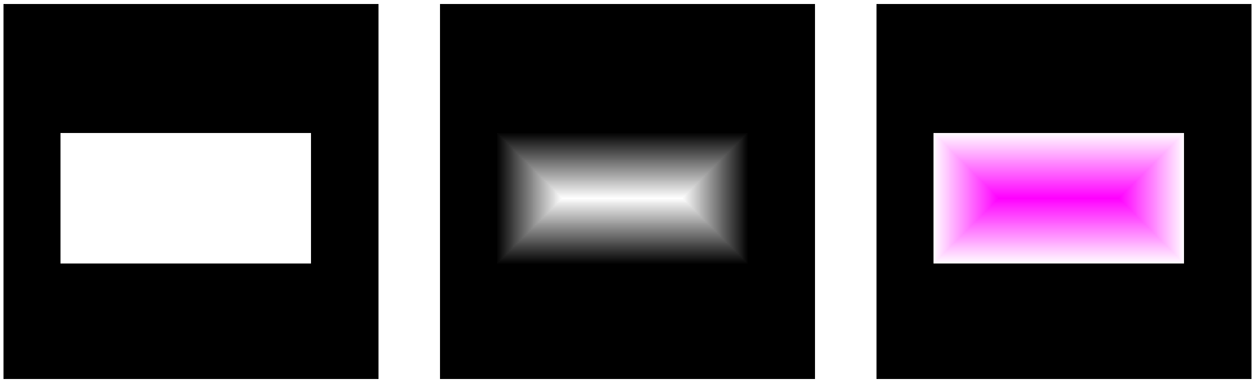
1 Problem 1



(a) Original Image

(b) Distance-Mapped Image

(c) Overlay Image

Figure 1: 2-Pass Distance Transform

In this problem, I applied the two-pass distance transformation as described in class. Figure 1a shows the original image. Figure 1b shows the output of the output of the distance transform. Figure 1c shows an overlay of the two to confirm appropriate output. The code used for this problem is contained in `problem1.py` and `distT.py`

`problem1.py`

```python
#Athanassios Athanassiadis Feb 2012
from scipy.misc import imread, imsave
from distT import *

im = imread('img_distance.tif')
im *= 1.0 / im.max()
dmap = dist_t(im)

im *= 1.0 * dmap.max()
imsave('5-1a.png',dmap)
imsave('5-1b.png',(im,im-dmap,im))
```

distT.py

```python
#Athanasios Athanassiadis Feb 2012
import numpy as np
inf = np.inf

#pad image with zeros
def pad_image(im, pad=1):
    newim = np.zeros(np.array(im.shape) + 2*pad)
    newim[pad:-pad,pad:-pad] = im.copy()

    return newim

#compute the distance transform of a binary image
def dist_t(im):
    #make a binary copy, thresholding at 0
    #make anything inside of the region infinity for the forward pass
    dmap = 1.0 * (pad_image(im) > 0)
    dmap[dmap==1] = inf

    #first pass (forward)
    for i in range(1,im.shape[0]):
        for j in range(1,im.shape[1]):
            dmap[i,j] = min(dmap[i,j],dmap[i-1,j]+1,dmap[i,j-1]+1)

    #second pass (reverse)
    for i in range(1,im.shape[0]+1)[::-1]:
        for j in range(1,im.shape[1]+1)[::-1]:
            dmap[i,j] = min(dmap[i,j],dmap[i+1,j]+1,dmap[i,j+1]+1)

    #remove padding
    return dmap[1:-1,1:-1]
```