Universidad Veracruzana

Maestría en Inteligencia Artificial

Visión por Computadora

Tarea 7. Aplicación de transformaciones afines en imágenes con escala de grises en MATLAB.

Ángel García Báez

Profesor: Dr. Héctor Acosta Mesa

April 22, 2025

Contents

1 Objetivo de la práctica

Se tiene la siguiente imagen, sobre la cual se quieren realizar transformaciones afines :

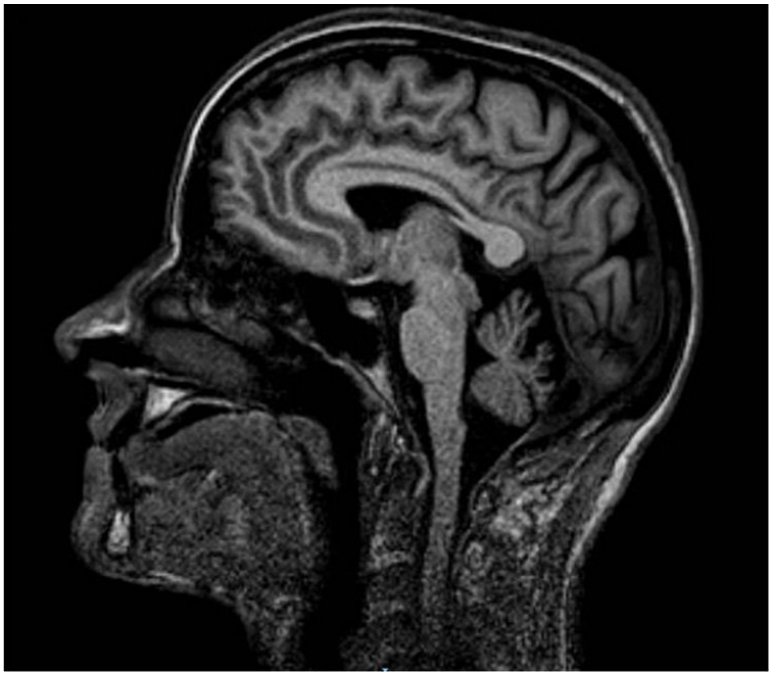


Figure 1: Imagen base.

Se desea implementar y aplicar transformaciones afines, digase la posibilidad de aplicar rotación, traslación y escalado a la imagen original, con la finalidad de manipular la imagen a conveniencia para obtener una mejor visualización de la misma. Para esto se quiere diseñar una función a la cual se le pase como parametros de entrada la imagen, la traslación en X y en Y, el factor de escala para X y para Y así como su angulo de rotación θ

2 Metodología

Una forma de definir las transformaciones afines sobre imágenes acorde con Coste (2012) es una función que mapea un objeto desde un espacio afín a otro con la particularidad de que mantiene su estructura. Esta idea aplicada al procesamiento de imágenes, se traduce en un conjunto de transformaciones que se aplican sobre los pixeles de la imagen original de tal forma que logran rotarla, escalarla o trasladarla respecto a su espacio original pero sin perder la estructura de la imagen.

Para llevar a cabo esto sobre cada uno de los pixeles de una imagen, es necesario escribir a cada pixel en base a su posición en fila y en columna, de forma entonces que lo que nos interesa del pixel es su posición como un punto en un plano de dos dimensiones.

Dado que se están trabajando con imágenes en 2 dimensiones, es necesario dotar a cada uno de las coordenadas de los pixeles de una dimensión extra para mantener la homogeneidad de tras aplicar las transformaciones.

Las transformaciones que se van a manejar acorde con Gonzalez and Woods (2018) y para los fines de esta tarea son: traslación, escalamiento y rotación.

La matriz de traslación representa que tanto se va a recorrer la imagen en X y en Y respecto a la imagen original y toma la siguiente forma:

$$T = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

La matriz de rotación representa la rotación respecto a la esquina superior izquierda de la imagen de magnitud θ y toma la siguiente forma:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matriz de escalado representa la magnitud respecto a la cual se va a escalar la imagen tanto en X como en Y respecto a la imagen original y toma la siguiente forma toma la siguiente forma:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Definidas las matrices que se van a aplicar, se define el orden de las transformaciones, el orden en que se aplican las transformaciones impacta en el resultado final de la imagen, en este caso se define a toda la matriz de transformaciones afines como sigue:

$$M = T \cdot R \cdot S$$

Primero se aplica el escalado, luego la rotación y finalmente la traslación.

