UNIVERSITY of DAYTON

Lab 4

CPS592 – Visual Computing and Mixed Reality

Preparation

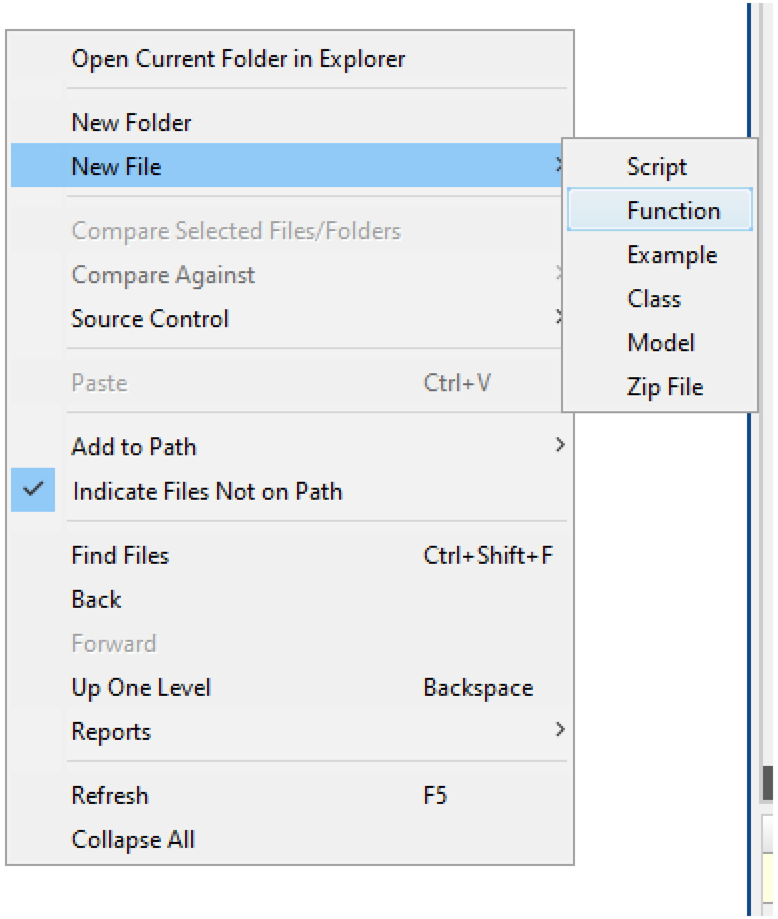
- Open MATLAB
- Create Lab4 folder
- Copy *mountain.jpg* and *bolt.jpg* to Lab4 folder

Copy script file

- Copy Lab3.m inside Lab4 folder

Create one function file

- Create new function *createRangeKernel*



Migrate the range kernel creation code to the new function file

```
function range_kernel  = createRangeKernel(img_gray, i, j, kernel_size, sigma_range)
%CREATERANGEKERNEL Summary of this function goes here
%   Detailed explanation goes here
    indent = (kernel_size - 1)/2;
    range_kernel = exp(-abs(img_gray(i - indent:i + indent,j - indent:j + indent )- img_gray(i,j)).^2/(sigma_range * sigma_range));
end
```

Update Lab3.m

```
for i = indent + 1:height - indent
  for j = indent + 1:width - indent
    %       range_kernel = exp(-abs(img_gray(i - indent:i + indent,j - indent:j + indent )-img_gray(i,j)).^2/(sigma_range * sigma_range));
    range_kernel = createRangeKernel(img_gray, i, j, kernel_size, sigma_range);
    kernel = range_kernel .* gaussian_kernel;
    normalization = 1/sum(kernel(:));
    temp = (kernel.*double(img_gray(i - indent:i + indent,j - indent:j + indent))) * normalization;
    img_results(i,j) = sum(temp(:));
  end
end
```

