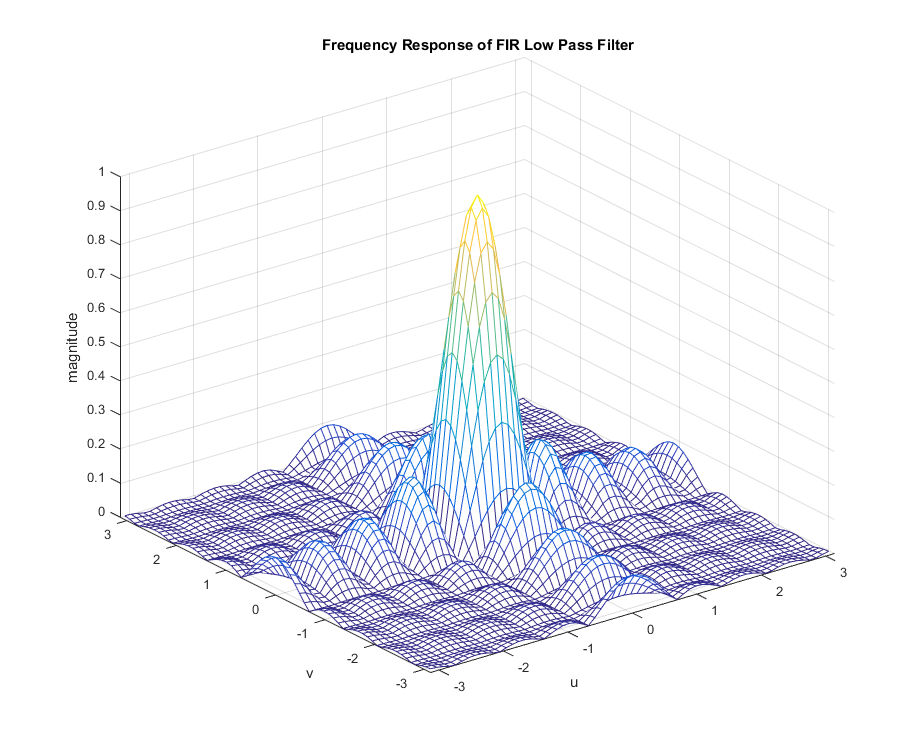
Lab 1 Report

# Section 3 FIR Low Pass Filter

1. A derivation of the analytical expression for .
2. A plot of .



1. The color image in imag03.tif



1. The filtered color image



1. A listing of C code

#include <math.h>

#include "tiff.h"

#include "allocate.h"

#include "randlib.h"

#include "typeutil.h"

void fir\_lowpass\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** double **\*\***kernel**,** int i**,** int j**,** int width**,** int height**,** int kernel\_size**);**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**);**

int main **(**int argc**,** char **\*\***argv**)**

**{**

FILE **\***fp**;**

struct TIFF\_img input\_img**,** filter\_img**;**

double **\*\***output**;**

int kernel\_size **=** 9**;**

double **\*\***kernel**;**

int32\_t i**,** j**;**

// check for argument count

**if** **(** argc **!=** 2 **)** **{**

fprintf**(** stderr**,** "Missing Argument\n"**);**

exit**(**1**);**

**}**

//check for error in reading files

**if** **(** **(** fp **=** fopen **(** argv**[**1**],** "rb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file %s\n"**,** argv**[**1**]** **);**

exit **(** 1 **);**

**}**

// check for reading tiff file

**if** **(**read\_TIFF**(**fp**,** **&**input\_img**))** **{**

fprintf**(** stderr**,** "error reading file %s\n"**,** argv**[**1**]** **);**

exit**(**1**);**

**}**

fclose**(**fp**);**

**if** **(**input\_img**.**TIFF\_type **!=** 'c'**)** **{**

fprintf **(** stderr**,** "error: image must be 24-bit color\n" **);**

exit **(** 1 **);**

**}**

//allocate memory

output **=** **(**double **\*\*)**get\_img**(**input\_img**.**width**,** input\_img**.**height**,** **sizeof(**double**));**

kernel **=** **(**double **\*\*)**get\_img**(**kernel\_size**,** kernel\_size**,** **sizeof(**double**));**

