Sprawozdanie Grafika komputerowa i wizualizacja

Przekształcenia w obrazach cyfrowych

Kamil Brzozowski
gr.1,nr. indexu: 134891,
Informatyka WIMII

Filtracją obrazu nazywamy proces przetwarzania obrazu cyfrowego w celu redukcji niepożądanych efektów(zmniejszenie szumu),czy tez poprawy jakości, np. wyostrzenie. Omawianą filtrację stosuje się ją w wielu dziedzinach, takich jak medycyna, analiza obrazów satelitarnych, grafika komputerowa i przemysłowe przetwarzanie obrazów. Filtracja obrazu wykorzystuje różne rodzaje filtrów, takie jak dolnoprzepustowy, medianowy, Laplace'a, Sobel'a itp. Wybór odpowiedniego filtra zależy od konkretnego zastosowania i sytuacji.

Wybrane filtry:

Filtr Sobel'a *jest to rodzaj filtru używanego w przetwarzaniu obrazów do wykrywania krawędzi. Wykorzystuje dwa operatory gradientowe - jeden dla krawędzi poziomych, drugi dla krawędzi pionowych. Działa przez obliczenie magnitudy gradientu obrazu, co pozwala wykrywać krawędzie w różnych kierunkach i kształtach. Filtr Sobel'a znajduje zastosowanie w wykrywaniu krawędzi, wyostrzaniu obrazów, redukcji szumu, analizie obrazów medycznych oraz w grafice komputerowej.*

Filtr Laplace'a *jest używany w przetwarzaniu obrazów do wykrywania krawędzi i zmian w intensywności pikseli. Stosuje operator Laplace'a, który wykrywa nagłe zmiany jasności. Może poprawić ostrość, wydobyć detale i redukować szumy. Jednak jest wrażliwy na szum, co może prowadzić do błędnych wykryć krawędziowych. Wymaga czasem dodatkowych technik filtracji.*

Filtr dolnoprzepustowy *jest używany w przetwarzaniu obrazów cyfrowych. Działa przez wygładzanie obrazu i usuwanie szybko zmieniających się detali. To tłumi wysokie częstotliwości, czyli szybkie zmiany wartości pikseli. Efektem jest mniej skomplikowany i ujednolicony obraz. Filtr dolnoprzepustowy znajduje zastosowanie w medycynie, grafice komputerowej, analizie obrazów satelitarnych i wielu innych dziedzinach. Jest podstawowym narzędziem w obróbce cyfrowych obrazów i często stosowany w połączeniu z innymi filtrami, aby uzyskać pożądane efekty.*

Filtr medianowy *stosuje się do redukcji szumu w obrazach cyfrowych poprzez zastępowanie wartości pikseli medianą z ich otoczenia. Jest*

skuteczny w usuwaniu szumów impulsowych i znajduje zastosowanie głównie w medycynie, eliminując zakłócenia z obrazów diagnostycznych. Jednak może być obliczeniowo kosztowny, co ogranicza jego zastosowanie w czasie rzeczywistym przy dużych obrazach lub złożonych operacjach przetwarzania obrazów.

