Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

VISUALIZACIÓN DE PROCESOS DE COMBUSTIÓN CON CÁMARA DE ALTA VELOCIDAD (ANEXOS)

Autor:
Rodríguez González, Mauro

Tutor:
Dr. Melgar Bachiller, Andrés
Ingeniería Energética y Fluidomecánica

Valladolid, mayo 2018

Contenido

1. CÓDIGO MATLAB

Se compone de:

- 1 documento principal
    1. AnalisisVideo.m
- 15 documentos con funciones
    1. ControlTratamiento.m
    2. coordPixelsCont.m
    3. dibujaCirc.m
    4. dibujaCircABC.m
    5. dibujaRectaABC.m
    6. eligePuntosRand.m
    7. ExcelCol.m
    8. GeneraMascara.m
    9. InliersCirc.m
    10. InliersRecta.m
    11. LMSCirc.m
    12. LMSRecta.m
    13. ransacCirc2.m
    14. ransacRecta.m
    15. RestringeRadio.m

AnalisisVideo.m

```
clc; clear; close all;

%% INICIALIZACION
global UnaVez;
global Parar;
global Terminar;
global Avante;
global NumFotograma;
UnaVez = 0;
Parar = 0;
Terminar = 0;
Avante = 1;
NumFotograma = 0;
Ejecutar = 1;

RadioMaximo = 1000;
NumeroIteracionesRansac = 100;
AreaCelulaMinima = 1;
VelocidadCaptura = 3124; %fps

TamanoPantalla = get(groot,'ScreenSize');
PosicionFigu = [10 50 TamanoPantalla(3)*0.8
TamanoPantalla(4)*0.83];

Nombre_Video = 'H2_fr07_p100kPa_T30';
Nombre_Extension_Video = 'avi';
Directorio_Codigo = cd;
```

```
Directorio_Videos = cd('Videos\BujiaCentrada\H2\2017-10-17');
Video_Info = VideoReader(strcat(Nombre_Video, '.',
Nombre_Extension_Video));
Carpeta = cd(Directorio_Codigo);

Num_Fotogramas = Video_Info.Numberofframes;
Primer_Fotograma = read(Video_Info, 1);
NombreExcel = strcat('Datos_', Nombre_Video, '.xlsx');

%% CALCULOS PREVIOS
%RADIO DE LA CAMARA
Contorno = edge(Primer_Fotograma);
[X, Y] = coordPixelsCont(Contorno, RadioMaximo, 0, 0);
[xInliers, yInliers] = ransacCirc2(X, Y, 50000, RadioMaximo,0);
[A, B, C] = LMSCirc(xInliers, yInliers);
RadCam = sqrt(A^2+B^2-4*C)/2;
CntCamX = -A/2;
CntCamY = -B/2;
Calibracion = RadCam/5.7; %pixel/cm

%% BUCLE
ControlTratamiento();
movegui(ControlTratamiento, 'east');

Nini = 1;
Nfin = Num_Fotogramas;
NumeroFotograma = Nini;

%Variables Salida
radio = zeros(1, Nfin-Nini);
tiempo = zeros(1, Nfin-Nini);
NumeAreas = zeros(1, Nfin-Nini);
Sn = zeros(1, Nfin-Nini+1);
Stretch = zeros(1, Nfin-Nini+1);

while (Terminar == 0)
    if (NumFotograma == 1)
        NumeroFotograma = Nini;
        NumFotograma = 0;
    elseif (NumFotograma == -1)
        NumeroFotograma = Nfin;
        NumFotograma = 0;
    end
    if (Parar == 0 || UnaVez == 1)
        if (Avante == 1)
            if (NumeroFotograma < Nfin)
                NumeroFotograma = NumeroFotograma + 1;
            else
                NumeroFotograma = Nini;
            end
        else
            if (NumeroFotograma > Nini)
                NumeroFotograma = NumeroFotograma - 1;
            else
                NumeroFotograma = Nfin;
            end
        end
        UnaVez = 0;
```

```
        Ejecutar = 1;
    else
        pause(0.001);
    end
    if (Ejecutar == 1)
        Fotograma = read(Video_Info, NumeroFotograma);
        Linea(:,1) = Fotograma(Video_Info.Height/2,:);

        %RADIOS FRENTE DE LLAMA
        Contorno2 = edge(Fotograma).*(1 - Contorno);
        [X, Y] = coordPixelsCont(Contorno2, RadCam, CntCamX,
CntCamY);
        if NumeroFotograma == Nini
            [xInliers, yInliers] = ransacCirc2(X, Y,
NumeroIteracionesRansac, RadCam*.9, 0);
        else
            [xInliers, yInliers] = ransacCirc2(X, Y,
NumeroIteracionesRansac, RadCam*.9, radio(NumeroFotograma-Nini));
        end
        [A, B, C] = LMSCirc(xInliers,yInliers);
        radio(NumeroFotograma-Nini+1) = (sqrt(A^2+B^2-
4*C)/2)/Calibracion;
        tiempo(NumeroFotograma-Nini+1) = (NumeroFotograma-
Nini)*(1000/VelocidadCaptura);

        %CELULARIDAD
        MatrizRMS = stdfilt(Fotograma);
        MatrizRMS  = RestringeRadio(MatrizRMS , CntCamX ,CntCamY,
Calibracion*radio(NumeroFotograma-Nini+1)*0.8);
        FotogramaRMS = im2uint8(MatrizRMS./(max(max(MatrizRMS))));
        FotogramaRMS_Binario = im2bw(FotogramaRMS, 0.1);
        FotogramaRMSBinarioSinRuido =
bwareaopen(FotogramaRMS_Binario, 50);
        FotogramaAdelgazado =
bwmorph(FotogramaRMSBinarioSinRuido,'thin','Inf');
        FotogramaRMSBinarioInversa =
imcomplement(FotogramaAdelgazado);
        ComponentesConexos =
bwconncomp(FotogramaRMSBinarioInversa, 4);
        Propiedades = regionprops(ComponentesConexos);
        for i = 1:length(Propiedades)
            if (Propiedades(i).Area > AreaCelulaMinima)