//create kernel

printf**(**"Create Kernel\n"**);**

**for** **(**i **=** 0**;** i **<** kernel\_size**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** kernel\_size**;** j**++)** **{**

kernel**[**i**][**j**]** **=** 1.0 **/** 81.0**;**

**}**

**}**

//apply the filter

printf**(**"Apply filter\n"**);**

printf**(**"Image size: %d %d\n"**,** input\_img**.**width**,** input\_img**.**height**);**

get\_TIFF**(** **&**filter\_img**,** input\_img**.**height**,** input\_img**.**width**,** 'c'**);**

**for** **(**int c **=** 0**;** c **<** 3**;** c**++){**

**for** **(**i **=** 0**;** i **<** input\_img**.**height**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** input\_img**.**width**;** j**++)** **{**

fir\_lowpass\_filter**(**input\_img**.**color**[**c**],** output**,** kernel**,** i**,** j**,**

input\_img**.**width**,** input\_img**.**height**,** kernel\_size**);**

**}**

**}**

printf**(**"Channel %d complete\n"**,** c**);**

apply\_color**(**filter\_img**,** output**,** c**);**

printf**(**"Applied channel %d color\n"**,** c**);**

**}**

/\* open image file for write \*/

**if** **(** **(** fp **=** fopen **(** "lowpass\_filter.tif"**,** "wb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file lowpass\_filter.tif\n"**);**

exit **(** 1 **);**

**}**

/\* write green image \*/

**if** **(** write\_TIFF **(** fp**,** **&**filter\_img **)** **)** **{**

fprintf **(** stderr**,** "error writing TIFF file %s\n"**,** argv**[**2**]** **);**

exit **(** 1 **);**

**}**

/\* close green image file \*/

fclose **(** fp **);**

/\* de-allocate memory \*/

free\_TIFF**(&(**input\_img**));**

free\_TIFF**(&(**filter\_img**));**

free\_img**((**void**\*\*)**output**);**

free\_img**((**void**\*\*)**kernel**);**

**}**

void fir\_lowpass\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** double **\*\***kernel**,** int i**,** int j**,** int width**,** int height**,** int kernel\_size**)**

**{**

double sum **=** 0.0**;**

**for** **(**int k **=** 0**;** k **<** kernel\_size**;** k**++)** **{**

**for** **(**int l **=** 0**;** l **<** kernel\_size**;** l**++)** **{**

int loc\_i **=** i **+** k **-** kernel\_size **/** 2**;**

int loc\_j **=** j **+** l **-** kernel\_size **/** 2**;**

**if** **(**loc\_i **>=** 0 **&&** loc\_i **<** height **&&** loc\_j **>=** 0 **&&** loc\_j **<** width**)** **{**

sum **+=** kernel**[**k**][**l**]** **\*** img**[**loc\_i**][**loc\_j**];**

**}**

**}**

**}**

output**[**i**][**j**]** **=** sum**;**

**}**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**)**

**{**

**for** **(**int i **=** 0**;** i **<** output**.**height**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** output**.**width**;** j**++)** **{**

int32\_t pixel **=** **(**int32\_t**)**input**[**i**][**j**];**

**if** **(**pixel **>** 255**)** **{**

pixel **=** 255**;**

**}**

output**.**color**[**channel**][**i**][**j**]** **=** **(**int32\_t**)**input**[**i**][**j**];**

