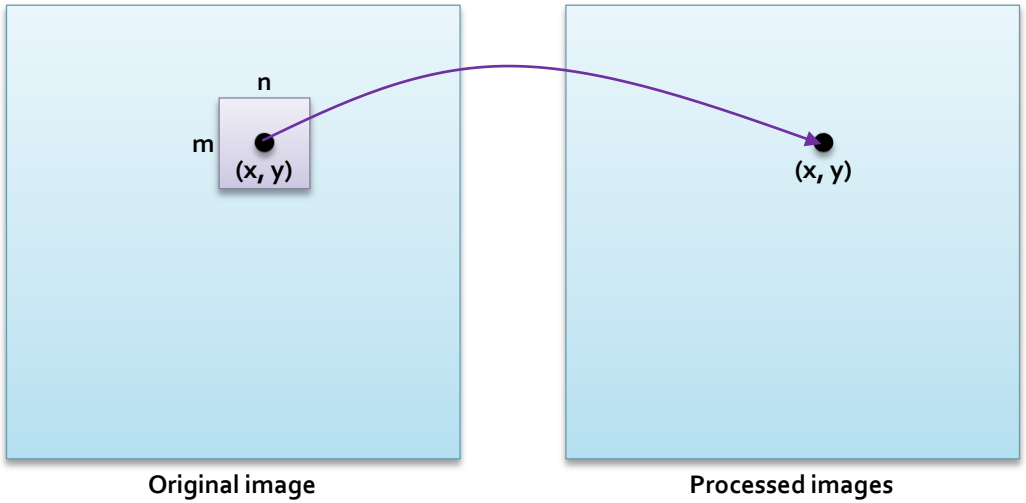
Simple Spatial Filter

2011.05.09

Jihye Yun

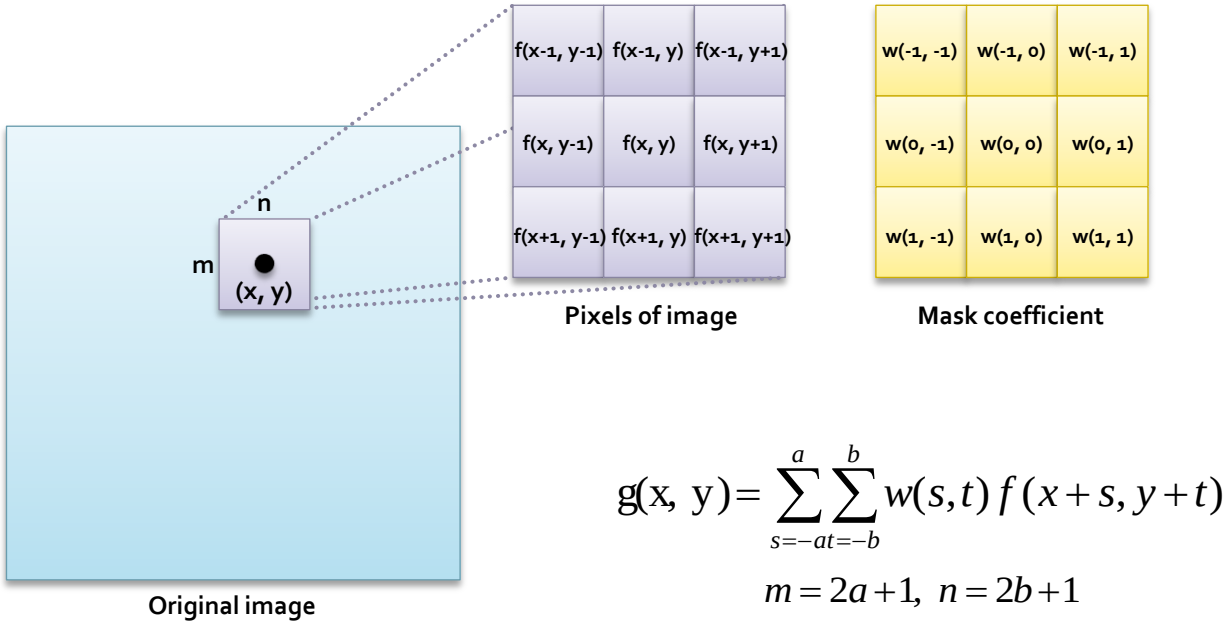
Spatial Filter

- Simply move filter mask from point to point
- Response of the filter is calculated using a pre-defined relationship



Spatial Filter

- Sum of products of the mask coefficients with the corresponding pixels directly under the mask



$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$m = 2a+1, \quad n = 2b+1$$