                NumeAreas(NumeroFotograma-Nini+1) =
ComponentesConexos.NumObjects/pi/(radio(NumeroFotograma-
Nini+1)*0.8)^2;
            end
        end
        Fusion = imadjust(imfuse(imadjust(Fotograma),
FotogramaRMSBinarioInversa, 'blend'));

        %Sn y Stretch

        if(NumeroFotograma-Nini+1 <= Nfin)
            if(NumeroFotograma-Nini+1 == 1)
                continue;
            else
                Sn(NumeroFotograma-Nini+1) =
(radio(NumeroFotograma-Nini+1)-radio(NumeroFotograma-
```

```
Nini))/(tiempo(NumeroFotograma-Nini+1)-tiempo(NumeroFotograma-
Nini)); %dr/dt
                Stretch(NumeroFotograma-Nini+1) =
(Sn(NumeroFotograma-Nini+1))/(radio(NumeroFotograma-Nini));
            end
        end

        if(NumeroFotograma-Nini+1 == 1)
            continue;
        else
             end

%IMPRESION 1
        figure(1);
        nombreplot1 =
strcat('Fotograma:',num2str(NumeroFotograma),'
Tiempo:',num2str((NumeroFotograma-
Nini+1)*(1000/VelocidadCaptura)),'ms');

set(figure(1),'name',nombreplot1,'numbertitle','off','Position',Po
sicionFigu);
        subplot('position', [0 0.5 0.3 0.5]);
        imshow(Fotograma); hold on;
        subplot('position', [0.3 0.5 0.3 0.5]);
        imshow(Contorno2); hold on;
        dibujaCircABC(A, B, C, 'r');
        subplot('position', [0.63 0.6 0.3 0.3]);
        plot(tiempo, radio);
        subplot('position', [0 0 0.3 0.5]);
        imshow(FotogramaRMS_Binario); hold on;
FotogramaRMS_Binario  %FotogramaAdelgazado
%         for i = 1:length(Propiedades)
%             if (Propiedades(i).Area > AreaCelulaMinima)
%                 plot(Propiedades(i).Centroid(1),
Propiedades(i).Centroid(2), 'r.');
%             end
%         end
        subplot('position', [0.3 0 0.3 0.5]);
        imshow(Fusion); hold on;
%         for i = 1:length(Propiedades)
%             if (Propiedades(i).Area > AreaCelulaMinima)
%                 plot(Propiedades(i).Centroid(1),
Propiedades(i).Centroid(2), 'r.');
%             end
%         end
        subplot('position', [0.63 0.1 0.3 0.3]);
        plot(tiempo, NumeAreas);


%IMPRESION 2
        figure(2);
        set(figure(2),'Position',PosicionFigu);
        subplot('position', [0 0 0.5 1]);
        imshow(Fotograma);
        subplot('position', [0.55 0.2 0.45 0.65]);
        plot(Linea, 'b');
        grid on;
```

```
%IMPRESION 3
        figure(3);
        set(figure(3),'Position',PosicionFigu);
        FotogramaBinario = im2uint8(FotogramaAdelgazado);
        FotogramaBinario = imcomplement(FotogramaBinario);
        FotogramaFusionado = imadjust(imfuse(FotogramaBinario,
Fotograma, 'blend'));
        subplot('position', [0 0 0.5 1]);
        imshow(Fotograma); hold on;
        subplot('position', [0.5 0.25 0.5 0.5]);
        imshow(FotogramaFusionado); hold on;
        for i = 1:length(Propiedades)
            plot(Propiedades(i).Centroid(1),
Propiedades(i).Centroid(2), 'r.', 'MarkerSize', 5);
        end



        fprintf('**************************\n');
        fprintf('Nombre Video: %s\n', Video_Info.Name);
        fprintf('Fotograma: (%d/%d)\n', NumeroFotograma-Nini+1,
Num_Fotogramas);
        fprintf('Tiempo: %.2f (ms)\n', tiempo(NumeroFotograma-
Nini+1));
        fprintf('radio: %.2f (cm)\n', radio(NumeroFotograma-
Nini+1));
        fprintf('N celulas: %.2f (N/cm2)\n',
NumeAreas(NumeroFotograma-Nini+1));
        fprintf('Sn: %.2f (N/cm2)\n', Sn(NumeroFotograma-Nini+1));
        fprintf('Stretch: %.2f (N/cm2)\n',
Stretch(NumeroFotograma-Nini+1));

        Ejecutar = 0;
        %Solo ejecutar una vez
        if NumeroFotograma == Nfin
            Terminar = 1;
        end
    end
end

%% GUARDAR FICHERO
NombreVideo = Video_Info.Name(1:end-4);
%TABLA Titulo de cada video
TablaTitulo = table({NombreVideo});
%TABLA Variables de cada video
TablaDatos = table;
TablaDatos.tiempo = tiempo';
TablaDatos.radio = radio';
TablaDatos.NumeAreas = NumeAreas';
TablaDatos.Sn = circshift(Sn', [-1, 0]);
TablaDatos.Stretch = circshift(Stretch', [-1, 0]);

TamanoTabla = size(TablaDatos);
for i = 1:TamanoTabla(1)
    for j = 1:TamanoTabla(2)
        if(strcmp(TablaDatos(i, j).Variables, 'Inf'))
            TablaDatos(i, j) = {0};
        end
```

```
    end
end
RangoExcel1 = strcat(char(ExcelCol(1)), char('1'));
RangoExcel2 = strcat(char(ExcelCol(1)), char('2'));

writetable(TablaTitulo, NombreExcel, 'Range', RangoExcel1,
'WriteVariableNames', 0);
writetable(TablaDatos, NombreExcel, 'Range', RangoExcel2);
```