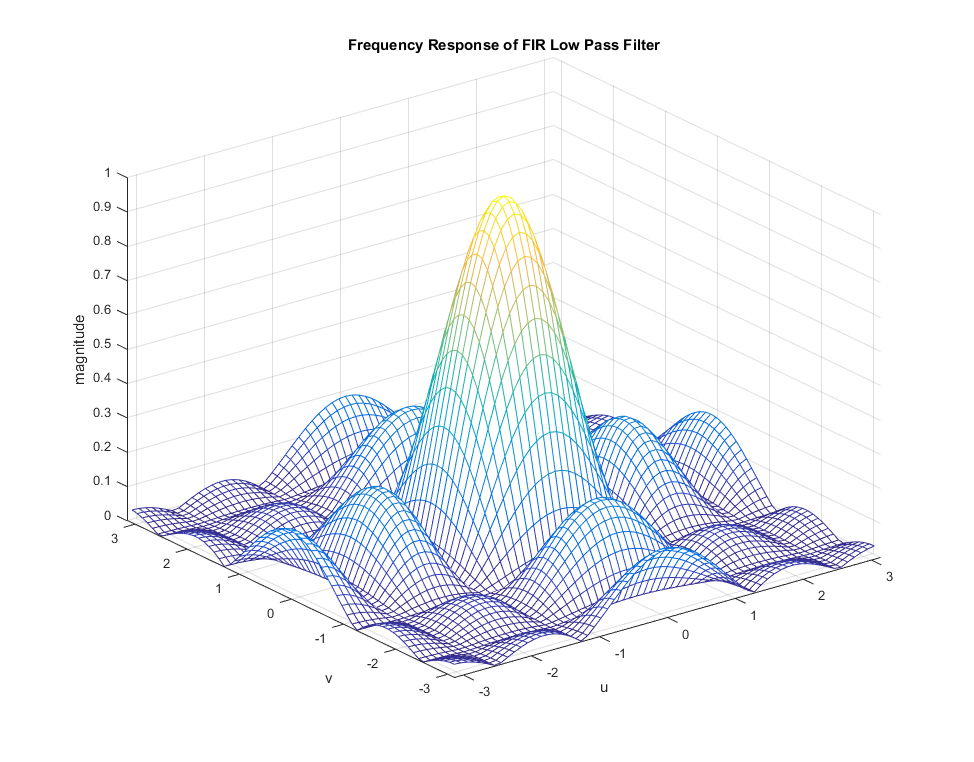
**}**

**}**

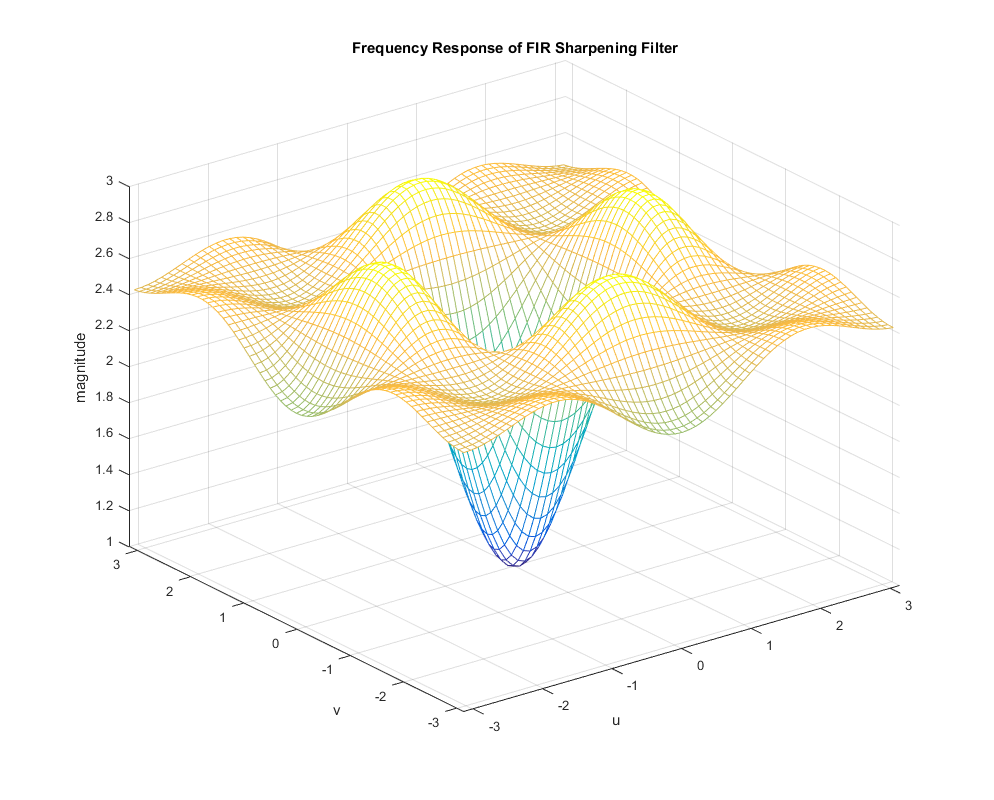
**}**

# Section 4 FIR Sharpening Filter

1. A derivation of the analytical expression for .
2. A derivation of the analytical expression for .
3. A plot of .



1. A plot of for



1. The input color image imgblur.tif.



1. The output sharpened color image for



1. A listing of C code.

#include <math.h>

#include "tiff.h"

#include "allocate.h"

#include "randlib.h"

#include "typeutil.h"

void fir\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** double **\*\***kernel**,** int i**,** int j**,** int width**,** int height**,** int kernel\_size**);**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**);**

int main **(**int argc**,** char **\*\***argv**)**

**{**

FILE **\***fp**;**

struct TIFF\_img input\_img**,** filter\_img**;**

double **\*\***output**;**

int kernel\_size **=** 5**;**

double **\*\***kernel**;**

int32\_t i**,** j**;**

double lambda**;**

// check for argument count

**if** **(** argc **!=** 3 **)** **{**

fprintf**(** stderr**,** "Missing Argument\n"**);**

exit**(**1**);**

**}**

//check for error in reading files

**if** **(** **(** fp **=** fopen **(** argv**[**1**],** "rb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file %s\n"**,** argv**[**1**]** **);**

exit **(** 1 **);**

**}**

// check for reading tiff file

**if** **(**read\_TIFF**(**fp**,** **&**input\_img**))** **{**

fprintf**(** stderr**,** "error reading file %s\n"**,** argv**[**1**]** **);**

