Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет Систем Управления и Робототехники

Дисциплина: Обработка цифровых изображений

ОТЧЁТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №2

Геометрические преобразования изображений

Выполнил студент Топольницкий А.А.
Группа R4134c
Преподаватель Шаветов С.В.

Санкт-Петербург
2023 г.

Цель работы:

Освоить основные виды отображений и использование геометрических преобразований для решения задач пространственной коррекции изображений.

Краткое теоретическое обоснование:

Геометрическое преобразование изображений – изменение положений пикселей в другое положение при сохранении интенсивности.

Используется понятие однородных координат – координат, определяющих объект, причём объект не меняется при умножении всех координат на одно и то же число. Однако, для этого нужно использовать на одну координату больше, чем размерность пространства. Например, в двумерном координаты x, y, и некоторое число w. По сути, точка с декартовыми координатами *(x, y)* в однородных координатах запишется как *(x, y, 1)* и общий вид системы для расчёта координат следующий:

$$\begin{cases} x' = Ax + By + C \\ y' = Dx + Ey + F \end{cases} \quad (1)$$

Ход работы

Лабораторная работа выполнялась в MATLAB.

Листинг 1. *Построение сдвига изображения*

```
clc
clear all
I = imread('spb.jpg');

C = 350;
D = 250;
T = [1 0 0; 0 1 0; C D 1]; % задаём матрицу преобразований
tform = affine2d(T); % создаём матрицу преобразований
I_shift = imwarp(I, tform, 'OutputView',imref2d(size(I),...
   [1 size(I,2)], [1 size(I,1)])); % применяем матрицу к изображению

figure;
subplot(1,2,1)
imshow(I)
```

```
title('Оригинал')
subplot(1,2,2)
imshow(I_shift)
title('Сдвинутое изображение')
```

Комментарий: для сдвига коэффициенты $A = E = 1, B = D = 0$ в системе (1), тогда матрица преобразований T имеет вид, отражённый в листинге. С помощью функции *affine2d* создаём матрицу преобразований, *imwarp* применяет её к изображению. Параметр *OutputView* помогает при построении картинки отобразить смещение относительно оригинала, иначе бы MATLAB отцентрировал изображение.

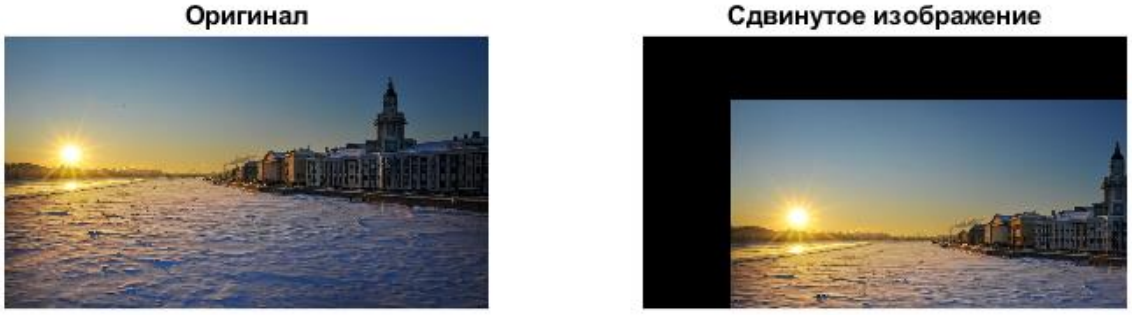

Рисунок 1. Применение сдвига к изображению

Листинг 2. *Пример отражения изображения*

```
T = [1 0 0; 0 -1 0; 0 0 1];
tform = affine2d(T);
I_reflect = imwarp(I,tform);

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_reflect)
title('Отражённое изображение')
```

Комментарий: для отражения коэффициенты $A = 1, E = -1, B = D = C = F = 0$ в системе (1).

Рисунок 2. Применение отражения к изображению

Листинг 3. *Пример однородного масштабирования*

```
alpha = 1.5;
betta = 1.5;

T = [alpha 0 0; 0 betta 0; 0 0 1];
tform = affine2d(T);
I_scale = imwarp(I, tform, 'OutputView',imref2d(size(I),...
   [1 size(I,2)], [1 size(I,1)]));

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_scale)
title(' Увеличенное изображение')
```

Комментарий: при масштабировании $A = \alpha,\ E = \beta, B = D = C = F = 0$. Если коэффициенты меньше 1, то изображение уменьшается, если больше, то увеличивается, а если не равны между собой, то пропорции будут тоже не одинаковыми по ширине и высоте. На рисунке 4 видно, что при разных коэффициентах происходит неодинаковое масштабирование.