ControlTratamiento.m

```
function varargout = ControlTratamiento(varargin)
% CONTROLTRATAMIENTO MATLAB code for ControlTratamiento.fig
%      CONTROLTRATAMIENTO, by itself, creates a new
CONTROLTRATAMIENTO or raises the existing
%      singleton*.
%
%      H = CONTROLTRATAMIENTO returns the handle to a new
CONTROLTRATAMIENTO or the handle to
%      the existing singleton*.
%
%
CONTROLTRATAMIENTO('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in CONTROLTRATAMIENTO.M with the
given input arguments.
%
%      CONTROLTRATAMIENTO('Property','Value',...) creates a new
CONTROLTRATAMIENTO or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before ControlTratamiento_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
ControlTratamiento_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
ControlTratamiento

% Last Modified by GUIDE v2.5 17-Jul-2017 15:16:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@ControlTratamiento_OpeningFcn, ...
                   'gui_OutputFcn',
@ControlTratamiento_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

end

% --- Executes just before ControlTratamiento is made visible.
function ControlTratamiento_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ControlTratamiento (see
VARARGIN)

% Choose default command line output for ControlTratamiento
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ControlTratamiento wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
end

% --- Outputs from this function are returned to the command
line.
function varargout = ControlTratamiento_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
end

% --- Executes on button press in Pausa.
function Pausa_Callback(hObject, eventdata, handles)
% hObject    handle to Pausa (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of Pausa
global Parar;
if (Parar == 0)
    Parar = 1;
else
    Parar = 0;
end
end

% --- Executes on button press in Atras.
function Atras_Callback(hObject, eventdata, handles)
% hObject    handle to Atras (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of Atras
global Avante;
global Parar;
global UnaVez;
Avante = 0;
if (Parar == 1)
    UnaVez = 1;
end
end

% --- Executes on button press in Adelante.
function Adelante_Callback(hObject, eventdata, handles)
% hObject    handle to Adelante (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of Adelante
global Avante;
global Parar;
global UnaVez;
Avante = 1;
if (Parar == 1)
    UnaVez = 1;
end
end

% --- Executes on button press in Primer_fotograma.
function Primer_fotograma_Callback(hObject, eventdata, handles)
% hObject    handle to Primer_fotograma (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global NumFotograma;
NumFotograma = 1;
end

% --- Executes on button press in ultimo_fotograma.
function ultimo_fotograma_Callback(hObject, eventdata, handles)
% hObject    handle to ultimo_fotograma (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global NumFotograma;
NumFotograma = -1;
end



% --- Executes on button press in terminar.
function terminar_Callback(hObject, eventdata, handles)
% hObject    handle to terminar (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Terminar;
Terminar = 1;
end
```

coordPixelsCont.m

```
function [X,Y] = coordPixelsCont(C,Rd,Cx,Cy)
%Devuelve dos vectores con las x(colum) y las y(filas)
%de los pixels activos que hay en la imagen binaria C
%Si no encuentra nada devuelve
%    X(1)=-1;
%    Y(1)=-1;
% Se podría hacer [Y,X] = find(C);

[filas, columnas] = size(C);
numPixelsCont = 0;
for i = 1:filas
    for j = 1:columnas
        if C(i,j) == 0
            continue;
        else
            if ((i-Cy)^2+(j-Cx)^2<(Rd*0.95)^2)
                numPixelsCont = numPixelsCont + 1;
                X(numPixelsCont) = j; %coord x
                Y(numPixelsCont) = i; %coord y
            end
        end
    end
end
if numPixelsCont == 0 %Si no encuentra nada devuelve -1
    X(1) = -1;
    Y(1) = -1;
end
end
```

dibujaCirc.m

```
function dibujaCirc(centro, R, color)
%Dibujar circunferencia a partir del centro y radio.
%Para sobreimpresionar visualizar previamente imagen con
imshow(I); hold on

plot(centro(1), centro(2), [color, '+']);
i = 1;
for theta = 0:0.01:2*pi
```

```
    x(i) = centro(1) + R*cos(theta);
    y(i) = centro(2) + R*sin(theta);
    i = i + 1;
end
hold on;
plot(x, y, [color, '-']);
```

dibujaCircABC.m

```
function dibujaCircABC(A, B, C, color)
%para sobreimpresionar, la imagen tiene que ser visualizada
previamente con imshow(I); hold on;
radio = sqrt(A^2 + B^2 - 4*C)/2;
centro = [-A/2, -B/2];
dibujaCirc(centro, radio, color)
```