exit**(**1**);**

**}**

fclose**(**fp**);**

**if** **(**input\_img**.**TIFF\_type **!=** 'c'**)** **{**

fprintf **(** stderr**,** "error: image must be 24-bit color\n" **);**

exit **(** 1 **);**

**}**

sscanf**(**argv**[**2**],** "%lf"**,** **&**lambda**);**

//allocate memory

output **=** **(**double **\*\*)**get\_img**(**input\_img**.**width**,** input\_img**.**height**,** **sizeof(**double**));**

kernel **=** **(**double **\*\*)**get\_img**(**kernel\_size**,** kernel\_size**,** **sizeof(**double**));**

//create kernel

printf**(**"Create Kernel\n"**);**

**for** **(**i **=** 0**;** i **<** kernel\_size**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** kernel\_size**;** j**++)** **{**

**if** **(**i **==** kernel\_size **/** 2 **&&** j **==** kernel\_size **/** 2**)** **{**

kernel**[**i**][**j**]** **=** 1.0 **+** lambda **\*** **(**1.0 **-** 1.0**/**25.0**);**

**}** **else** **{**

kernel**[**i**][**j**]** **=** lambda **\*** **(-**1.0**/**25.0**);**

**}**

**}**

**}**

**for** **(**i **=** 0**;** i **<** kernel\_size**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** kernel\_size**;** j**++)** **{**

printf**(**"%f,"**,** kernel**[**i**][**j**]);**

**}**

printf**(**"\n"**);**

**}**

//apply the filter

printf**(**"Apply filter\n"**);**

printf**(**"Image size: %d %d\n"**,** input\_img**.**width**,** input\_img**.**height**);**

get\_TIFF**(** **&**filter\_img**,** input\_img**.**height**,** input\_img**.**width**,** 'c'**);**