Oryginalne zdjęcie do przekształceń

filtr czerwony

filtr zielony

filtr niebieski

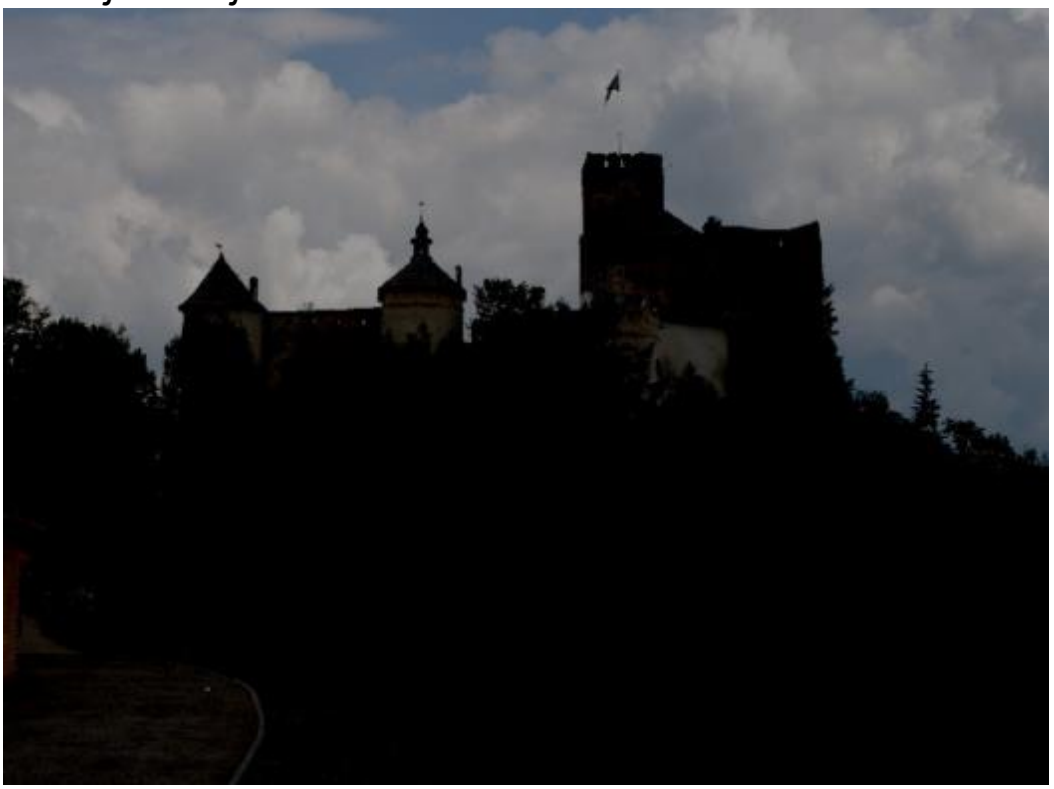
filtr Sobel’a

filtr dolnoprzepustowy uśredniający

Filtr max

Zwiększenie jasności

Zmniejszenie jasności



kod CPP

```
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#include <iostream>
#include <fstream>
#endif
#include <stdlib.h>

static int slices = 8,stacks = 8;
const int w=1339,k=2000;
int Rs[w][k];
int Gs[w][k];
int Bs[w][k];
int Rn[w][k];
int Gn[w][k];
int Bn[w][k];
int m1[3][3] = {1,1,1,1,1,1,1,1,1};
float lw,lk;

using namespace std;
static void resize(int width, int height)
{
const float ar = (float) width / (float) height;
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
// glFrustum(-ar, ar, -1.0, 1.0, 2.0, 100.0);
glOrtho(0,k,0,w,2.0,100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity() ;
}
static void display(void)
{
const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
const double a = t*90.0;
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glColor3d(1,0,0);
glPushMatrix();
glTranslated(0,0,-6);
glBegin(GL_POINTS);
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
glColor3f(Rs[i][j]/255.,Gs[i][j]/255.,Bs[i][j]/255.);
glVertex3d(j,i,0);
}
glEnd();
glPopMatrix();
glutSwapBuffers();
}
static void key(unsigned char key, int x, int y)
{
switch (key)
{
case 27 :
case 'q':
;
break;
case 'r':
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=Rn[i][j];
Gs[i][j]=Gn[i][j];
Bs[i][j]=Bn[i][j];
}
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Gs[i][j]=0;
Bs[i][j]=0;
}
break;
case 'g':
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=Rn[i][j];
Gs[i][j]=Gn[i][j];
Bs[i][j]=Bn[i][j];
```

```
}
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=0;
Bs[i][j]=0;
}
break;
case 'b':
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=Rn[i][j];
Gs[i][j]=Gn[i][j];
Bs[i][j]=Bn[i][j];
}
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=0;
Gs[i][j]=0;
}
break;
case 'x':
for(int i=0; i<lw; ++i)
for(int j=0; j<lk; ++j)
{
Rn[i][j] += 20;
Gn[i][j] += 20;
Bn[i][j] += 20;
}
for(int i=0; i<lw; ++i)