dibujaRectaABC.m

```
function dibujaRectaABC(A, B, C, color)
%Dibuja recta Ax + By + C = 0.  Eje X es horizontal
%Visualizar imagen previamente y mantener con hold on
% Ejemplo:
% I=imread('cameraman.tif');
% imshow(I);
% hold on;
% dibujaRectaABC(A,B,C,'g');
%-------------------------------------------------------------------
-------
if B == 0 %Recta vertical, pendiente infinita
    %punto inicial
    P(1) = -C;
    P(2) = 1;
    %punto final
    Q(1) = -C;
    Q(2) = 600; %valor alto para que valga para img grandes
    dibujaRectaPQ(P, Q, color);
else
    %lo pongo en la forma y=mx+b
    m = -A/B;
    b = -C/B;
    x(1) = 0;
    y(1) = b;
    x(2) = 600; %valor alto para que valga para img grandes
    y(2) = m*x(2) + b;
    plot(x, y, [color, '-']);
end
```

eligePuntosRand.m

```
function [xSample, ySample] = eligePuntosRand(xPixelsCont,
yPixelsCont, N)
% Elige N puntos aleatoriamente del conjunto
(xPixelsCont,yPixelsCont)
% Devuelve dos vectores: xSample con las coord x y ySample con
las coord y
xSample = zeros(N, 1);
```

```
ySample = zeros(N, 1);
numPixelsCont = length(xPixelsCont);
% N numero de puntos aleatorios
for k = 1:N
    indexRand = floor(numPixelsCont*rand()) + 1;
    xSample(k) = xPixelsCont(indexRand);
    ySample(k) = yPixelsCont(indexRand);
end
end
```

ExcelCol.m

```
function Out=ExcelCol(In)
%EXCELCOL  Converts between column name and number for Excel
representation
%   Out=ExcelCol(In) takes the input In, which may be a number,
vector,
%   char, or cell and converts it to the other representation
%
%   If IN is numeric, output will be a column cell of the column
name
%   If IN is char or cell, output will be a number or column
vector,
%      ignoring any numberic part which may be included in input
%
%   EXAMPLES:
%   ExcelCol(100)                          %Number to column name
%   ExcelCol('CV')                         %Column name to number
%   ExcelCol([1 10 100 1000 16383])        %Multiple conversions
%   ExcelCol({'A' 'J' 'CV' 'ALL' 'XFC'}) %Multiple conversions
%
%


%
% $ Author: Mike Sheppard
% $ Original Date: 4/7/2010
% $ Version: 1.0


%Optional to change representation and base
ABC=['ABCDEFGHIJKLMNOPQRSTUVWXYZ'];
base=26;

if isnumeric(In)
%Converts from column number to alpha
%1=A, 2=B,... 26=Z, 27=AA, ... 16383=XFC
In=In(:);
if ~all(In>0)
error('MATLAB:ExcelCol:NegativeColumnNumber', 'Column numbers
must be positive');
end
  for row=1:size(In,1)
   diff=1;
   i=0;
   n=In(row,:);
   while diff<=n
       letter_ind=1+mod(floor((n-diff)/base^i),base);
```

```
        i=i+1;
        temp(i)=ABC(letter_ind);
        diff=diff+base^i;
    end
    Out{row}=fliplr(temp);
    clear temp
  end
  Out=Out(:);
else
%Converts from alpha to column number
%A=1, B=2, ..., Z=26, AA=27, ... XFC=16383
    In=cellstr(upper(In));
    In=In(:);
    for row=1:size(In,1)
        alpha=char(In(row,:));
        %Delete any numbers which may appear
        alpha=(char(regexp(alpha,'\D','match')))';
        lng=length(alpha);
        temp=((base^(lng) - 1) / (base-1));
        for i=1:lng
            ind=strfind(ABC, alpha(i));
            if isempty(ind)  %ERROR
                  error('MATLAB:ExcelCol:Mixofcharacters', 'Must
be only alpha-numeric values {A-Z}, {a-z}, {0-9}');
            end
            temp=temp+(ind-1)*(base^(lng-i));
        end
        Out(row)=temp;
    end
    Out=Out(:);
end


end
```

GeneraMascara.m

```
function [ImagenModificada] = GeneraMascara(tamano, CentroX,
CentroY, Radio)
ImagenModificada = zeros(tamano, 'uint8');
for i = 1:tamano(1)
    for j = 1:tamano(2)
        if (i-CentroY)^2+(j-CentroX)^2 <= Radio^2
            ImagenModificada(i, j) = 1;
        end
    end
end
end
```

InliersCirc.m

```
function [xInliers, yInliers] = InliersCirc(x, y, centro, radio,
tolerancia)
% Calcula los inliers en una circunferencia con centro y radio
que hay en
% el conjunto de píxeles con coord x e y (vectores con coord x y
```

```
coord y)
% C es la imagen de contornos.
% Hasta 10.000 inliers por preasignacion de memoria
numInliers = 0;
%franja de consenso (R1,R2)
R1 = radio - tolerancia;
R2 = radio + tolerancia;
numPixelsCont = length(x);
%preasignacion para mayor rapidez
xInliers = zeros(1, 10000);  %hasta 10.000 inliers
yInliers = zeros(1, 10000);

for k = 1:numPixelsCont
    distCentro = sqrt((x(k) - centro(1))^2 + (y(k) -
centro(2))^2);
    % Si el pixel entra dentro de la franja de consenso(R1,R2)
    if ((distCentro > R1) && (distCentro < R2))
        numInliers = numInliers + 1;
        xInliers(numInliers) = x(k);
        yInliers(numInliers) = y(k);
    end
end
%Devuelve dos vectores con longitud numInliers
%(si numInliers es cero tambien funciona)
xInliers = xInliers(1:numInliers);
yInliers = yInliers(1:numInliers);
end
```

