMAGE CONVOLUTION EXAMPLE

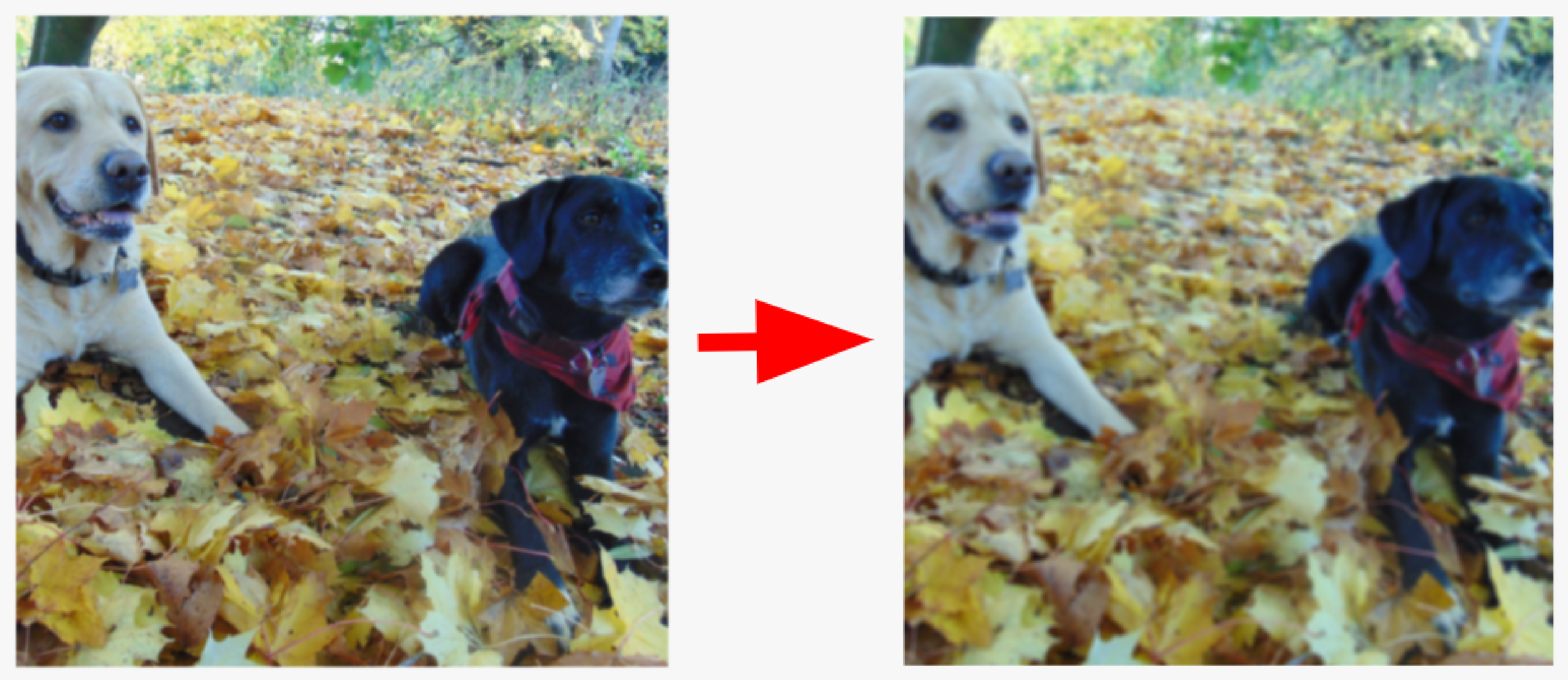
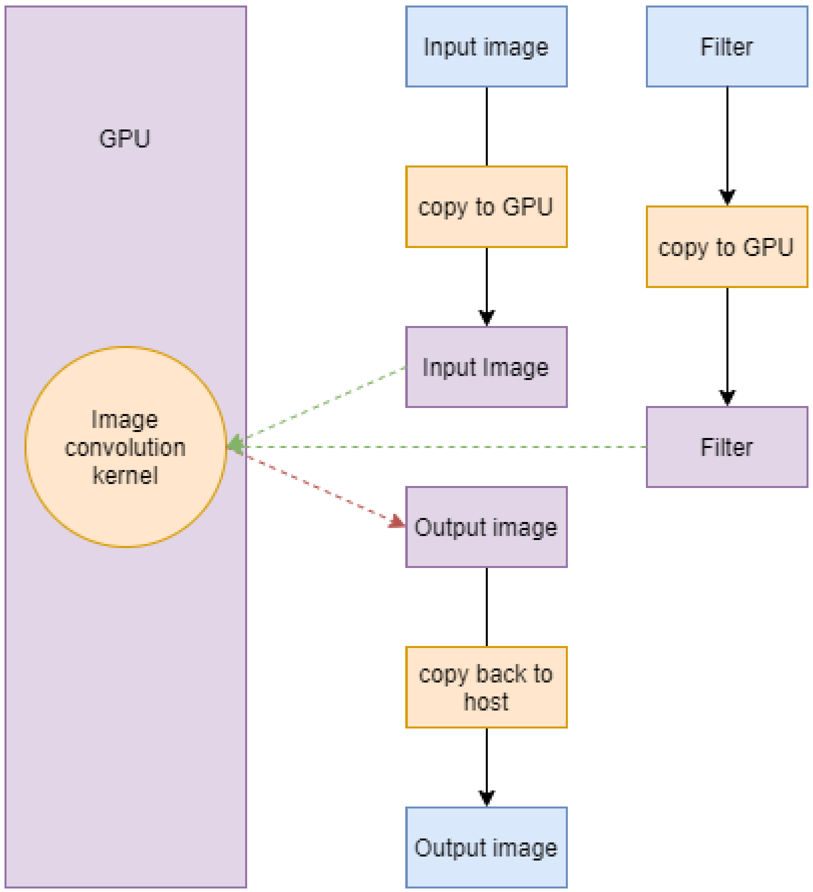