Mean Filter

- Simply average of the pixels contained in the neighborhood of the filter mask

Original image

Processed images

Mean Filter

- Simply average of the pixels contained in the neighborhood of the filter mask

$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

Standard average

$\frac{1}{16}\times$

1	2	1
2	4	2
1	2	1

Weighted average

$$g(x,\ y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

Mean Filter

- Simply average of the pixels contained in the neighborhood of the filter mask

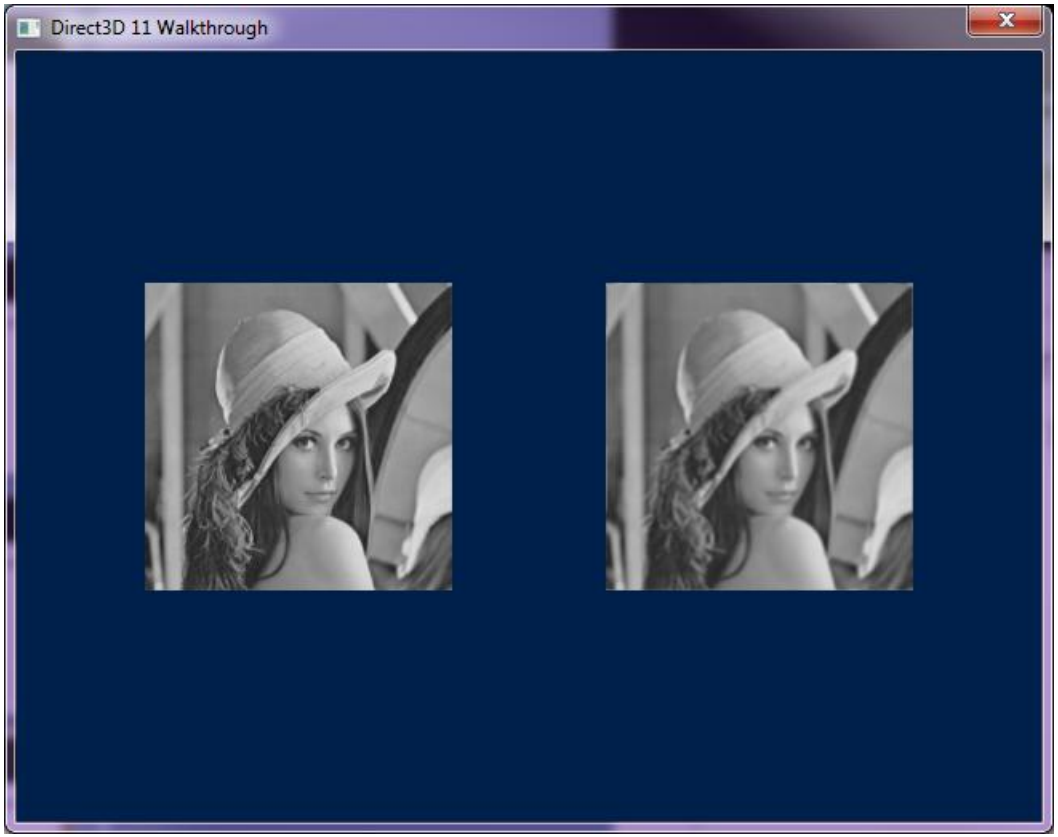
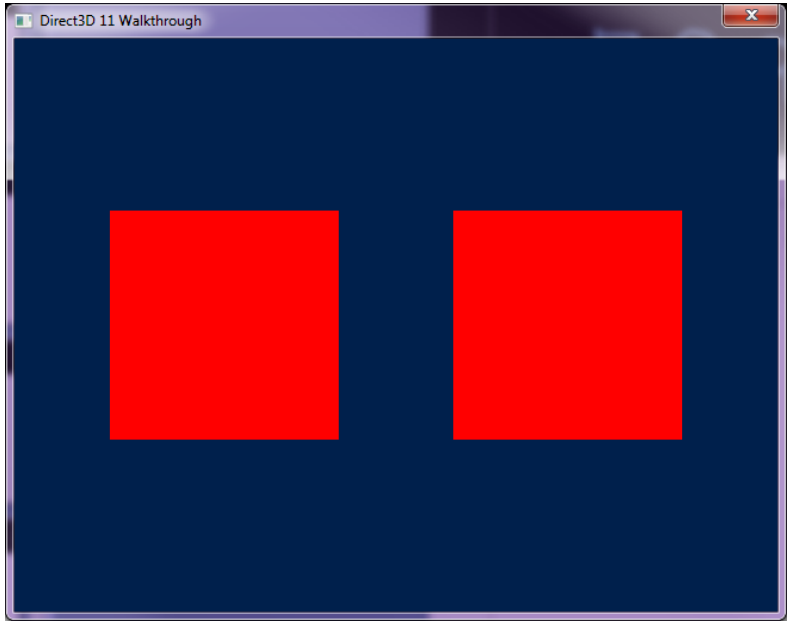


Image Plane

- Create two image-planes for input & output image
- Also Create individual pixel shaders

```
XMMATRIX matTrans = XMMatrixTranslation( x, y, z );
g_pImmediateContext->UpdateSubresource( g_pConstantBuffer, 0, NULL, &cb, 0, 0 );
```