InliersRecta.m

```
function [xInliers, yInliers] = InliersRecta(x, y, A, B, C,
tolerancia)
%Calcula inliers de la recta Ax + By + C = 0
% Devuelve [A,B,C].
% Hasta 10.000 inliers por preasignacion de memoria
%Ejemplo:
% C=edge(I);
% [X,Y] = coordPixelsCont(C);
% [xInliers,yInliers] = InliersRecta( x,y,A,B,C,tolerancia)
%  n(A,B) unitario para calculo dist(P,r)
normAB = sqrt(A^2 + B^2);
A = A / normAB;
B = B / normAB;
C = C / normAB;
%preasignacion para mayor rapidez
xInliers = zeros(1, 10000);  %hasta 10.000 inliers
yInliers = zeros(1, 10000);
numInliers = 0;
for k = 1:length(x)
    distPuntoRecta = abs(A*x(k) + B*y(k) + C);
    if (distPuntoRecta < tolerancia) % Si el pixel se ajusta a la
recta
        numInliers = numInliers + 1;
        xInliers(numInliers) = x(k);
        yInliers(numInliers) = y(k);
    end
end
```

```
xInliers = xInliers(1:numInliers);
yInliers = yInliers(1:numInliers);
if (numInliers == 0) % Si no encuentra inliers devuelve [-1,-1]
    xInliers(1) = -1;
    yInliers(1) = -1;
end
end
```

LMSCirc.m

```
function [A, B, C] = LMSCirc(x, y)
%minimos cuadrados circunferencia
% Devuelve los parametros A,B,C de la circunferencia
x2+y2+Ax+By+C=0
%que ajusta los puntos con las coord en los vectores x,y
%Ejemplo ejecución:
% I=imread('chaveta01.png');
% imshow(I);
% J=imfill(I,'holes');
% C=edge(J);
% imshow(C);
% [Y,X] = find(C);
% [A,B,C]=LMSCirc(X,Y);
% hold on;
% dibujaCircABC(A,B,C, 'r');
suma_x = 0;
suma_x2 = 0;
suma_y = 0;
suma_y2 = 0;
suma_x_x2_y2 =0;
suma_y_x2_y2 = 0;
suma_x2_y2 = 0;
suma_xy = 0;

n=length(x);
for k=1:n
    suma_x=suma_x + x(k);
    x2 = x(k)*x(k);
    suma_x2 = suma_x2 + x2;
    suma_y = suma_y + y(k);
    y2 = y(k)*y(k);
    suma_y2 = suma_y2 + y2;
    s_x2_y2 = x2 + y2;
    suma_x_x2_y2 = suma_x_x2_y2 + x(k)*s_x2_y2;
    suma_y_x2_y2 = suma_y_x2_y2 + y(k)*s_x2_y2;
    suma_x2_y2 = suma_x2_y2 + s_x2_y2;
    suma_xy = suma_xy + x(k)*y(k);
end

M1 = zeros(3);
M1(1,1) =  suma_x2;
M1(1,2) =  suma_xy;
M1(1,3) =  suma_x;
M1(2,1) =  suma_xy;
M1(2,2) =  suma_y2;
M1(2,3) =  suma_y;
M1(3,1) =  suma_x;
```

```
M1(3,2) =  suma_y;
M1(3,3) =  n;

M2(1,1) = -suma_x_x2_y2;
M2(2,1) = -suma_y_x2_y2;
M2(3,1) = -suma_x2_y2;

inversaM1 = pinv(M1);
R = inversaM1*M2;

A = R(1);
B = R(2);
C = R(3);
```

LMSRecta.m

```
function [A,B,C] = LMSRecta(x,y)
%Ajusta por minimos cuadrados a recta Ax + By + C = 0 calcula eje
inercia
% Devuelve [A,B,C]. n(A,B) unitario. Viene bien para calculo
dist(P,r)
% x,y contiene las x (columnas)y las y (filas) de los pix de cont

sum_x=0;                %sumatorio x
sum_y=0;                %sumatorio j
Ixx = 0;
Iyy = 0;
Ixy = 0;
n=0;

% Media de las coord X e y (centro gravedad)
cx=mean(x);
cy=mean(y);
centro=[cx,cy];

n=length(x);
for k=1:n
    Ixx = Ixx + (x(k) - cx)^2;
    Iyy = Iyy + (y(k) - cy)^2;
    Ixy = Ixy + (x(k) - cx)*(y(k) - cy);
end

%Tita es el angulo de la recta con el eje x (horiz)
tita=0.5*atan2( 2*Ixy, Ixx-Iyy );
m=tan(tita);
b=cy-m*cx;
% mx-y+b=0.
A=m; B=-1; C=b;

%Normalizo n(A,B)=1
n=[A,B];
normAB=norm(n);
A = A/normAB;
B = B/normAB;
C = C/normAB;
```

```
end
```