IMAGE CONVOLUTION DATA FLOW



- We have a single kernel function.
- It must read from the input image data and writes to the output image data.
- It must also read from the filter.
- The input image data and the filter don't need to be copied back to the host.
- The output image data can be uninitialized.

IMPLEMENTATION

- We provide a naive implementation of a SYCL application which implements the image convolution algorithm.
- This will be the basis for optimization in later lectures and exercises.
- The implementation uses the stb image library to allow us to visualize our results.
- The implementation also uses a benchmark function to allow us to measure the performance as we make optimizations.

REFERENCE IMAGE

- We provide a reference image to use in the exercise.
- This is in Code_Exercises/Images
- This image is a 512x512 RGBA png.
- Feel free to use your own image but we recommend keeping to this format.

INPUT/OUTPUT IMAGE LOCATIONS

```
const char* inputImageFile = "../Code_Exercises/Images/dogs.png";
const char* outputImageFile = ../Code_Exercises/Images/blurred_dogs.png";
```

- The reference code and the solutions to the remaining exercises use these strings to refernce the location of the input and output image.
- Before compiling these you will have to update this to point to the image in the development environment.

CONVOLUTION FILTERS

```
auto filter = util::generate_filter(util::filter_type filterType, int width);
```

- The utility for generating the filter data takes a `filter_type` enum which can be either `identity` or `blur` and a width.
- Feel free to experiment with different variations.
- Note that the filter width should always be an odd value.