Definiendo a P como el vector que contiene las coordenadas X, Y del pixel más una tercerda dimensión con la unidad, la transformación del pixel con el espacio afín se ve como sigue:

$$\mathbf{p}' = M \cdot \mathbf{p} = T \cdot R \cdot S \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Ahora que se tienen los valores remapeados en el espacio de transformación, es necesario encontrar su correspondiente en el espacio de la imagen original para que se vea reflejada dicha transformación. Para ello se hace el calculo de la inversa de la matriz de transformaciones M como sigue:

$$\mathbf{p} = M^{-1} \cdot \mathbf{p}'$$

Aplicandolo

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

De esta forma, se recuperan los valores pero en escala continua, esto debe ser corregido debido a que el dominio original de las coordenadas de la imagen es discreto, aquí se aplica la interpolación bilineal que ya se hizo y se explico en el desarrollo de la tarea 2.

Con toda la maquinaria teórica para poner en funcionamiento las transformaciones afines, se programo la implementación en matlab y se hicieron 4 aplicaciones de la implementación mostrando cada una de las transformaciones de manera individual y una donde se combinan las 3.

3 Resultados de aplicar traslación

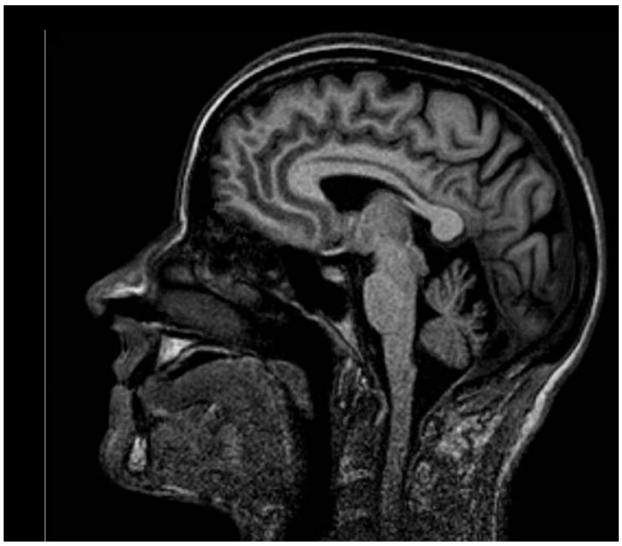


Figure 2: Traslación de la imagen.

Tras aplicarle una breve traslación de 50 pixeles en X y 30 unidades en Y, se muestra practicamente igual a la figura original, pero con la diferencia de que marca una raya blanca a la izquierda que indica que a partir de donde comienza la imagen trasladada, si se mira con más atención, la parte del cráneo de encuentra más pegada hacia el borde derecho de la imagen.

4 Resultados de aplicar rotación

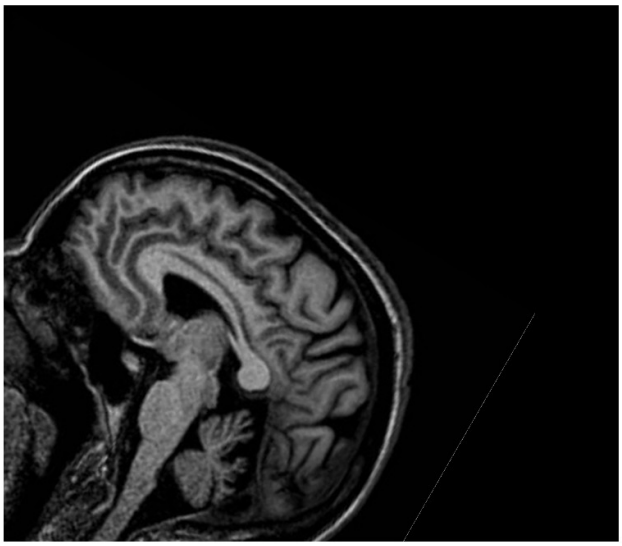


Figure 3: Rotación de la imagen.

Tras aplicar una rotación de 30 grados respecto al inicio de la imagen, que es el pixel superior izquierdo, se muestra como gran parte de la imagen original sale del cuadro, esto debido a que la transformación hace el mapeo con la rotación y muestra unicamente lo que debería quedar dentro de la imagen.