for(int j=0; j<lk; ++j)
{
R[i][j] = Rn[i][j];
G[i][j] = Gn[i][j];
B[i][j] = Bn[i][j];
}
break;
case 'c':
for(int i=0; i<lw; ++i)
for(int j=0; j<lk; ++j)
{
Rn[i][j] -= 20;
Gn[i][j] -= 20;
```

```
Bn[i][j] -= 20;
}
for(int i=0; i<lw; ++i)
for(int j=0; j<lk; ++j)
{
R[i][j] = Rn[i][j];
G[i][j] = Gn[i][j];
B[i][j] = Bn[i][j];
}
break;
case 'z':
for(int i=0;i<lw;++i)
for(int j=0;j<lk;++j)
{
Rs[i][j]=Rn[i][j];
Gs[i][j]=Gn[i][j];
Bs[i][j]=Bn[i][j];
}
break;
case 'm':
for(int i=1;i<lw-1;++i)
for(int j=1;j<lk-1;++j)
{
Rs[i][j]=(Rs[i-1][j-1]*m1[0][0]+Rs[i-1][j]*m1[0][1]+Rs[i-1]
[j+1]*m1[0][2]+Rs[i][j-1]*m1[1][0]+Rs[i][j]*m1[1][1]+Rs[i][j+1]*m1[1]
[2]+Rs[i+1][j-1]*m1[2][0]+Rs[i+1][j]*m1[2][1]+Rs[i+1][j+1]*m1[2][2])/9;
Bs[i][j]=(Bs[i-1][j-1]*m1[0][0]+Bs[i-1][j]*m1[0][1]+Bs[i-1]
[j+1]*m1[0][2]+Bs[i][j-1]*m1[1][0]+Bs[i][j]*m1[1][1]+Bs[i][j+1]*m1[1]
[2]+Bs[i+1][j-1]*m1[2][0]+Bs[i+1][j]*m1[2][1]+Bs[i+1][j+1]*m1[2][2])/9;
Gs[i][j]=(Gs[i-1][j-1]*m1[0][0]+Gs[i-1][j]*m1[0][1]+Gs[i-1]
[j+1]*m1[0][2]+Gs[i][j-1]*m1[1][0]+Gs[i][j]*m1[1][1]+Gs[i][j+1]*m1[1]
[2]+Gs[i+1][j-1]*m1[2][0]+Gs[i+1][j]*m1[2][1]+Gs[i+1][j+1]*m1[2][2])/9;
}
break;
case 'p':
{
int M[3][3]={{1,2,1},{0,0,0},{-1,-2,-1}};
for(int i=1;i<lw-1;++i)
for(int j=1;j<lk-1;++j)
{
Rn[i][j]=(R[i-1][j-1]M[0][0]+R[i-1][j]M[0][1]+R[i-1][j+1]M[0][2]+ R[i ][j-
1]M[1][0]+R[i ][j]M[1][1]+R[i ][j+1]M[1][2]+ R[i+1][j-1]M[2][0]+R[i+1]
[j]M[2][1]+R[i+1][j+1]M[2][2]) /( (M[0][0]+M[0][1]+M[0][2]+M[1]
[0]+M[1][1]+M[1][2]+M[2][0]+M[2][1]+M[2][2])==0 ? 1 : (M[0]
[0]+M[0][1]+M[0][2]+M[1][0]+M[1][1]+M[1][2]+M[2][0]+M[2]
```

```
[1]+M[2][2]) ); Gn[i][j]=(G[i-1][j-1]M[0][0]+G[i-1][j]M[0][1]+G[i-1]
[j+1]M[0][2]+ G[i ][j-1]M[1][0]+G[i ][j]M[1][1]+G[i ][j+1]M[1][2]+
G[i+1][j-1]M[2][0]+G[i+1][j]M[2][1]+G[i+1][j+1]M[2][2]) /( (M[0]
[0]+M[0][1]+M[0][2]+M[1][0]+M[1][1]+M[1][2]+M[2][0]+M[2][1]+M[2]
[2])==0 ? 1 : (M[0][0]+M[0][1]+M[0][2]+M[1][0]+M[1][1]+M[1][2]+M[2]
[0]+M[2][1]+M[2][2]) ); Bn[i][j]=(B[i-1][j-1]M[0][0]+B[i-1][j]M[0][1]+B[i-
1][j+1]M[0][2]+ B[i ][j-1]M[1][0]+B[i ][j]M[1][1]+B[i ][j+1]M[1][2]+
