# Examen Parcial 2.

**Unidad 2. Geometría de la imagen**

**Integrantes:** Palomeque Rebollo Jonathan

Estrada Gonzalez Carlos Daniel

## Objetivo:

Comparar el resultado de una sucesión de transformaciones de imágenes, con respecto a una composición que afectúe en el mismo orden dichas transformaciones, cuidando de rellenar con valor a aquellos pixeles de la imagen de salida que no tuvieron cálculo.

## Pasos:

1. Dadas tres transformaciones, cuyas matrices de transformaciones (tamaño 3x3) denotaremos 𝐴1, 𝐴2 y 𝐴3, respectivamente. Calcular la matriz equivalente de la composición, como:

𝐴𝑇 = 𝐴3 𝐴2𝐴1

1. Obtener la fórmula para X′ y Y′, a saber:
2. Generar una función que aplique fórmula y estandarice coordenadas para evitar negativos y

decimales.

1. Modificar el resultado obtenido para recuperar datos de pixeles sin resultado en la transformación, mediante la búsqueda en sus pixeles vecinos.
2. Calcular el número de pixeles que permanecen sin cálculo, los recuperados y el porcentaje que representan cada uno de ellos del total de pixeles (no. de renglones x no. de columnas).
3. Crear un programa1 que realice la transformación compuesta con cinco conjuntos de parámetros y se visualice las imágenes resultantes.
4. Crear un programa2 que haga la llamada a las funciones individuales de imágenes, como se hicieron en clase, que ejecute la misma secuencia de transformaciones. Aplicar a las mismas imágenes los mismos conjuntos de parámetros y visualice las imágenes resultantes.
5. Incluir en el reporte una tabla comparativa por cada programa con los siguientes datos: Programa 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parámetros de la transformación** | **Imagen original (No. renglones**  **x No. de columnas)** | | **Imagen resultante (No. renglones**  **x No. de columnas)** | | **No. de pixeles sin cálculo** | |
| **#** | **%** |
| 1.- (I, 1, -100, 0.001) | 862 | 600 | 1589 | 1754 | 195,993  195993  195993195993 195,993  195,993 | 7.03 |
| 2.-(I, 0.5, 45, 0.002) | 862 | 600 | 1339 | 3,584 | 948,298 | 19.73 |
| 3.-(I, 0.7, 100, 0.005) | 862 | 600 | 1335 | 7063 | 3,143,579 | 33.33 |
| 4.-(I, 0.1, 10, 0.004) | 862 | 600 | 968 | 3195 | 851,919 | 27.54 |
| 5.-(I, 0.2, -50, 0.003) | 862 | 600 | 1145 | 3975 | 1,147,667 | 25.21 |

Programa 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parámetros de la transformación** | **Imagen original (No. renglones**  **x No. de columnas)** | | **Imagen resultante (No. renglones**  **x No. de columnas)** | | **No. de pixeles sin cálculo** | |
| **#** | **%** |
| 1.- (I, 1, -100, 0.001) | 862 | 600 | 1543 | 648 | 30,425  30,425  30,425  30,425  30,425  30,425 | 3.04 |
| 2.-(I, 0.5, 45, 0.002) | 862 | 600 | 1246 | 1261 | 198,986  198,9986 198,986 | 12.66 |
| 3.-(I, 0.7, 100, 0.005) | 862 | 600 | 1158 | 4713 | 1,028,587 | 18.84 |
| 4.-(I, 0.1, 10, 0.004) | 862 | 600 | 751 | 2522 | 351,085 | 18.53 |
| 5.-(I, 0.2, -50, 0.003) | 862 | 600 | 954 | 1745 | 116,822 | 7.01 |

## Conclusiones: El desarrollo de este proyecto nos permitió poner en practica las distintas transformaciones que hicimos en clase y lograr comprender mas a fondo como se tratan las imágenes con estas transformaciones, El juntar tres transformaciones en un programa que las realice en una sola función represento un reto el cual requirió un análisis y calculo de las operaciones necesarias para transformar la matriz de pixeles de la imagen y tratarla para corregir la perdida de pixeles, pudimos observar que al realizar la transformación por partes se genera “espacio de mas” en comparación si la realizamos con una sola función, ya que al realizar la rotación no guarda ese espacio y la imagen conserva el tamaño original.