ransacCirc2.m

```
function [xInliers_max, yInliers_max] = ransacCirc2(xPixelsCont,
yPixelsCont, numMinInliers, rmax, rmin)
% RANSAC Circ que finaliza cuando alcanza un num.  superior a
numMinInliers
% xPixelsCont,yPixelsCont contiene las x % (columnas)y las y
(filas) de los
% pix de cont
% numMinInliers: Itera hasta que encuentre al menos numInliers
% xInliers son las columnas!
% yInliers son las filas!
% xInliers e yInliers hay que pasarlos LMS para calcular
circunferencia
% con mejor precisión
% ------------------------- EJECUCION---------------------------
-----
% Cont=edge(I);
% [X,Y] = coordPixelsCont(Cont);   %coord punt.contorno a vectores
X e Y
% [xInliers,yInliers] = ransacCirc2( X,Y, 900);    %900 inliers
% [A,B,C]=LMSCirc(xInliers,yInliers);   %ajuste de los inliers por
min.cuadrados
% dibujaCircABC(A,B,C, 'r');
% ------------------------------------------------------------------
----
numInliers = 10;
xInliers_max = zeros(1,numInliers);
yInliers_max = zeros(1,numInliers);
%numIteraciones = 0;
tolerancia = 10;
for i = 1:numMinInliers
    [xSample, ySample] = eligePuntosRand(xPixelsCont, yPixelsCont,
3);
    %circunf que pasa por estos tres puntos
    [A, B, C] = LMSCirc(xSample, ySample);
    radio = sqrt(A^2 + B^2 - 4*C)/2;
    centro = [-A/2, -B/2];
    if (radio < rmax && radio > rmin)
        [xInliers, yInliers] = InliersCirc(xPixelsCont,
yPixelsCont, centro, radio, tolerancia);
        if(length(xInliers) > numInliers)
            numInliers = length(xInliers);
            xInliers_max = xInliers;
            yInliers_max = yInliers;
        end
    end
end
end
```

ransacRecta.m

```
function [mejorxInliers, mejoryInliers] = ransacRecta(x, y,
numMaxIter)
% x,y contiene las x (columnas)y las y (filas) de los pix de cont
```

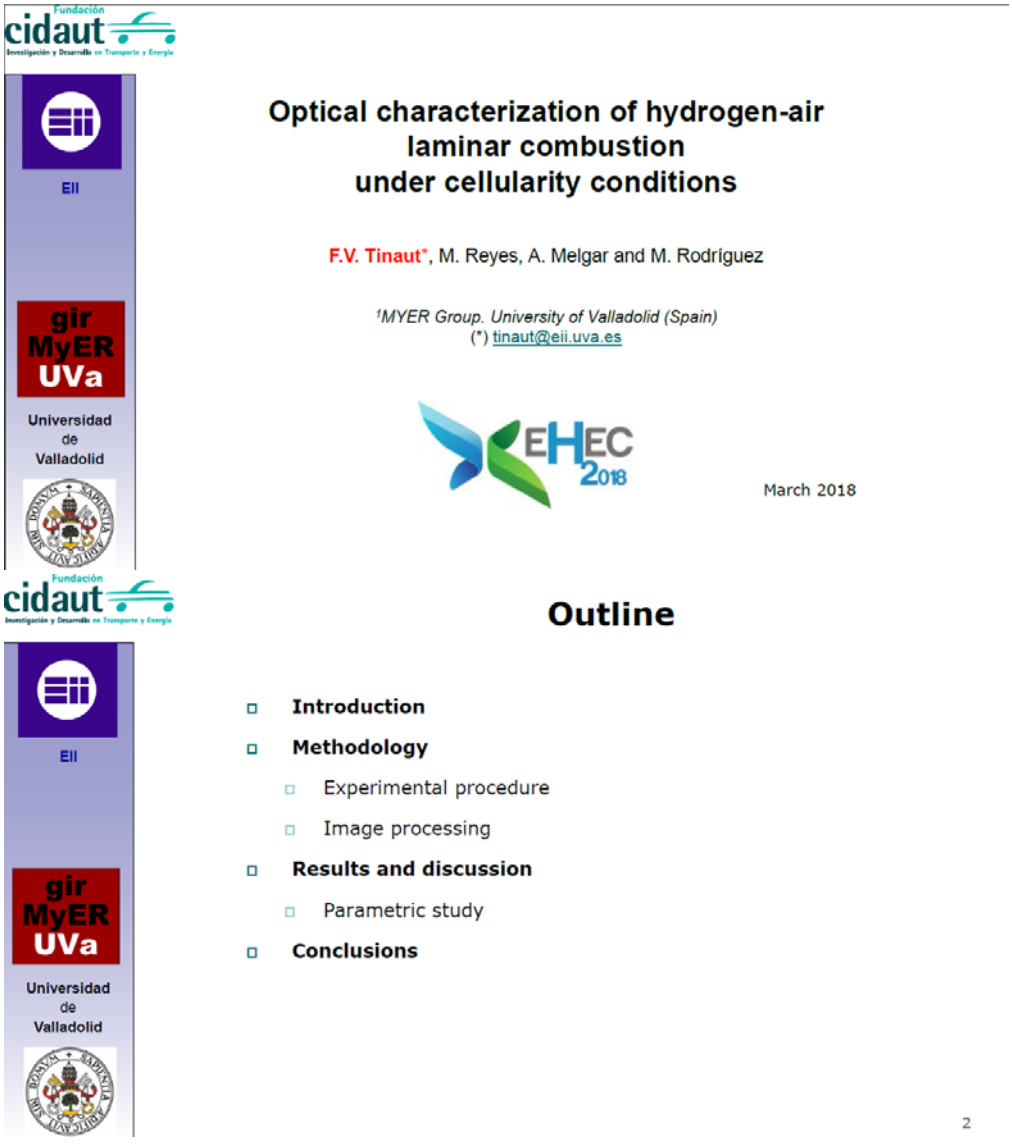
```
% numMaxIter numero máximo de iteraciones
% mejorxInliers son las columnas! del mejor modelo encontrado
% mejoryInliers son las filas!
% xInliers e yInliers hay que pasarlos LMS para afinar recta
% -------------------------- EJECUCION--------------------------
---------
% Cont=edge(I);
% [y,x] = find(Cont);   %coord punt.contorno a vectores X e Y
% [xInliers,yInliers] = ransacRecta( x,y, 100);    %10
iteraciones
% [A,B,C]=LMSRecta(xInliers,yInliers);   %ajuste inliers por
min.cuadrados
% dibujaRectaABC(A,B,C, 'r');
% ----------------------------------------------------------------
---------
tolerancia = 1; %distancia max a recta para que sea inlier
numInliers = 0;
mejorNumInliers = 0; %el numero de inliers más alto q se ha
encontrado
for numIteraciones = 1 : numMaxIter
    [xSample,ySample] = eligePuntosRand(x, y, 2);
    %recta que pasa por estos dos puntos
    [A, B, C] = LMSRecta(xSample, ySample);
    [xInliers, yInliers] = InliersRecta(x, y, A, B, C,
tolerancia);
    numInliers = length(xInliers);
    if numInliers > mejorNumInliers
        mejorNumInliers = numInliers;
        mejorxInliers = xInliers;
        mejoryInliers = yInliers;
    end
end
```