B[i+1][j-1]M[2][0]+B[i+1][j]M[2][1]+B[i+1][j+1]*M[2][2]) /( (M[0]
[0]+M[0][1]+M[0][2]+M[1][0]+M[1][1]+M[1][2]+M[2][0]+M[2][1]+M[2]
[2])==0 ? 1 : (M[0][0]+M[0][1]+M[0][2]+M[1][0]+M[1][1]+M[1][2]+M[2]
[0]+M[2][1]+M[2][2]) ); } } for(int i=0;i<lw;++i) for(int j=0;j<lk;++j) { R[i]
[j]=Rn[i][j]; G[i][j]=Gn[i][j]; B[i][j]=Bn[i][j]; } break;
case '+':
slices++;
stacks++;
break;
case '-':
if (slices>3 && stacks>3)
{
slices--;
stacks--;
}
break;
}
case 'q':
for(int i=1;i<lw-1;++i)
for(int j=1;j<lk-1;++j)
{
int Rmax=R[i-1][j-1];
for(int n=-1;n<2;n++)
for(int m=-1;m<2;m++)
{
if(Rmax<R[i+n][j+m])
Rmax=R[i+n][j+m];
}
Rn[i][j]=Rmax;
int Gmax=G[i-1][j-1];
for(int n=-1;n<2;n++)
for(int m=-1;m<2;m++)
{
if(Gmax<G[i+n][j+m])
Gmax=G[i+n][j+m];
}
Gn[i][j]=Gmax;
int Bmax=B[i-1][j-1];
```

```
for(int n=-1;n<2;n++)
for(int m=-1;m<2;m++)
{
if(Bmax<B[i+n][j+m])
Bmax=B[i+n][j+m];
}
Bn[i][j]=Bmax;
}
break;
glutPostRedisplay();
}
static void idle(void)
{
glutPostRedisplay();
}
const GLfloat light_ambient[] = { 0.0f, 0.0f, 0.0f, 1.0f };
const GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };
const GLfloat mat_ambient[] = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };
/* Program entry point */
int main(int argc, char *argv[])
{
ifstream plik("C:/Users/kamil546/Desktop/zd5.txt"); // sciezka do pliku
plik>>lw>>lk;
cout<<"wiersze="<<lw<<" kolumny="<<lk<<endl;
for(int i=0;i<lw;++i)
{
for(int j=0;j<lk;++j)
{
plik>>Rs[i][j];
plik>>Gs[i][j];
plik>>Bs[i][j];
}
}
plik.close();
for(int i=0;i<lw;++i){
for(int j=0;j<lk;++j){
Rn[i][j]=Rs[i][j];
Gn[i][j]=Gs[i][j];
Bn[i][j]=Bs[i][j];
```

```
}
}
glutInit(&argc, argv);
glutInitWindowSize(640,480);
glutInitWindowPosition(10,10);
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
glutCreateWindow("GLUT Shapes");
glutReshapeFunc(resize);
glutDisplayFunc(display);
glutKeyboardFunc(key);
glutIdleFunc(idle);
glClearColor(1,1,1,1);
glEnable(GL_CULL_FACE);
glCullFace(GL_BACK);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);
glEnable(GL_LIGHT0);
glEnable(GL_NORMALIZE);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
glutMainLoop();
return EXIT_SUCCESS;
}
```