## Código:

%% Programa 1 - Por partes

clc;

clear all;

close all;

I1=imread('C:\Users\Carlos Daniel\Desktop\TopicoI\_2\jordan.jpg');

tic; [Inclinar1]=inclinar(I1,0.2); toc;

tic; [IRot1,IBand1]=rotar(Inclinar1,-50); toc;

tic;[perspectiva1, IBand1]=perspectiva(IRot1,0.003);toc;

tic; [IR1, IBand1R, noPixeles1] = rellenarImg(perspectiva1, IBand1, 1); toc;

imwrite(IRot1, 'C:\Users\Carlos Daniel\Desktop\TopicoI\_2\rotar5\_enPartes.jpg');

imwrite(perspectiva1, 'C:\Users\Carlos Daniel\Desktop\TopicoI\_2\perspectiva5\_enPartes.jpg');

imwrite(IR1, 'C:\Users\Carlos Daniel\Desktop\TopicoI\_2\rellenada5\_enPartes.jpg');

imwrite(Inclinar1, 'C:\Users\Carlos Daniel\Desktop\TopicoI\_2\Inclinar5\_enPartes.jpg');

fprintf("======================================================\n");

total1=size(IR1,1)\*size(IR1,2);

fprintf("Imagen 1(%d,%d)\n",size(IR1,1),size(IR1,2));

fprintf("Pixeles recuperados %d (%7.5f porcentaje)\n",noPixeles1, (noPixeles1/total1)\*100);

figure;

image(I1); title('Imagen original');

figure;

image(IR1); title('en partes');

%%-----------------------------------------------------------------------%%

%% Programa 2 - Compuesto

clc;

close all;

clear all;

I=imread('C:\Users\Carlos Daniel\Desktop\TopicoI\_2\jordan.jpg');

tic; [Icompuesta, Iband1] = transfCompuesta(I,.1,10,0.004); toc;

tic;[IcompuestaR, Iband1R,noPixeles1] = ...

rellenarImg(Icompuesta, Iband1, 1); toc;

fprintf("======================================================\n");

total1=size(Icompuesta,1)\*size(Icompuesta,2);

fprintf("Imagen 1(%d,%d)\n",size(Icompuesta,1),size(Icompuesta,2));

fprintf("Pixeles recuperados %d (%7.5f porcentaje)\n",noPixeles1, (noPixeles1/total1)\*100);

fprintf("======================================================\n");

imwrite(IcompuestaR, 'C:\Users\Carlos Daniel\Desktop\TopicoI\_2\rellenada\_compuesta\_4.jpg');

figure;

image(IcompuestaR); title('compuesta rellenada');

%%-----------------------------------------------------------------------%%

%% Funcion - Tranformacion Compuesta

function [ G, IBand ] = transfCompuesta( I, AnIn, AnRo , Kpe)

G = 0;

IBand = 0;

cosIn = cosd(AnRo);

senIn = sind(AnRo);

nCoord=zeros(size(I,1),size(I,2),2); %% X' , Y'

for i=1:size(I,1) %% renglones

for j=1:size(I,2) %% columnas

tempx = (cosIn\*AnIn)- senIn;

tempy = ((senIn\*AnIn)+cosIn);

nx = (i \* cosIn) + (j\* tempx);

ny=(i\*senIn)+(j\*tempy);

npx=nx\*ny\*Kpe;

npy=ny;

nCoord(i,j,1)=npx;

nCoord(i,j,2)=npy;

end

end

%% validar decimales y negativos en las coordenadas

nCoord=round(nCoord);

minX =min( min(nCoord(:,:,1)));

minY =min( min(nCoord(:,:,2)));

maxX=max( max(nCoord(:,:,1)));

maxY=max( max(nCoord(:,:,2)));

if minX <= 0

nCoord(:,:,1)=nCoord(:,:,1)+abs(minX)+1;

end

if minY <= 0

nCoord(:,:,2) = nCoord(:,:,2)+abs(minY)+1;

end

%% asignar nuevas coordenas

G = zeros(maxY,maxX,size(I,3));

IBand=zeros(maxY,maxX);

for i=1:size(I,1) %% renglones

for j=1:size(I,2) %% columnas

nx=nCoord(i,j,1);

ny=nCoord(i,j,2);

G(ny,nx,:)=I(i,j,:);

IBand(ny,nx)=1;

end

end

G=uint8(G);

IBand=uint8(IBand);

end

%%-----------------------------------------------------------------------%%

%% inclinar una imagen con base a k

%% Entradas

%% I imagen de entrada

%% factor de inclinacion. valores de 0 ..1 (puede ser negativo)

%%salidas

%% G imagen de salida

%% IBand bandera con indicador de valor (1=existe valor)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ G ] = inclinar( I, k )

deltaY = round((1:size(I, 1))\*k);

minY = min(deltaY);

if (minY<0)

deltaY=deltaY+abs(minY);

end

G = zeros(size(I, 1), size(I, 2)+max(deltaY), size(I, 3));

nc = size(I, 2);

for i = 1:size(I, 1)