NAIVE IMAGE CONVOLUTION PERFORMANCE

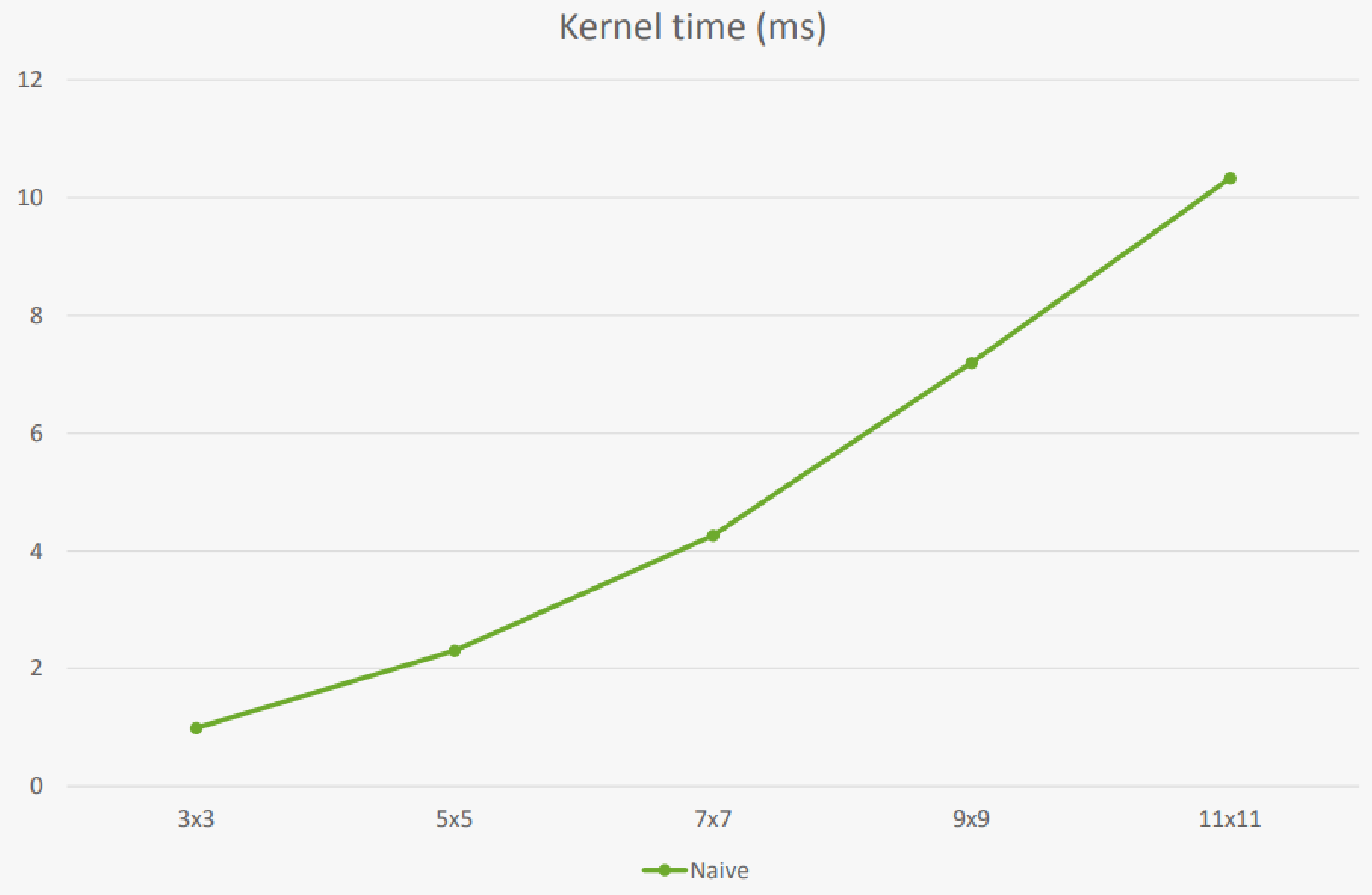


Image conv

512x512 source image

Intel HD Graphics 530

QUESTIONS

EXERCISE

Code_Exercises/Exercise_15_Image_Convolution/reference

Take a look at the naive image convolution code provided and familiarize yourself with it.