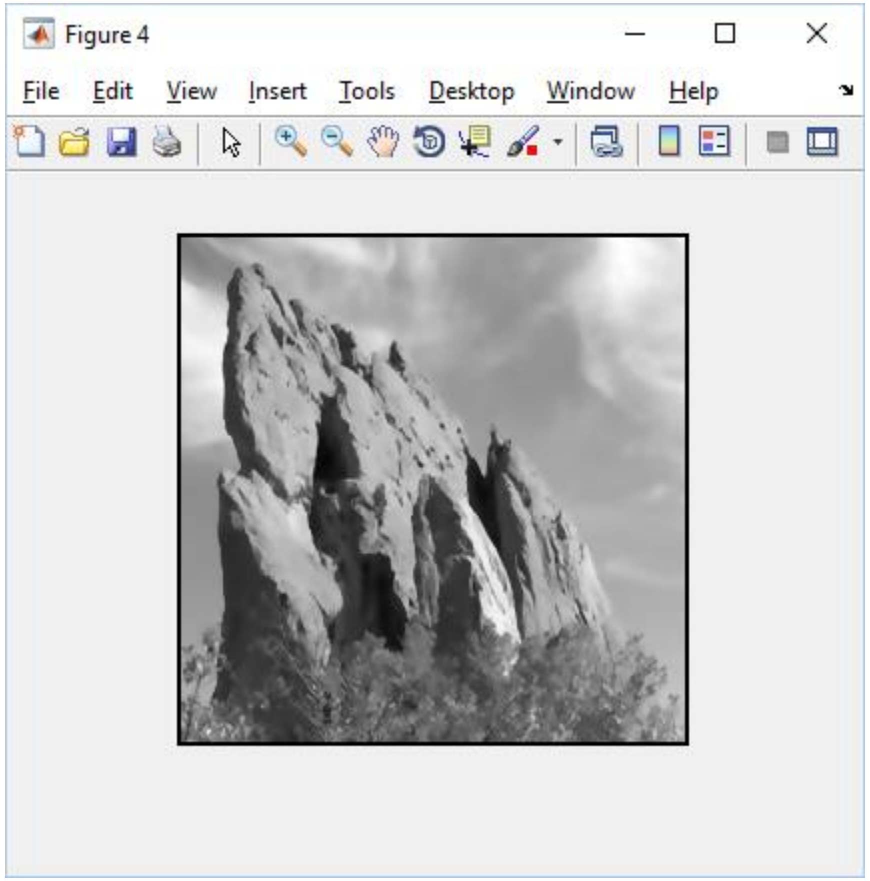
Create new function file: bilateralFilter

```
function img_results = bilateralFilter(img)
%BILATERALFILTER Summary of this function goes here
%   Detailed explanation goes here
img_gray = rgb2gray(img);
kernel_size = 5;
gaussian_kernel = fspecial('gaussian', [kernel_size kernel_size], 5);
img_gray_gaussian = imfilter(img_gray, gaussian_kernel, 'replicate');
% Preparation for BF
indent = (kernel_size - 1)/2;
[height, width] = size(img_gray);
img_results = zeros(height,width);
img_gray = double(img_gray);
sigma_range = 25;

for i = indent + 1:height - indent
    for j = indent + 1:width – indent
……………………………………………………………
```

