PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UN ALGORITMO DE ESTABILIZACIÓN DE VIDEO ORIENTADO A LA DETECCIÓN DE PERSONAS

Tesis para optar el Título de **INGENIERO ELECTRÓNICO,** que presenta el bachiller:

Alberto Hiroshi Inafuku Yoshida

ASESOR: Renán Alfredo Rojas Gómez

Lima, mayo de 2015

ÍNDICE DE CONTENIDO

ANEXO

A. Pseudo Códigos

A1. Algoritmo Principal

Algoritmo 1. Algoritmo de Estabilización de Video

Entradas: Archivos PNT, Frames de entradas (f)

Salidas: Frames de salida (f')

Lectura de Frames y archivos PNT

Detección de Trayectorias Útiles — Algoritmo 2

Normalización

Para todos los puntos de cada Frame:

Fila Norm. = Fila / Máximo de Fila

Columna Norm. = Columna / Máximo de Columna

Primera Función de Transferencia Epipolar — Algoritmo 3

Filtrado de Trayectorias — Algoritmo 4

Segunda Función de Transferencia Epipolar — Algoritmo 5

Filtrado de Trayectorias (Opcional) — Algoritmo 4

Desnormalización

Para todos los puntos de cada Frame:

Fila Desnorm. = Fila * Máximo de Fila

Columna Desnorm. = Columna * Máximo de Columna

Para Todos los frames f

Frame Warping — Algoritmo 6

Remuestreo — Algoritmo 7

Para Todos los puntos, para todos los frames
    Si punto (p) cumple con la condición
        Máscara (p) = 1
        Si trayectoria se mantiene por 50 frames o más
            Máscara W=1
        Si no
            Máscara W=0.02 * Número de frames detectados
                hasta el momento
    Si no
        Máscara (p) =0;
        Para 50 frames
            Si el valor de W es menor al W anterior
                Máscara W = 0.02 * número de frame

Para Todas las trayectorias
    Hallar Inicio y Final
    Descartar trayectorias es menor a 20 frames
    Si trayectoria < 10
        W =0.5

A2. Primera Función de Transferencia Epipolar

El algoritmo tiene tres partes principales: el cálculo de matrices y líneas epipolares, el filtrado de líneas y la intersección de líneas epipolares.

Entradas

El algoritmo recibe como entradas las coordenadas de los puntos detectados en los archivos .pnt, la máscara encontrada en el algoritmo 2 y el número de frames del video.

Salida

El algoritmo devuelve como salida los puntos virtuales hallados por la función de transferencia epipolar.

Algoritmo 2. 1era. Función de Transferencia Epipolar

Entradas: Puntos de Entrada, Máscara, Nro. De Frames
Salidas: Puntos de Salida

```
Inicializar variables
Para frame 1 al 10
        Puntos de Salida = Puntos de Entrada
Para frame 11 al último
        Utilizar máscara como índice de puntos adecuados
        Para frame anterior t1 al t10
                Matriz fundamental ti, t
                Línea epipolar ti, t
        Evaluación de norma y ángulo
        Mínimos Cuadrados para hallar intersección
```

A3. Algoritmo para el filtrado

Algoritmo 3. Filtrado

Entradas: Trayectorias, Puntos Virtuales, Longitud del Filtro, Varianza del Filtro

Salidas: Puntos filtrados

Inicializar variables

Inicializar filtro Gaussiano H con datos de Longitud y Varianza

Para Todos los puntos

    Para Todos los frames en las trayectorias

        Tray. Filtrada filas = tray fila * H

        Tray. Filtrada columnas = tray fila * H

A4. Segunda Función de Transferencia Epipolar

Algoritmo 4. 2da. Función de Transferencia Epipolar

Entradas: Puntos de Entrada, Puntos Virtuales, Máscara, Nro. De Frames

Salidas: Puntos de Salida

Inicializar variables

Para primeros y últimos cinco frames

    Puntos de Salida = Puntos de Entrada

Para frame 6 al último-6

    Utilizar máscara como índice de puntos adecuados

    Para frame anterior t-5 al t+5

        Matriz fundamental ti, t

        Línea epipolar ti, t

    Evaluación de norma y ángulo

    Mínimos Cuadrados para hallar intersección

A5. Frame Warping

Algoritmo 5. Frame Warping

Entradas: Puntos Originales, Puntos Virtuales, Matriz de pesos, Número Frame, Tamaño de filas, R