**for** **(**int c **=** 0**;** c **<** 3**;** c**++){**

**for** **(**i **=** 0**;** i **<** input\_img**.**height**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** input\_img**.**width**;** j**++)** **{**

fir\_filter**(**input\_img**.**color**[**c**],** output**,** kernel**,** i**,** j**,** input\_img**.**width**,** input\_img**.**height**,** kernel\_size**);**

**}**

**}**

printf**(**"Channel %d complete\n"**,** c**);**

apply\_color**(**filter\_img**,** output**,** c**);**

printf**(**"Applied channel %d color\n"**,** c**);**

**}**

/\* open image file for write \*/

**if** **(** **(** fp **=** fopen **(** "sharpen.tif"**,** "wb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file lowpass\_filter.tif\n"**);**

exit **(** 1 **);**

**}**

/\* write green image \*/

**if** **(** write\_TIFF **(** fp**,** **&**filter\_img **)** **)** **{**

fprintf **(** stderr**,** "error writing TIFF file %s\n"**,** argv**[**2**]** **);**

exit **(** 1 **);**

**}**

/\* close green image file \*/

fclose **(** fp **);**

/\* de-allocate memory \*/

free\_TIFF**(&(**input\_img**));**

free\_TIFF**(&(**filter\_img**));**

free\_img**((**void**\*\*)**output**);**

free\_img**((**void**\*\*)**kernel**);**

**}**

void fir\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** double **\*\***kernel**,** int i**,** int j**,** int width**,** int height**,** int kernel\_size**)**

**{**

double sum **=** 0.0**;**

**for** **(**int k **=** 0**;** k **<** kernel\_size**;** k**++)** **{**

**for** **(**int l **=** 0**;** l **<** kernel\_size**;** l**++)** **{**

int loc\_i **=** i **+** k **-** kernel\_size **/** 2**;**

int loc\_j **=** j **+** l **-** kernel\_size **/** 2**;**

**if** **(**loc\_i **>=** 0 **&&** loc\_i **<** height **&&** loc\_j **>=** 0 **&&** loc\_j **<** width**)** **{**

sum **+=** kernel**[**k**][**l**]** **\*** img**[**loc\_i**][**loc\_j**];**

**}**

**}**

**}**

output**[**i**][**j**]** **=** sum**;**

**}**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**)**

**{**

**for** **(**int i **=** 0**;** i **<** output**.**height**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** output**.**width**;** j**++)** **{**

int32\_t pixel **=** **(**int32\_t**)**input**[**i**][**j**];**

**if** **(**pixel **>** 255**)** **{**

pixel **=** 255**;**

**}**

output**.**color**[**channel**][**i**][**j**]** **=** **(**int32\_t**)**input**[**i**][**j**];**