G(i, deltaY(i)+1:deltaY(i)+nc, :) = I(i, :, :);

end

G=uint8(G);

end

%%-----------------------------------------------------------------------%%

%% Rotar Una Imagen

% Entradas:

% I - Imagen

% angulo - angulo de rotaci?n

% Salidas:

% G - Imagen de salida

% IBand - Bandera con indicador de valor (1-existe valor);

function [ G, IBand ] = rotar( I, angulo )

G = 0;

IBand = 0;

%%calcular y guardar coordenadas

coseno=cosd(angulo);

seno=sind(angulo);

nCoord=zeros(size(I,1),size(I,2),2);

for i=1:size(I,1) % renglones y

for j=1:size(I,2) % columnas x

nx=(j\*coseno)-(i\*seno);

ny=(j\*seno)+(i\*coseno);

nCoord(i,j,1) = nx;

nCoord(i,j,2) = ny;

end

end

%%validar decimales y negativos en las coordenadas

nCoord=round(nCoord);

minX=min(min(nCoord(:,:,1)));

minY=min(min(nCoord(:,:,2)));

if (minX < 0)

nCoord(:,:,1)=nCoord(:,:,1)+abs(minX)+1;

end

if (minY < 0)

nCoord(:,:,2)=nCoord(:,:,2)+abs(minY)+1;

end

maxX=max(max(nCoord(:,:,1)));

maxY=max(max(nCoord(:,:,2)));

%% asignar nuevas coordenas

G = zeros(maxY,maxX,size(I,3));

IBand=zeros(maxY,maxX);

for i=1:size(I,1) %% renglones

for j=1:size(I,2) %% columnas

nx=nCoord(i,j,1);

ny=nCoord(i,j,2);

G(ny,nx,:)=I(i,j,:);

IBand(ny,nx)=1;

end

end

G=uint8(G);

IBand=uint8(IBand);

End

%%-----------------------------------------------------------------------%%

%% Perspectiva

%% Entradas

%% I - imagen

%% K - Constante de perspectiva (cercano a cero)

%% IBand con bandera con indicador de valor (1-existe valor)

function [G,Bandera] = perspectiva(I,k)

nCoord=zeros((size(I,1)),(size(I,2)));

for i =1:size(I,1)

for j =1:size(I,2)

nX=j\*(i\*k);

nCoord(i,j)=nX;

end

end

%%validar coordenadas

nCoord=round(nCoord);

minXs = min((nCoord(:)));

if (minXs<1)

nCoord(:,:)=nCoord(:,:)+abs(minXs)+1;

end

%%Crear imagen resultante y guardar

maxXs = max((nCoord(:)));

maxYs = size(I,1);

if(maxXs>10000)

G=0;

Bandera=0;

else

G = zeros(maxYs, maxXs, size(I, 3));

Bandera = zeros(maxYs, maxXs);

for i =1:size(I,1)

for j =1:size(I,2)

G(i,nCoord(i,j),:)=I(i,j,:);

Bandera(i,nCoord(i,j))=1;

end

end

end

G = uint8(G);

Bandera = uint8(Bandera);

end

%%-----------------------------------------------------------------------%%

%% Rellenar la imagen

%% Entradas

%% I - imagen

%% Band - bandera

%% lambda - no. de pixeles alrededor del actual

%% se van a buscar

%% Salidas

%% G - imagen rellenada

%% BandN - bandera nueva

%% noPixeles - recuperados

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [ G, BandN, noPixeles ] = rellenarImg( I, Band, lambda )

G=I; %% imagen a trabajar

BandN=Band; %% bandera a modificar

noPixeles=0;

for i=1:size(I,1) %% renglones

for j=1:size(I,2)%% columnas

if (Band(i,j)==0) %% no tiene calculo

nVecinos=0; %% no. de vecinos con valor

valores=zeros(1,size(I,3)); %% valores de esos vecinos

for r=(i-lambda):(i+lambda)

for c=(j-lambda):(j+lambda)

if ((r>0)&&(r<=size(I,1)))

if ((c>0)&&(c<=size(I,2)))

if (Band(r,c)>0)

nVecinos=nVecinos+1;

valores(nVecinos,:)=I(r,c,:);

end

end

end

end

end

if (nVecinos>0) %% vecinos con valor

if (nVecinos==1) %% unico vecinos

G(i,j,:)=valores;

else %% dos o mas vecinos

G(i,j,:)=round(mean(valores));

end

BandN(i,j)=1;

noPixeles=noPixeles+1;

end

end

end

end

end

%%-----------------------------------------------------------------------%%

Imágenes y programa: https://drive.google.com/drive/folders/1y9mm7BPvcCR3a71E1WujNaD6zjhrvmHx?usp=sharing