Input image & Sampler State

- Load input image from file

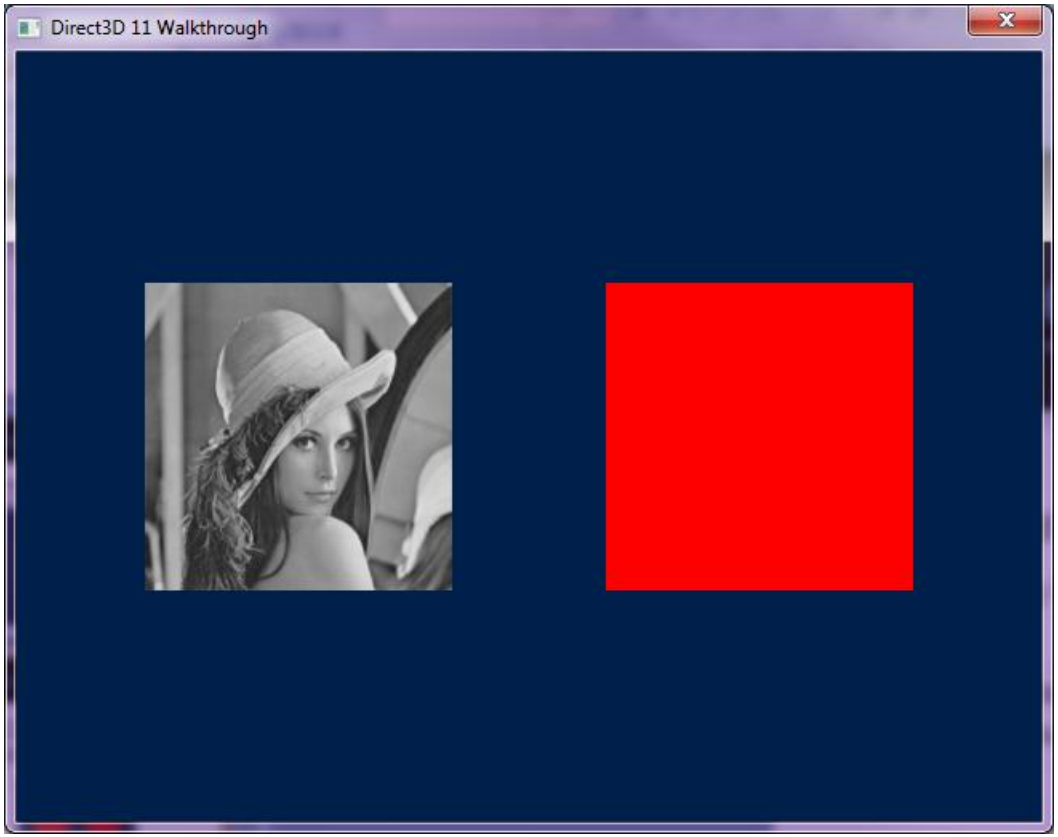
```
hr = D3DX11CreateShaderResourceViewFromFile( g_pd3dDevice, L"lena.bmp", NULL, NULL,
                                             &g_pTextureSRV, NULL);
if( FAILED(hr) )
        return hr;
```

- Create sampler state

```
D3D11_SAMPLER_DESC sampleDesc;
Zeromemory( & sampleDesc, sizeof(sampleDesc) );
sampleDesc.Filter = D3D11_FILTER_MIN_MAG_MIP_LINEAR;
sampleDesc.AddressU  = D3D11_TEXTURE_ADDRESS_WRAP;
sampleDesc.AddressV  = D3D11_TEXTURE_ADDRESS_WRAP;
sampleDesc.AddressW  = D3D11_TEXTURE_ADDRESS_WRAP;
sampleDesc.ComparisonFunc = D3D11_COMPARISON_NEVER;
sampleDesc.MinLOD = 0;
sampleDesc.MaxLOD = D3D11_FLOAT32_MAX;
hr = g_pd3dDevice->CreateSamplerState( &sampleDesc, &g_pSamplerLinear );
if ( FAILED(hr) )
        return hr;
```

Render Input Image

- Modify pixel shader for input image
  - Simply sample the loaded texture



Render Output Image

- Modify pixel shader for output image
  - Implement mean filter

```
float4 PS_Mean( VS_OUTPUT In ) : SV_Target
{
        // mask size = 3x3
        for j from -1 to 1 // for y-coordinate
        {
                for i from -1 to 1 // for x-coordinate
                {
                        sum += txInput.Sample( samplerLinear,
                                                float2(In.Tex.x+i*xDiff, In.Tex.y+j*yDiff) );
                }
        }

        return sum/maskSize;
}
```

Render Output Image

- Modify pixel shader for output image
  - Implement mean filter