Рисунок 3. Применение масштабирования с коэффициентами $\alpha = 1.5, \beta = 1.5$

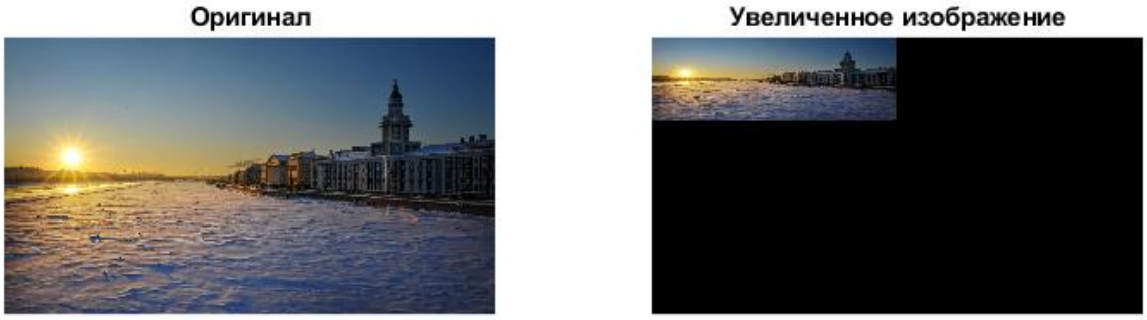

Рисунок 4. Применение масштабирования с коэффициентами $\alpha = 0.5, \beta = 0.3$

Листинг 4. *Поворот изображения*

```
phi = 23 * pi / 180;
T = [cos(phi) sin(phi) 0; -sin(phi) cos(phi) 0; 0 0 1];
tform = affine2d(T);
I_rot = imwarp(I, tform);

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_rot)
title({'Повёрнутое на ', num2str(rad2deg(phi)), 'градуса изображение'})
```

Комментарий: при повороте по часовой стрелке $A = cos\varphi,\ B = -sin\varphi, D = sin\varphi, E = cos\varphi, C = F = 0$.

Рисунок 5. Поворот изображения на 23 градуса

Листинг 5. *Пример скоса изображения*

```
s = -0.5;

T = [1 0 0; s 1 0; 0 0 1];
tform = affine2d(T);
I_skos = imwarp(I,tform);

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_skos)
title('Скошенное изображение')
```

Комментарий: Скос относится к аффинным отображениям, то есть таким отображениям, при которых параллельные прямые переходят в параллельные прямые, пересекающиеся в пересекающиеся, а скрещивающиеся в скрещивающиеся, при этом сохраняются отношения длин отрезков, принадлежащих одной прямой, и отношения площадей фигур. При скосе по часовой стрелке $A = E = 1, B = s, D = C = F = 0$.

Рисунок 6. Пример скоса изображения

Листинг 6. *Кусочно-линейное отображение*

```
middle = round(size(I,1) / 2);
I_left = I(:,1:middle, :);
strech = 2;
I_right = I(:,(middle + 1:end), :);
T = [strech 0 0; 0 1 0; 0 0 1];
tform = affine2d(T);
I_scale = imwarp(I_left, tform);
I_piecelinear = [I_scale I_right];

figure
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_piecelinear)
title('Кусочно-линейное преобразование')
```

Комментарий: кусочно – линейное отображение – это отображение, когда к частям изображения применяются разные линейные преобразования. Например, делим картинку на две части и для левой части используем масштабирование по одной из осей.

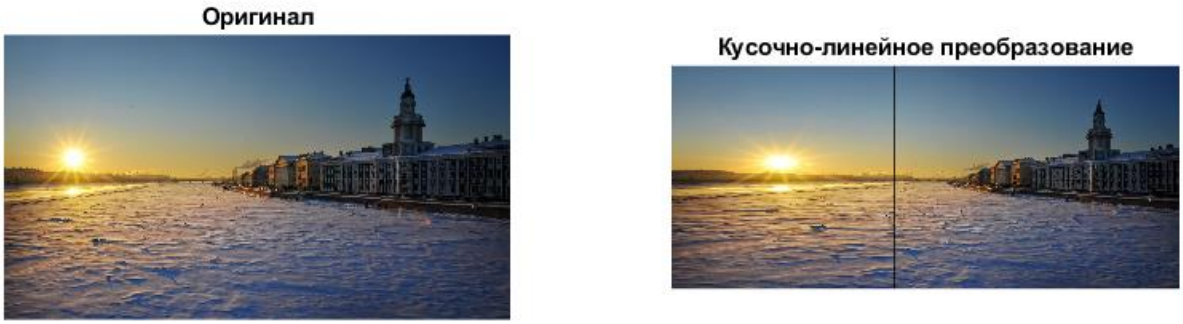

Рисунок 7. Кусочно-линейное отображение. Применение масштабирования для одной части