MOVE ALL THE CODE OF APPLYING BILATERAL FILTER TO HERE

New Lab3.m

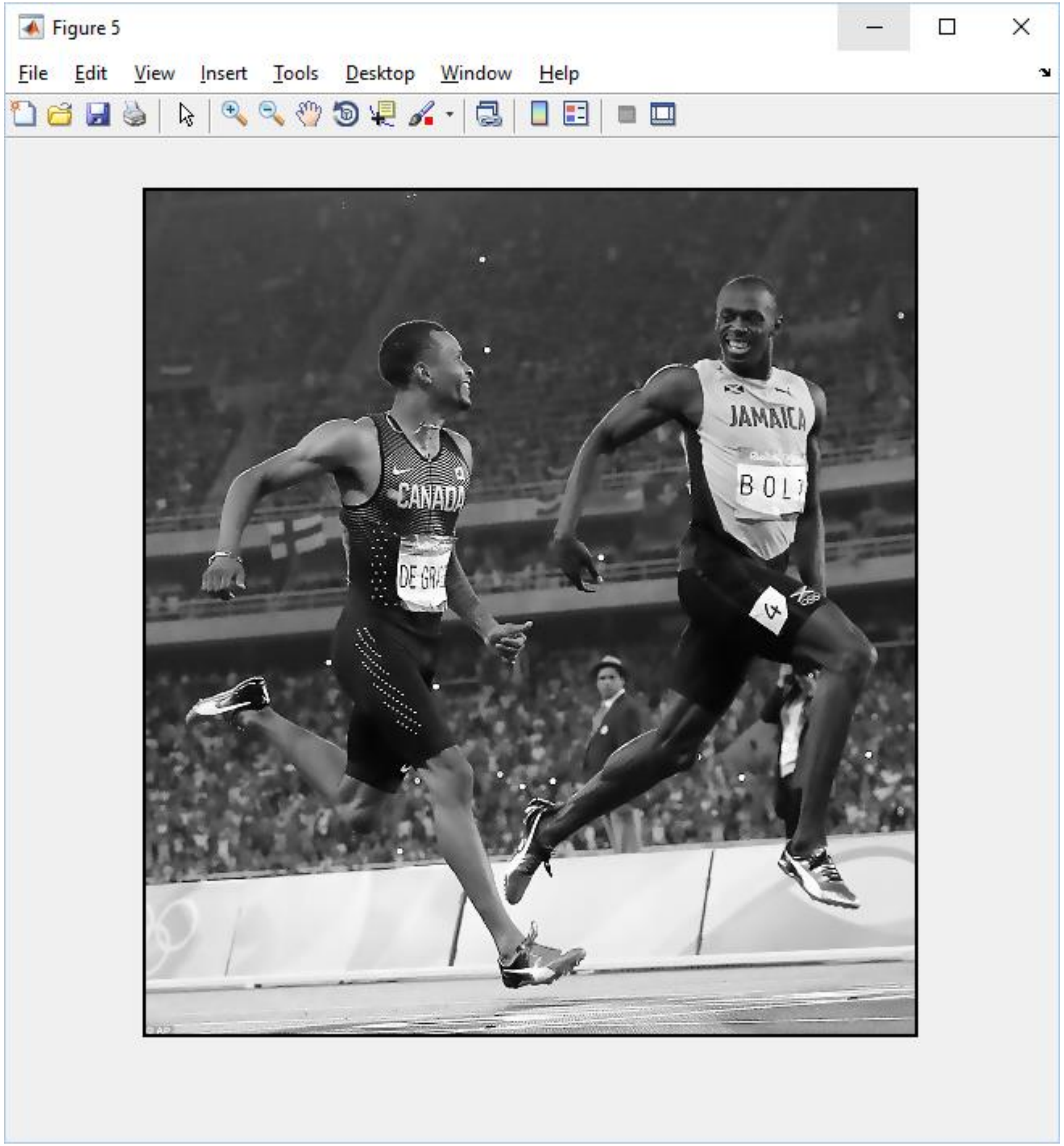
```
img = imread('mountain.jpg');
img_results = bilateralFilter(img);
figure, imshow(img_results,[]);
```



Try with bolt.jpg

```
img = imread('bolt.jpg');
img_results = bilateralFilter(img);
figure, imshow(img_results,[]);
```

DO WE WANT COLOR IMAGE?



Create Lab4.m

```
close all;
clear all;
clc;

img = imread('bolt.jpg');
img_results = bilateralFilter(img);
figure, imshow(img_results,[]);
```

Update img_results

```
img = imread('bolt.jpg');
img_results = zeros(size(img));
```

Update bilateralFilter call

```
for c = 1:3
    img_results(:,:,c) = bilateralFilter(img(:,:,c));
end
```

Update bilateralFilter function

```
% img_gray = rgb2gray(img);
img_gray = img;
```

Update Lab4.m

```
close all;
clear all;
clc;

img = imread('bolt.jpg');
img_results = zeros(size(img));
figure, imshow(img);
%%%
%%% Input the code of extracting edge image here
%%%
for c = 1:3
    img_results(:,:,c) = bilateralFilter(img(:,:,c));
end

figure, imshow(uint8(img_results));
```