RestringeRadio.m

```
function [ImagenModificada] = RestringeRadio(ImagenOriginal,
CentroX, CentroY, Radio)
[filas, columnas] = size(ImagenOriginal);
ImagenModificada = zeros(size(ImagenOriginal));

for i = 1:filas
    for j = 1:columnas
        if (i-CentroY)^2 + (j-CentroX)^2 >= Radio^2
            ImagenModificada(i, j) = 0;
        else
            ImagenModificada(i, j) = ImagenOriginal(i, j);
            %ImagenModificada(i, j) = 1;
        end
    end
end
end
```

2. OPTICAL CHARACTERIZATION OF HYDROGEN-AIR LAMINAR COMBUSTION UNDER CELLULARITY CONDITIONS



Fundación cidaut
Investigación y Desarrollo en Transporte y Energía

EII

gir MyER UVa

Universidad de Valladolid

Optical characterization of hydrogen-air laminar combustion under cellularity conditions

F.V. Tinaut*, M. Reyes, A. Melgar and M. Rodríguez

[1]MYER Group. University of Valladolid (Spain)
(*) tinaut@eii.uva.es

EHEC 2018

March 2018

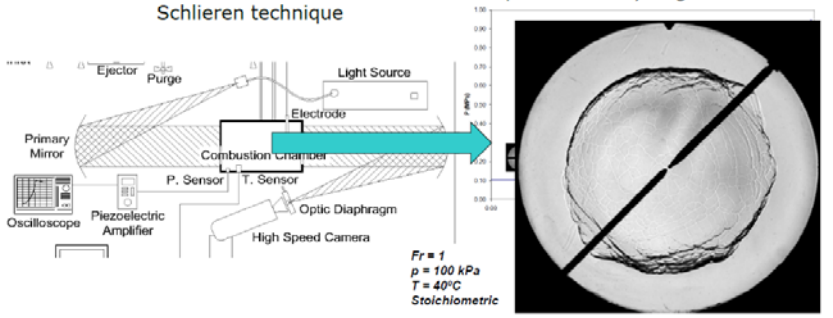
Outline

- **Introduction**
- **Methodology**
  - Experimental procedure
  - Image processing
- **Results and discussion**
  - Parametric study
- **Conclusions**

2

1. Introduction

- Characterization of hydrogen as a **single fuel** in the combustion process inside an **optical-access cylindrical combustion bomb**
- Combustion bomb instrumented:
  - **Instantaneous pressure** during the combustion is registered
  - **Visualization of** the combustion process of hydrogen with Schlieren technique

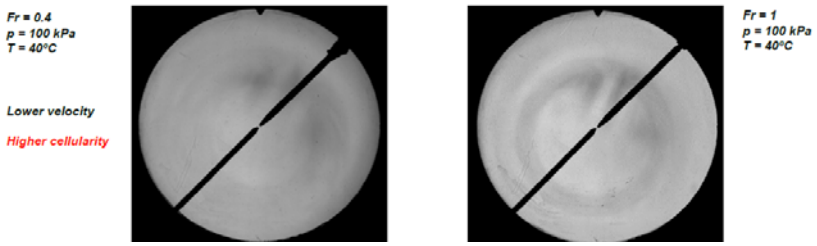


3

1. Introduction

- Characterization of hydrogen as a **single fuel** in the combustion process inside an **optical-access cylindrical combustion bomb**
- Combustion bomb instrumented:
  - **Instantaneous pressure** during the combustion is registered
  - **Visualization of** the combustion process of hydrogen with Schlieren technique

Lean mixture | *Stoichiometric mixture*



4

1. Introduction

- **Cellular structure:** wrinkling of flame front due to instabilities
  - This causes an **increase of apparent combustion velocity**

Fig. 11. Wrinkled flame front sketch

- **OBJECTIVE**: characterization of the *lean combustion* of *hydrogen-air* mixtures to investigate the onset of **cellular instabilities** and quantify their intensity
  - **Cellularity Density (Cell#/cm2)**
  - Identify the dependence of the cellularity density on the main operating variables: **pressure, temperature and equivalence ratio**.