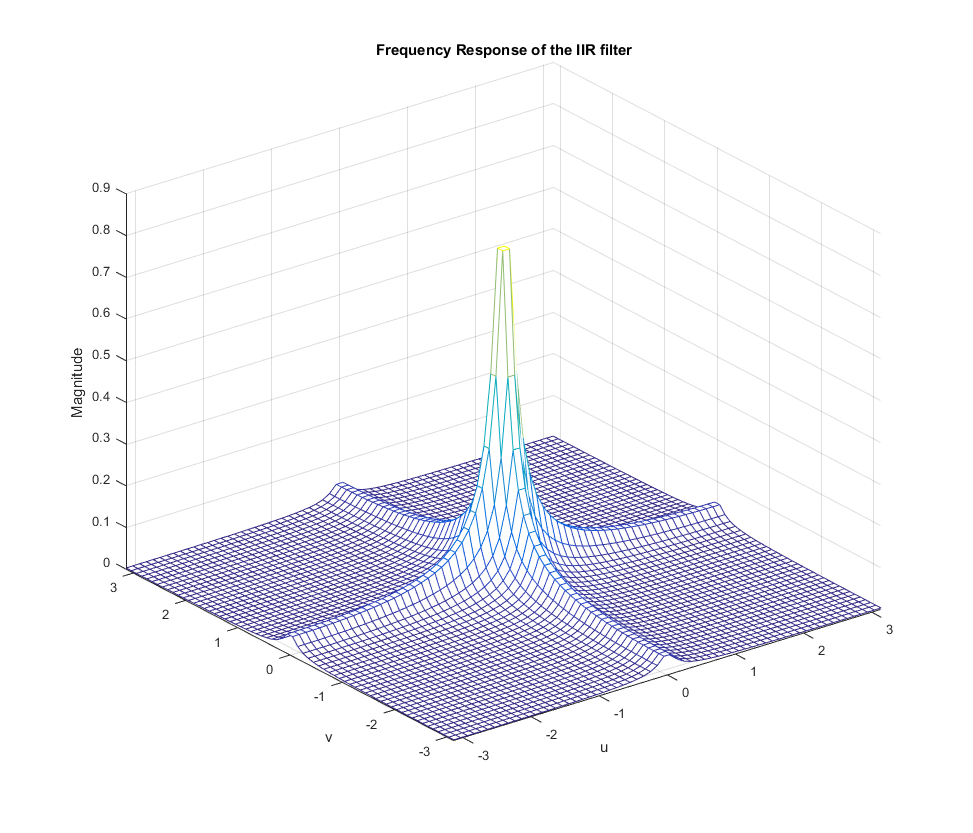
**}**

**}**

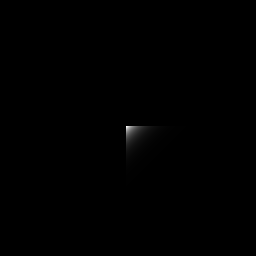
**}**

# Section 5 IIR Filter

1. A derivation of the analytical expression for .
2. A plot of .



1. An image of the point spread function.



1. The filtered output color image



1. A listing of C code.

#include <math.h>

#include "tiff.h"

#include "allocate.h"

#include "randlib.h"

#include "typeutil.h"

void iir\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** int i**,** int j**,** int width**,** int height**);**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**);**

int main **(**int argc**,** char **\*\***argv**)**

**{**

FILE **\***fp**;**

struct TIFF\_img input\_img**,** filter\_img**,** psf\_img**;**

double **\*\***output**;**

double **\*\***kernel**;**

int32\_t i**,** j**;**

// check for argument count

**if** **(** argc **!=** 3 **)** **{**

fprintf**(** stderr**,** "Missing Argument\n"**);**

exit**(**1**);**

**}**

//check for error in reading files

**if** **(** **(** fp **=** fopen **(** argv**[**1**],** "rb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file %s\n"**,** argv**[**1**]** **);**

exit **(** 1 **);**

**}**

// check for reading tiff file

**if** **(**read\_TIFF**(**fp**,** **&**input\_img**))** **{**

fprintf**(** stderr**,** "error reading file %s\n"**,** argv**[**1**]** **);**

exit**(**1**);**

**}**

fclose**(**fp**);**

**if** **(**input\_img**.**TIFF\_type **!=** 'c'**)** **{**

fprintf **(** stderr**,** "error: image must be 24-bit color\n" **);**

exit **(** 1 **);**

**}**

**if** **((**fp **=** fopen**(**argv**[**2**],** "rb"**))** **==** **NULL)** **{**

fprintf **(** stderr**,** "cannot open file %s\n"**,** argv**[**2**]** **);**

exit **(** 1 **);**

**}**

**if** **(**read\_TIFF**(**fp**,** **&**psf\_img**))** **{**

fprintf**(** stderr**,** "error reading file %s\n"**,** argv**[**2**]** **);**

exit**(**1**);**

**}**

fclose**(**fp**);**

//allocate memory

output **=** **(**double **\*\*)**get\_img**(**input\_img**.**width**,** input\_img**.**height**,** **sizeof(**double**));**

kernel **=** **(**double **\*\*)**get\_img**(**psf\_img**.**width**,** psf\_img**.**height**,** **sizeof(**double**));**