Листинг 7. *Проекционное отображение*

```
A = 1.5;
B = 0.3;
C = 0;
F = 0;
D = 0.2;
E = 1.3;
G = 0.005;
H = 0.001;
I_k = 4;
T = [A B C; D E F; G H I_k];
tform = projective2d(T);
I_skos = imwarp(I,tform);

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_skos)
title('Проекционное отображение')
```

Комментарий: проекционное отображение – прямые остаются прямыми, но в общем случае параллельность прямых не сохраняется, однако выполняется коллинеарность точек. Система уравнений для такого преобразования выглядит следующим образом:

$$\begin{cases} x' = \dfrac{Ax + By + C}{Gx + Hy + I} \\ y' = \dfrac{Dx + Ey + F}{Gx + Hy + I} \end{cases} \quad (2)$$

Рисунок 8. Применение проекционного отображения

Листинг 8. *Полиноминальное отображение*

```
[nRows, nCols, Layers] = size(I);
T = [0 0; 1 0; 0 1; 0 0.0001; 0.00002 0.00002; 0.0001 0];
for k=1:1:Layers
   for y = 1:1:nCols
      for x = 1:1:nRows
         xnew = round(T(1,1) + T(2,1)*x + T(3,1)*y + T(4,1)*x^2+ ...
            T(5,1)*x*y + T(6,1)*y^2);
         ynew = round(T(1,2) + T(2,2)*x + T(3,2)*y + T(4,2)*x^2+ ...
            T(5,2)*x*y + T(6,2)*y^2);
         I_pol(xnew,ynew,k) = I(x,y,k);
      end
   end
end

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_pol)
title('Полиномиальное отображение')
```

Комментарий: полиномиальное отображение использует полиномы. Например, матрица преобразования второго порядка будет иметь следующий вид:

$$\begin{cases} x' = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2 \\ y' = b_1 + b_2 x + b_3 y + b_4 x^2 + b_5 xy + b_6 y^2 \end{cases} \quad (2)$$

В MATLAB нет встроенной функции для данного преобразования, потому просто для каждой точки вычисляем координаты. На полученном изображении есть чёрные точки – поскольку отсутствует интерполяция и при пересчёте пустующие символы переопределились как 0.

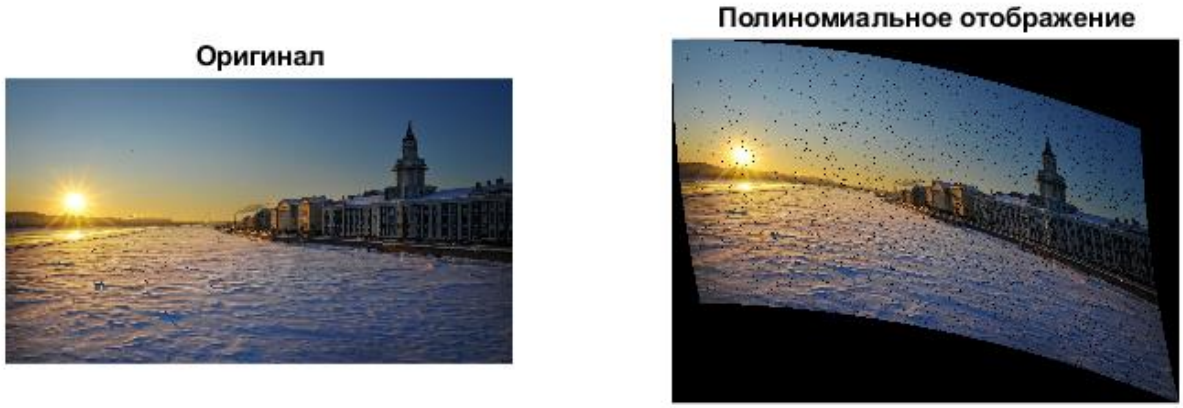

Рисунок 9. Применение полиномиального отображения

Листинг 9. *Синусоидальное искажение*

```
[xi, yi] = meshgrid(1:nCols, 1:nRows);
middle = round(size(I,2) / 2);
u = xi + 50 * sin(2 * pi * yi / 90);
v = yi;
tmap = cat(3,u,v);
resamp = makeresampler('linear','fill');
I_sin = tformarray(I,[],resamp, [2 1], [1 2], [], tmap, 0.3);

figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_sin)
```

```
title('Синусоидальное искажение')
```

Комментарий: с помощью функции *cat* объединяем массив ячеек в многомерный массив, с помощью *tformarray* применяем пространственную трансформацию. В итоге на изображении видно, что появились волны, подобные графику синуса.