Salidas: Vector de vértices

Utilizar matriz de pesos como índice de puntos útiles

Reducir imagen al tamaño $R\,x\,R$

Definir Número y tamaño de celdas

Matriz de coeficientes de interpolación bilineal: w

Inicializar Vectores de vértices y de puntos

Inicializar matriz dispersa de coeficientes M

Para Todos los puntos

    Definir Vector de vértices

    Dar valores de w a matriz M

Inicializar matriz dispersa N

Para Todas las celdas

    Dar pesos a valores respectivos de N

Minimización de energías:

$\min E = Ed + \alpha\, Es$

$V = (M^T M + N^T N)^{-1} M^T P$

A6. Remuestreo

Algoritmo 6. Remuestreo

Entradas: Vector de Vértices: V, Tamaño de celda, Número de celdas, Frame original: F_o, Tamaño de filas: R

Salidas: Frame estabilizado: F_e

Inicializar matrices A, Z, F_e y W

Para todas las celdas c

Hallar A_c= Z V_c

Para todos los puntos p_o

Hallar $W_p = [uv\ u\ v\ 1]$

$p_e = W_p\ A_c$

Para todos los puntos $1 \leq p_e \leq R$

$Fe(p_{e\,row}, p_{e\,col}) = F_o(p_{row}, p_{col})$

$Fe(p_{e\,row} + 1,\ p_{e\,col}) = F_o(p_{row,}\ p_{col})$

$Fe(p_{e\,row},\ p_{e\,col} + 1) = F_o(p_{row,}\ p_{col})$

$Fe(p_{e\,row} + 1,\ p_{e\,col} + 1) = F_o(p_{row,}\ p_{col})$

A7. Evaluación de Resultados

Algoritmo 7. Evaluación de Resultados

Entradas: Archivos PNT de video sin ruido, Archivos PNT de video con ruido,

Archivos PNT de video resultado

Salidas: Gráfico

Error de distancia

Inicialización de variables

Lectura de archivos .pnt

Para Todos los puntos, para todos los frames

Si punto (p) cumple con las condiciones

Formar matriz de puntos de correspondencia

Para puntos de la matriz

Hallar distancia de error entre original y distorsionada

Hallar distancia de error entre original y estabilizada

Almacenar distancias

Hallar promedio de distancias por frame

Graficar distancia error por frame

Hallar promedio de distancias para el video

B. Código en Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%    Algoritmo de Estabilización de Video     %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
%FACULTAD DE CIENCIAS E INGENIERÍA
%TRABAJO DE TESIS

% Alumno: Alberto Hiroshi Inafuku Yoshida
% Asesor: Renán Alfredo Rojas Gómez

%-------------------------------------------------------
% Procedimiento a seguir

%% 01. Limpieza de Workspace
%% 02. Lectura del video
%% 03. Creación de Máscaras
%% 04. Matriz de Puntos
%% 05. Normalización
%% 06. 1era Función de Transferencia Epipolar
%% 07. 1er Filtrado de Trayectorias
%% 08. 2da Función de Transferencia Epipolar
%% 09. Desnormalización
%% 10. 2da. Etapa de Filtrado
%% 11. Frame Warping y Remuestreo

%-------------------------------------------------------
%% 01. Limpieza de Workspace

clear all; close all;

%-------------------------------------------------------
%% 02. Lectura del video

% Primero se lee solo un frame para obtener el tamaño

I=imread('B:\Hiro\PUCP\TESIS\tesis\tesis2\Referencias\Banco de
Pruebas\video liu\liu_jpg\liu0000.jpg');

% Dado que la imagen tiene 3 capas de color (RGB) solo nos quedamos con
una
I=I(:,:,3);
[SizeR, SizeC]=size(I);   %Lectura del tamaño del frame

% Se indica el inicio, final del video y número total de frames

frame_start= 2; %Frame de inicio en video
frame_end= 250; %Frame final del video
frame_size= frame_end-frame_start+1; %Número total de frames

% Creación de las celdas de almacenamiento temporales para los puntos
tracker_info= cell( frame_size, 1);
```

```
type3d_cell= cell( frame_size, 1); %Evalúa el tipo de dato (0: No es
3D, 1: Esférico, 2: Cartesiano)
ident_cell= cell( frame_size, 1); %ID del punto
x_cell= cell( frame_size, 1); %Coordenada actual de columna
y_cell= cell( frame_size, 1); %Coordenada actual de fila