5 Resultados de aplicar escalado no uniforme

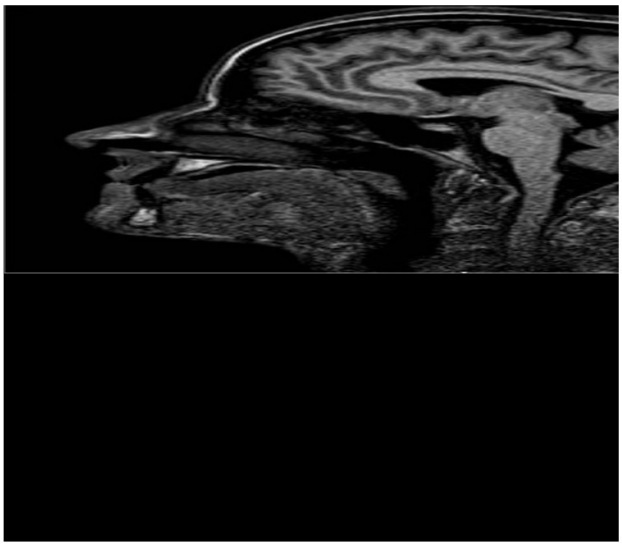


Figure 4: Escalado no uniforme de la imagen.

Tras aplicar un escalado no uniforme a la imagen de 1.5 en X y de 0.5 en Y, se observa de forma muy marcada como a lo ancho de la imagen, cabe en la mitad del espacio de la imagen original sin mayor problema, pero a lo largo se muestra como la imagen se "desborda" de la pantalla.

6 Resultados de aplicar todo

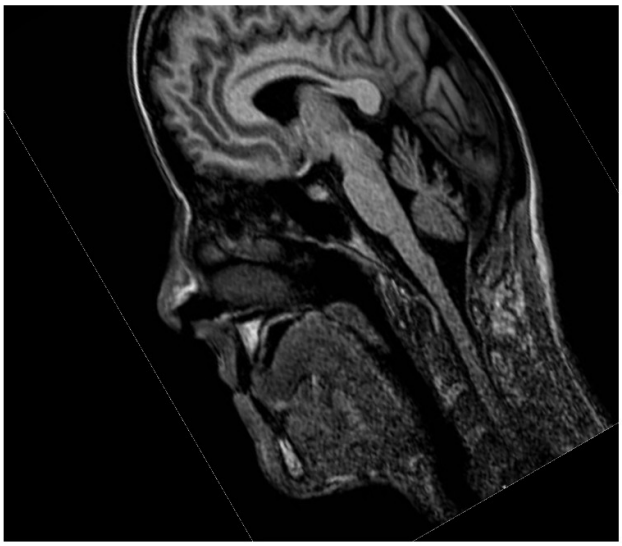
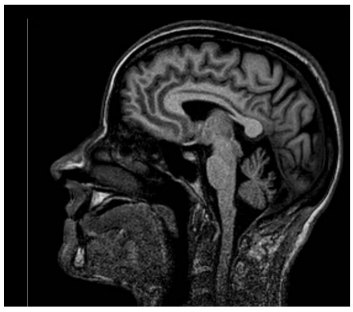


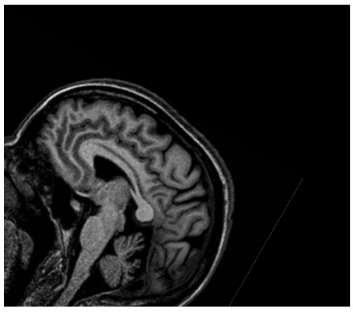
Figure 5: Traslación, rotación y escalado de la imagen.

Se muestra el resultado de aplicar una traslación de -40 unidades en X, 60 unidades en Y, un escalado de 0.8 y 1.2 respectivamente junto con una rotación de -30 grados. El resultado de todo esto es la imagen rotada tal que se ve de forma diagonal y se puede decir que "estirada" por el escalamiento.

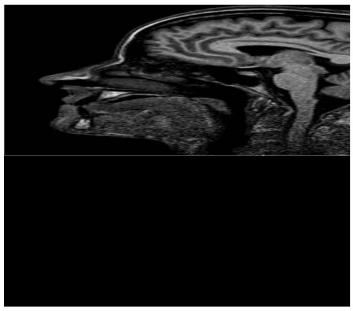
7 Resultados de todas las pruebas



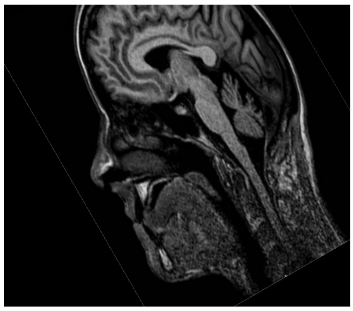
(a) Traslación de la imagen



(b) Rotación de la imagen



(c) Escalado no uniforme de la imagen



(d) Traslación, rotación y escalado de la imagen.