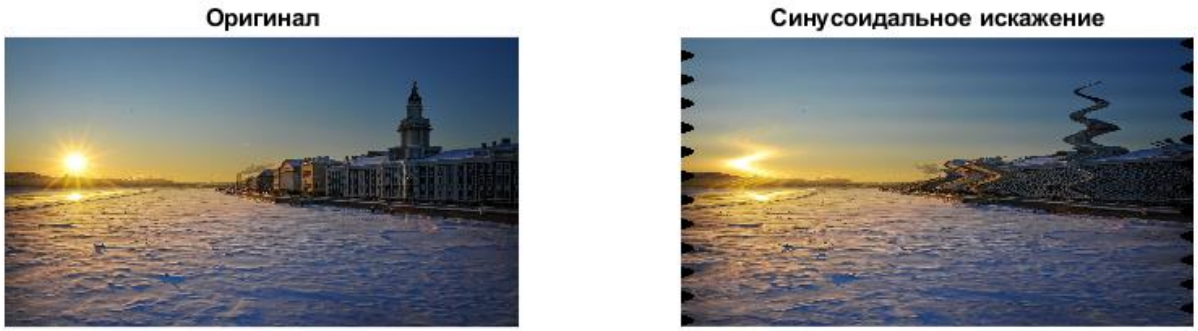

Рисунок 10. Пример синусоидального искажения

Листинг 10. *Исправление бочкообразной дисторсии*

```
for k = 1:1:Layers;
   Inew_eq(:,:,k) = histeq(I(:,:,k));

I = imread("city.jpg");

[nRows, nCols, Layers] = size(I);
[xi, yi] = meshgrid(1:nCols, 1:nRows);
middle = round(size(I,2)/ 2);
xt = xi(:) - middle;
yt = yi(:) - middle;

[theta, r] = cart2pol(xt,yt); % для удобства расчёта переводим в полярную СК
F3 = -0.0000000004;
F5 = -0.00000000008;
R = r + F3 * r.^2 + F5 * r.^4;
[ut, vt] = pol2cart(theta,R); % перевод обратно
u = reshape(ut, size(xi)) + middle;
v = reshape(vt, size(yi)) + middle;
tmap_B = cat(3, u, v);
resamp = makeresampler('linear', 'fill');
I_bar = tformarray(I, [], resamp,[2 1], [1 2], [], tmap_B, 0.3);
```

```
figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_bar)
title('Попытка исправления')
```

Комментарий: дисторсия – это оптическое искажение, выражающееся в искривлении прямых линий. Используется уравнение вида:

$$R = b_0 r + F_3 r^2 + F_5 r^4, \quad (3)$$

где R – точка с новыми координатами, b_0 – коэффициент линейного увеличения, F_i – коэффициенты дисторсии, вносящие наибольший вклад в искажение формы изображения. Для коррекции дисторсии используется подход, использованный и для проективного отображения.

На рисунке ниже приведён пример исправления, однако видно, что крайнее левое здание стало шире, при этом всё изображение стало выглядеть более естественным.

Рисунок 11. Попытка исправления бочкообразной дисторсии

Листинг 11. *Коррекция подушкообразной дисторсии*

```
I = imread("pin.jpg");
[nRows, nCols, Layers] = size(I);
[xi, yi] = meshgrid(1:nCols, 1:nRows);
middle = round(size(I,2)/ 2);
xt = xi(:) - middle;
```

```
yt = yi(:) - middle;
[theta, r] = cart2pol(xt,yt); % для удобства расчёта переводим в полярную СК
F3 = 0.000000004;
F5 = 0.0000000004;
R = r + F3 * r.^2 + F5 * r.^4;
[ut, vt] = pol2cart(theta,R); % перевод обратно
u = reshape(ut, size(xi)) + middle;
v = reshape(vt, size(yi)) + middle;
tmap_B = cat(3, u, v);
resamp = makeresampler('linear', 'fill');
I_pin = tformarray(I, [], resamp,[2 1], [1 2], [], tmap_B, 0.3);
figure;
subplot(1,2,1)
imshow(I)
title('Оригинал')
subplot(1,2,2)
imshow(I_pin)
title('Исправление')
```

Комментарий: Подход аналогичен исправлению бочкообразной дисторсии.