% Lectura de puntos encontrados

for i= frame_start: frame_end

    str=['000' num2str(i-1)];
    s=size(str);
    str=str((s(2)-3):s(2));
    tracker_info{i}=dlmread( ['B:\Hiro\PUCP\TESIS\liu\pnt\liu' str
'.pnt'], '\t', 1, 0);

    type3d_cell{ i, 1}= tracker_info{ i, 1}( :, 4);
    ident_cell{ i, 1}= tracker_info{ i, 1}( logical( type3d_cell{ i}),
8);
    x_cell{ i, 1}= tracker_info{ i, 1}( logical( type3d_cell{ i}), 1);
    y_cell{ i, 1}= tracker_info{ i, 1}( logical( type3d_cell{ i}), 2);

end
% Cantidad total de trayectoria
ident_max= max( cell2mat( ident_cell));
ident_index= 0: ident_max;
ident_index_size= length( ident_index);

% Creación de matrices temporales de trayectorias x e y.
traj_x_mat= zeros( ident_index_size, frame_size);
traj_y_mat= zeros( ident_index_size, frame_size);
traj_mask_temp=zeros( ident_index_size, frame_size);

for i= frame_start: frame_end
    %Lectura del número de puntos encontrados por frame
    temp_size= length( ident_cell{ i, 1});

    %Para todos los puntos encontrados
    for j= 1: temp_size
    %Almacenamiento de las posiciones x e y, creación de máscara
temporal
        traj_x_mat( ident_cell{ i, 1}( j)+ 1, i)= x_cell{ i, 1}( j); %
+1 since smallest index= 0
        traj_y_mat( ident_cell{ i, 1}( j)+ 1, i)= y_cell{ i, 1}( j);
        traj_mask_temp( ident_cell{ i, 1}( j)+ 1, i)= 1;
    end
%    disp( i); Opcional: Visualizar el avance
end

% figure;imagesc(traj_mask_temp); Opcional: Visualizar las trayectorias
encontradas


%-----------------------------------------------------
```

```
%% 03. Creación de Máscaras

%Inicialización de Matriz de Trayectorias: Tray
%Tray contiene: [1: Frame de inicio,
%                2: Frame de fin,
%                3: Nro de punto]

Tray=zeros(1,3);
j=1; %Variable auxiliar
%Para todos los puntos encontrados
for i=1:ident_index_size
    %temp_tray: contiene información del punto i encontrado
    temp_tray=traj_mask_temp(i,:);

    %t_der: contiene el inicio y final de la trayectoria del punto i
    %Esto mediante una diferencia entre la trayectoria actual y la
    %trayectoria desfasada.
    t_der=temp_tray(2:end)-temp_tray(1:end-1);

    %index_ini: Inicio de la trayectoria, en donde se halla el valor de
1
    index_ini=find(t_der==1);
    %index_ini: Fin de la trayectoria, en donde se halla el valor de -1
    index_fin=find(t_der==-1);

    %En caso no se encuentre el inicio o el final, se coloca un 1 segun
sea
    % el caso
    if isempty(index_ini)
        index_ini=1;
    end
    if isempty(index_fin)
        index_fin=frame_size;
    end

    %Trabajamos con trayectorias mayores a 20 frames
    if index_fin - index_ini >20 %Condición: Trayectorias > a 20 frames
    Tray(j,:)=[index_ini index_fin i];
    j=j+1;
    end
end

%Mascaras temporales: tray2 y tray3

tray2=zeros(size(traj_mask_temp)); %Mascara
tray3=zeros(size(traj_mask_temp)); %Mascara de Coherencia Temporal

%NumPts: número total de puntos
NumPts=size(Tray,1);
%Para todos los puntos
for i=1:NumPts
        %Asignar pesos:
        %   Frames  1 al 10    = 0.5
```

```
            %   Frames  11 al ultimo = 1
            tray2(Tray(i,3),Tray(i,1):Tray(i,1)+9)=0.5;
            tray2(Tray(i,3),Tray(i,1)+10:Tray(i,2))=1;

            %Matriz de Coherencia Temporal

            tray3(Tray(i,3),Tray(i,1):Tray(i,2))=1;

            %Se encuentra la mitad de la trayectoria,
            %si esta es mayor a 50 frames, el valor de umbral
            %queda en 50, si es menor queda en la mitad.

            temp_long=Tray(i,2)-Tray(i,1);
            mitad=floor(temp_long/2);

            if mitad > 50;
                umbral = 50;
            else
                umbral = mitad;
            end