Figure 6: Resumen de los resultados

8 Conclusiones

Tras la implementación y los experimentos realizados con las transformaciones afines, se observa que cumple bastante bien su trabajo de aplicar las transformaciones sobre la imagen original para proyectarla y luego representarla sobre su espacio original. Lo destacable aquí es que la imagen utilizada al no tener alguna rotación o re escalado por si misma no permite poner en practica que tan buena seria la transformación en el caso de tratar de restaurar la imagen a su estado original. Es decir, partir de una imagen como la mostrada en el resultado de aplicar todas las transformaciones o en donde se aplica el escalado uniforme y tratar de dejarla como se ve la imagen base.

9 Referencias

References

Coste, A. (2012). Cs6640: Image processing – project 3: Affine transformation, landmarks registration, non-linear warping. `https://www.sci.utah.edu/~acoste/uou/Image/project3/ArthurCOSTE_Project3.pdf`. Course project for CS6640, University of Utah.

Gonzalez, R. C. and Woods, R. E. (2018). *Digital Image Processing*. Pearson.

10 Anexos

10.1 Implementación de las transformaciones afines con la interpolación bilineal

```
1
2  %% Interpolación bilineal %%
3  function valor = bilineal(imagen,x,y)
4  [filas,columnas] = size(imagen);
5
6  % Coordenadas vecinas más próximas
7  x1 = floor(x); x2 = ceil(x);
8  y1 = floor(y); y2 = ceil(y);
9
10 % Limitar los valores al rango válido
11 if x1 < 1 || x2 > columnas || y1 < 1 || y2 > filas
12 valor = 0;
13 return;
14 end
15
16 % Valores de píxeles vecinos
17 Q11 = double(imagen(y1,x1));
18 Q21 = double(imagen(y1,x2));
19 Q12 = double(imagen(y2,x1));
20 Q22 = double(imagen(y2,x2));
21
22 % Pesos
23 dx = x - x1;
24 dy = y - y1;
25
26 % Interpolación bilineal
27 valor = (1-dx)*(1-dy)*Q11 + ...
28 dx*(1-dy)*Q21 + ...
29 (1-dx)*dy*Q12 + ...
30 dx*dy*Q22;
31 end
32
33 %% Transformación afin %%
34 function salida = trans_afin(imagen,Tx,Ty,sx,sy,theta)
35 % Definir una matriz de escalado %%
36 S = [sx, 0, 0;
```

```
0, sy, 0;
0, 0, 1];
% Definir una matriz de rotación en grados (no radianes) %%
% Convertir los grados a radianes
theta1 = theta * pi/180;
% Definir la matriz de rotación %
R = [cos(theta1), -sin(theta1),0;
sin(theta1), cos(theta1), 0;
0,0,1];
% Definir la matriz de Traslación %%
T = [1,0,Tx;
0, 1, Ty;
0, 0, 1];
% Orden de aplicación %%
% Se define el orden de aplicación tal cual se desarrollaron: Escalar,
% Rotar y Trasladar en ese orden.
M = T * R * S;
% Base de la salida %%
D = size(imagen);
salida = zeros(D,"uint8");
% Inversa de la matriz de transformación%
Minv = inv(M);
% Aplicar aqui la interpolación %
for i = 1:D(1)
for j = 1:D(2)
% Coordenadas de destino (x', y')
destino = [j;i;1];

% Coodenadas de origen (x,y) con la inversa %
origen = Minv * destino;
x = origen(1);
y = origen(2);
% Verificar validez del rango
if x>=1 && x<=D(2) && y>=1 && y<=D(1)
%  Aplicar la interpolación
valor = bilineal(imagen,x,y);
salida(i,j) = uint8(valor);
end
end
end
```

```
77  end
78
79
80  %% Prueba de la transformación %%
81  I = imread("IMG\\F1.jpg");
82  imagen = rgb2gray(I);
83  %% Mostrar la imagen %
84  imshow(imagen);
85
86  %% Solo traslación %
87  R1 = trans_afin(imagen,50,30,1,1,0);
88  imshow(R1)
89  %% Solo rotación %
90  R2 = trans_afin(imagen,0,0,1,1,30);
91  imshow(R2)
92  %% Escalado NO uniforme %
93  R3 = trans_afin(imagen,0,0,1.5,0.5,0);
94  imshow(R3)
95  %% Combinación completa %
96  R4 = trans_afin(imagen,-40,60,0.8,1.2,-30);
97  imshow(R4)
98
```