Extracting the edge image

- Inside Lab4.m

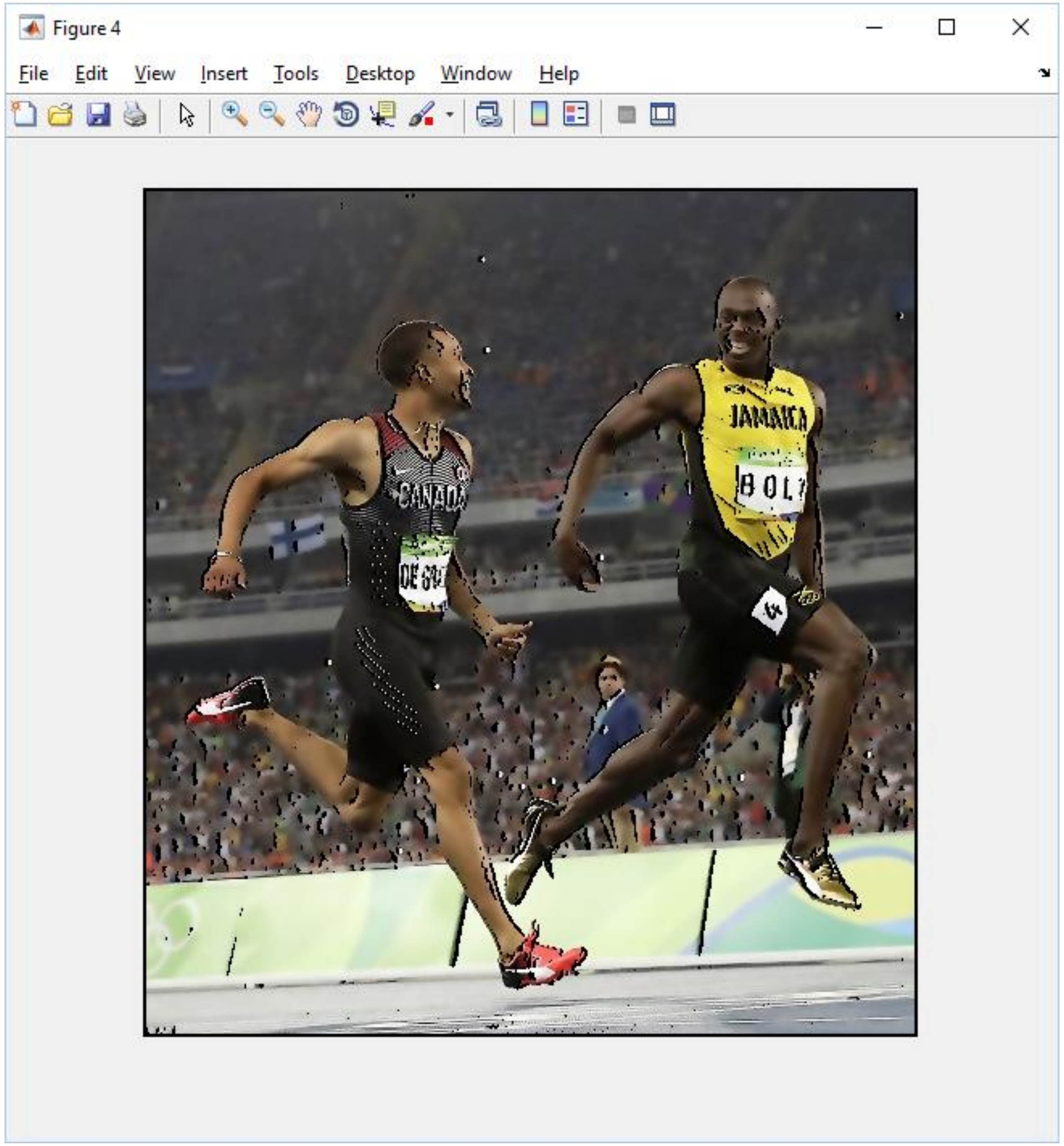
```
%%%
%%% Input the code of extracting edge image here
img_gray = rgb2gray(img);
kernel = [-1 0 1; -2 0 2; -1 0 1];
img_gray_sobel = imfilter(img_gray, kernel, 'replicate');
figure, imshow(img_gray_sobel,[]);

img_gray_sobel = double(1 - img_gray_sobel/max(img_gray_sobel(:)));
figure, imshow(img_gray_sobel);
%%%
```

Update the bilateralFilter call

```
for c = 1:3
%     img_results(:,:,c) = bilateralFilter(img(:,:,c));
    img_results(:,:,c) = bilateralFilter(img(:,:,c)).*img_gray_sobel;
end
```

Show the results

- What is this effect?



Q&A