            %Asignamos pesos:
              %El valor del peso del frame irá incrementando en 0.02
            %desde el frame de inicio hasta llegar al frame umbral,
            %una vez alcanzado el valor del umbral, el peso será 1
            %si el umbral es 50, en otro caso será de 0.02 x umbral.
            for j=Tray(i,1):Tray(i,1)+umbral
                tray3(Tray(i,3),j)=(j-Tray(i,1)+1)*0.02;
            end

            for j=Tray(i,2)-umbral:Tray(i,2)
                tray3(Tray(i,3),j)=(Tray(i,2)-j+1)*0.02;
            end
end

ind_val=Tray(:,3);
tray=tray2(ind_val(:),:);     %Máscara de trayectorias con pesos
tray_coh=tray3(ind_val(:),:); %Máscara de coherencia temporal

% figure;imagesc(tray); Opcional: Visualizar máscaras de trayectorias y
% figure;imagesc(tray_coh);         coherencia temporal


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

%% 04. Matriz de Puntos

%Inicialización de matrices auxiliares
tray_x=zeros(NumPts,frame_size);
tray_y=zeros(NumPts,frame_size);
tray_mask=traj_mask_temp;

%Creación de matriz de puntos: Puntos
%Puntos contiene: [1: Nro. de punto
%                  2: Par de coordenadas
```

```
%                    3: Nro. Frame ]
Puntos=zeros(NumPts,2,frame_size);

for i=1:NumPts
    j=ind_val(i);

    tray_x(i,Tray(i,1):Tray(i,2))=traj_x_mat(j,Tray(i,1):Tray(i,2));
    tray_y(i,Tray(i,1):Tray(i,2))=traj_y_mat(j,Tray(i,1):Tray(i,2));
    Puntos(i,1,:)=tray_y(i,:);
    Puntos(i,2,:)=tray_x(i,:);

end


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

%% 05. Normalización

%Pts_Normaliz: Matriz de puntos normalizados
%Normalización: Posición actual (R,C) / Tamaño máximo (R,C)

Pts_Normaliz=zeros(NumPts,2,frame_size);
Pts_Normaliz(:,1,:)=Puntos(:,1,:)/SizeR;
Pts_Normaliz(:,2,:)=Puntos(:,2,:)/SizeC;

%-----------------------------------------------------
%% 06. 1era Función de Transferencia Epipolar

%Puntos Virtuales, P1: puntos virtuales hallados
[P1, ~]= epipolarPT1_2015(Pts_Normaliz,tray,frame_size,NumPts);

%-----------------------------------------------------
%% 07. 1er Filtrado de Trayectorias

%Filtrado de trayectorias con filtro Gaussiano
PF1=filtrado_tray_2015(P1,Tray,25,5,NumPts);

%-----------------------------------------------------
%% 08. 2da Función de Transferencia Epipolar

[P2,~]=epipolarPT2_2015(frame_size,Pts_Normaliz,PF1,tray,NumPts,Pts_Nor
maliz);

%-----------------------------------------------------
%% 09. Desnormalización
PuntosD=zeros(NumPts,2,frame_size);

for j=1:frame_size
    for i=1:NumPts
        PuntosD(i,1,j)=P2(i,1,j)*SizeR;
        PuntosD(i,2,j)=P2(i,2,j)*SizeC;
    end
end
%-----------------------------------------------------
%% 10. 2da. Etapa de Filtrado
```

```
P3=filtrado_tray_2015(PuntosD,Tray,25,5,NumPts);
PuntosN=P3;

%-------------------------------------------------------
%% 11. Frame Warping y Remuestreo

%Definimos el tamaño de PixelSize
%Nro de cuadrantes en Filas

PixelSize=SizeR/5;

for f=13:200
    %Asumimos un valor de alpha constante = 20
    alpha = 20;
    %Función warping_final
    [Vn25, L, num_cells_r, num_cells_c, final_r, final_c, P_val,
alpha]=warping_2015(Puntos,PuntosN,tray_coh,f, SizeR, SizeC,
PixelSize,alpha) ;



    str=['000' num2str(f-1)];
    s=size(str);
    str=str((s(2)-3):s(2));
     I=imread(strcat('B:\Hiro\PUCP\TESIS\tesis\tesis2\Referencias\Banco
de Pruebas\video liu\liu_jpg\liu',str,'.jpg'));

    Iw=warp_2015(Vn25,PixelSize, num_cells_r,
num_cells_c,I,final_r,final_c);
    Iw2=Iw(21:final_r-20,21:final_c-20,:);
    disp(f);
    I2=I(21:final_r-20,21:final_c-20,:);
    Inew=[I2 Iw2];
    figure;imshow(Inew);axis off;title('Video Distorsionado
Video Estabilizado');
    print( [ 'B:\Hiro\PUCP\TESIS\tiempos2015\last_liu_' num2str( f)],
'-djpeg99', '-r150');
    close all;
end
```