Рисунок 12. Попытка исправления подушкообразной дисторсии

Листинг 12. *Сшивка изображения*

```
top = imread('top.jpg');
bot = imread('bot.jpg');

topHT = im2double(rgb2gray(top));
botHT = im2double(rgb2gray(bot));


[nRows, nCols, Layers] = size(top);
[nRowsb, nColsb, Layersb] = size(bot);

intersecPart = 10;
botCorrHT = zeros(intersecPart, nCols);
topCorrHT = zeros(intersecPart, nCols);
corArray = [];



for j = 1:1: nCols
   for i = 1:1: intersecPart
      botCorrHT (i, j) = botHT (i, j);
   end
end

for j = 0:1: nRows - intersecPart
   for i = 1:1: intersecPart
      topCorrHT (i ,:) = topHT (i + j,:);
   end
   corCoef =corr2 (topCorrHT , botCorrHT);
   corArray = [corArray corCoef];
   corCoef = 0;
end
[M, I] = max(corArray);

nRowsBotCorr = nRowsb + I - 1;
for k=1:Layers
   for j =1:nCols
```

```
        for i= 1:I - 1
            res_img(i,j,k) = top(i,j,k);
        end
        for i = I:1:nRowsBotCorr
            res_img(i,j,k) = bot(i-I+1,j,k);
        end
    end
end
figure;
subplot(1,3,1)
imshow(res_img)
title('Склейка')
subplot(1,3,2)
imshow(top)
title('Top')
subplot(1,3,3)
imshow(bot)
title('Bottom')
```

Комментарий: в этом задании алгоритм склеивания изображения следующий. Для простоты определения коэффициента корреляции с помощью функции *rgb2gray* преобразуем цветные изображения в полутоновые. Дальше определяем размеры и определим верхнюю строку нижнего изображения. Сравним полученную строку со строками верхнего изображения. С помощью функции *corr2* будем вычислять коэффициент корреляции между строками. В итоге, выберем строку, у которой коэффициент корреляции наибольший. Поскольку изображение делилось не по какому-то конкретному пикселю, а с запасом, то в итоге имеем строку, которая есть на обеих частях, потому коэффициент и наибольший. Строим склеенное изображение по этой границе.

Рисунок 13. Склеивание изображения, поделённого горизонтальной прямой

Выводы. В ходе лабораторной работы были выполнены следующие шаги:

- Для изображения был выполнен сдвиг, зеркальное отражение изображения, масштабирование и поворот на произвольный градус;
- Для изображения были применены аффинные отображения, в частности скос и кусочно-линейное преобразование;
- Для изображения были применены проекционное и полиномиальное отображения, а также синусоидальное искажение;
- Была проведена коррекция дисторсии. Для бочкообразной дисторсии коррекция прошла не совсем успешно, так как с одного края видно, что здание стало намного шире, чем на оригинале. Для подушкообразной дисторсии коррекция была выполнена успешно;
- Было проведено склеивание изображения, разделённого по горизонтали.

Ответы на вопросы к лабораторной работе:

1. Каким образом можно выполнить поворот изображения, не используя матрицу поворота?

   Ответ:

Если делать вручную, то можно через настройки изображения, выбрать поворот и повернуть на необходимый градус. В MATLAB есть функция *imrotate,* позволяющая повернуть изображение на необходимый градус. Данная функция имеет параметр *method*, в который можно добавить метод интерполяции, например, *bilinear*, *bicubic*, которые используют 2 на 2 и 4 на 4 ближайших соседей соответственно для определения значения пикселя.

Если поворот осуществляется на 90 градусов, то для координат по x в новой СК можно взять отрицательную координату по y (-y) исходного, а для y в новой взять x.

2. Какое минимальное количество соответствующих пар точек необходимо задать на исходном и искаженном изображениях, если порядок преобразования n = 4?

   Ответ:

   Для вычисления необходимых пар точек используется формула

$$t_{min} = \frac{(n+1)(n+2)}{2}$$

   Тогда можно рассчитать: (4 + 1) * (4 + 2) / 2 = 30 / 2 = 15 пар точек.

3. После геометрического преобразования изображения могут появиться пиксели с неопределенными значениями интенсивности. С чем это связано и как решается данная проблема?

   Ответ: Данная проблема решается с помощью интерполяции, например, на основе ближайших 4 или 16 соседей. Причина состоит в том, что расчёт производится перебором точек и, поскольку мы используем полином для вычисления координат, а затем в новые координаты записываем значение пикселя по старым(строка *I_new(xnew,ynew,k)=I(x,y,k)*), то не для всех пикселей происходит расчёт, потому они имеют неопределённую интенсивность.