//covert tif to kernel

printf**(**"Create Kernel\n"**);**

**for** **(**i **=** 0**;** i **<** psf\_img**.**width**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** psf\_img**.**width**;** j**++)** **{**

kernel**[**i**][**j**]** **=** psf\_img**.**mono**[**i**][**j**]** **/** 255.0 **/** 100.0**;**

**}**

**}**

//apply the filter

printf**(**"Apply filter\n"**);**

printf**(**"Image size: %d %d\n"**,** input\_img**.**width**,** input\_img**.**height**);**

get\_TIFF**(** **&**filter\_img**,** input\_img**.**height**,** input\_img**.**width**,** 'c'**);**

**for** **(**int c **=** 0**;** c **<** 3**;** c**++){**

**for** **(**i **=** 0**;** i **<** input\_img**.**height**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<** input\_img**.**width**;** j**++)** **{**

iir\_filter**(**input\_img**.**color**[**c**],** output**,** i**,** j**,** input\_img**.**width**,** input\_img**.**height**);**

**}**

**}**

printf**(**"Channel %d complete\n"**,** c**);**

apply\_color**(**filter\_img**,** output**,** c**);**

printf**(**"Applied channel %d color\n"**,** c**);**

**}**

/\* open image file for write \*/

**if** **(** **(** fp **=** fopen **(** "iir\_filter.tif"**,** "wb" **)** **)** **==** **NULL** **)** **{**

fprintf **(** stderr**,** "cannot open file lowpass\_filter.tif\n"**);**

exit **(** 1 **);**

**}**

/\* write green image \*/

**if** **(** write\_TIFF **(** fp**,** **&**filter\_img **)** **)** **{**

fprintf **(** stderr**,** "error writing TIFF file %s\n"**,** argv**[**2**]** **);**

exit **(** 1 **);**

**}**

/\* close green image file \*/

fclose **(** fp **);**

/\* de-allocate memory \*/

free\_TIFF**(&(**input\_img**));**

free\_TIFF**(&(**filter\_img**));**

free\_TIFF**(&(**psf\_img**));**

free\_img**((**void**\*\*)**output**);**

free\_img**((**void**\*\*)**kernel**);**

**}**

void iir\_filter**(**uint8\_t **\*\***img**,** double **\*\***output**,** int i**,** int j**,** int width**,** int height**)**

**{**

double sum **=** 0.0**;**

sum **=** 0.01 **\*** img**[**i**][**j**];**

**if** **(**i **-** 1 **>=** 0**)** **{**

sum **+=** 0.9 **\*** output**[**i**-**1**][**j**];**

**}**

**if** **(**j **-** 1 **>=** 0**)** **{**

sum **+=** 0.9 **\*** output**[**i**][**j**-**1**];**

**}**

**if** **(**i **-** 1 **>=** 0 **&&** j **-** 1 **>=** 0**)** **{**

sum **-=** 0.81 **\*** output**[**i**-**1**][**j**-**1**];**

**}**

output**[**i**][**j**]** **=** sum**;**

**}**

void apply\_color**(**struct TIFF\_img output**,** double **\*\***input**,** int channel**)**

**{**

**for** **(**int i **=** 0**;** i **<** output**.**height**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** output**.**width**;** j**++)** **{**

int32\_t pixel **=** **(**int32\_t**)**input**[**i**][**j**];**

**if** **(**pixel **>** 255**)** **{**

pixel **=** 255**;**

**}**

output**.**color**[**channel**][**i**][**j**]** **=** **(**int32\_t**)**input**[**i**][**j**];**

**}**

**}**

**}**