5

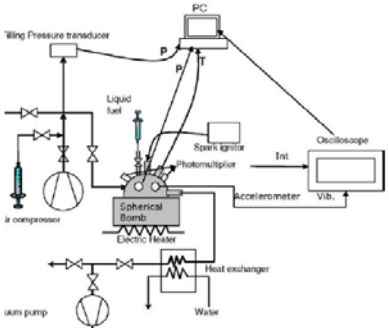
Outline

- Introduction
- **Methodology**
  - **Experimental procedure**
  - **Image processing**
- Results and discussion
  - Parametric study
- Conclusions

6

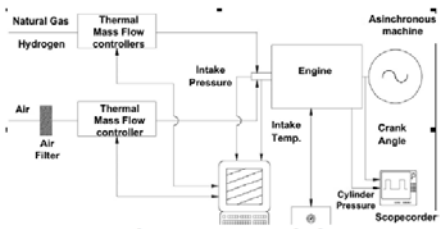
Methodology: Experimental facilities

A. Spherical Constant Volume Combustion Bomb



Moderate pressures (up to 200 bar)
Pressure and Chemiluminescence register

B. Internal Combustion Engine

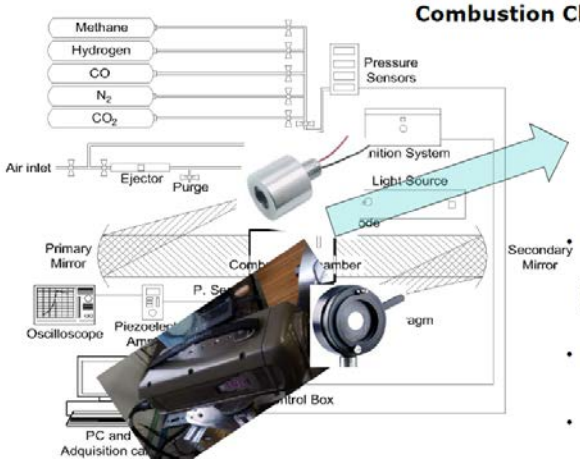


Some results presented in WHEC 2016

7

Experimental facilities

C. Optical-Access Cylindrical Combustion Chamber

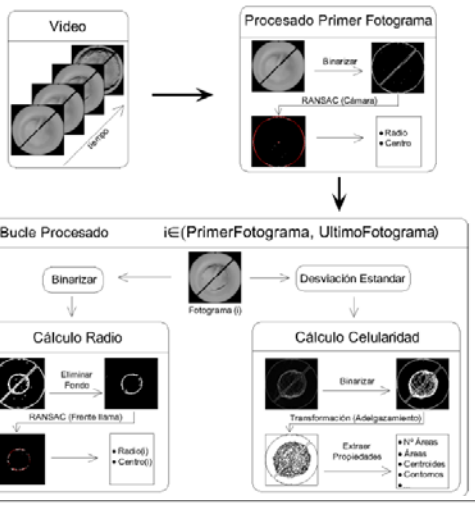


Cylindrical chamber with optical quartz bases
Diameter: 114.3 mm
Length: 135 mm

- Schlieren photography method based on density gradient.
- High speed CMOS camera – up to 300,000 fps

8

Image processing



9

Results

Flame development and cellularity study

- Flame Front Radius
- (Time derivative of Flame Front Radius provides Stretched Flame Velocity)
- Cellularity Density:

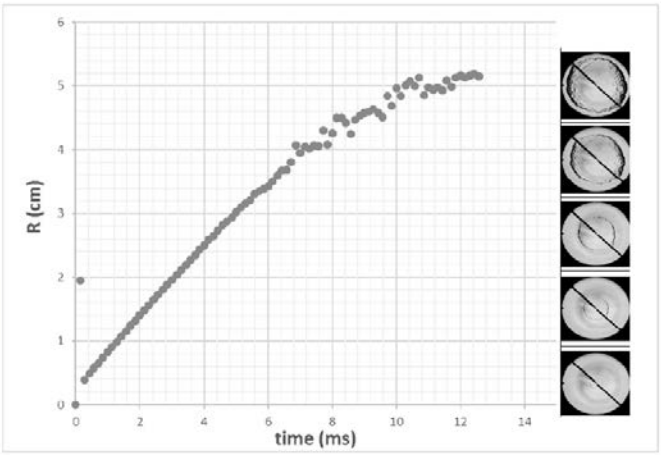
$$Cellularity\ Density\ = Cell\ \#/cm^2$$

- Parametrical study of the influence of:
  - Equivalence ratio F_r: 0.3 to 1.0
  - Pressure (initial pressure, P_i: 0.1 to 3.0 MPa)
  - Temperature (initial temperature, T_i: 40, 80ºC)

Test		Equivalence Ratio Fr					
		0.3	0.4	0.5	0.6	0.7	1.0
Initial Pressure (kPa)	100	X	X	X	X	X	X
	125			X			
	150			X			
	175			X			
	200			X			
	250			X			
	300			X			

10

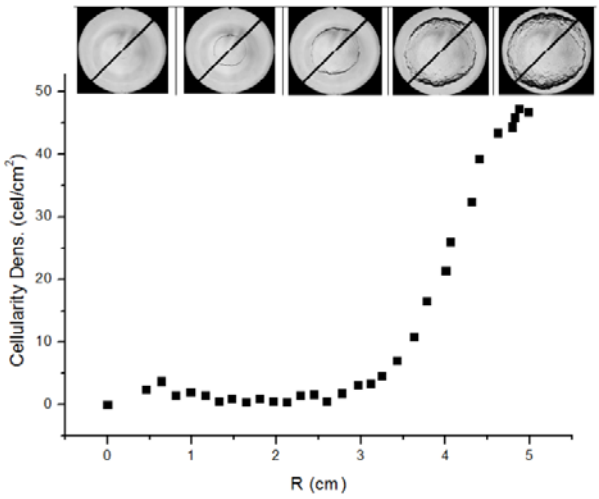
Results: Flame Front Radius vs Time



Fr = 0.6
p = 1 bar
T = 40ºC
Hydrogen

11

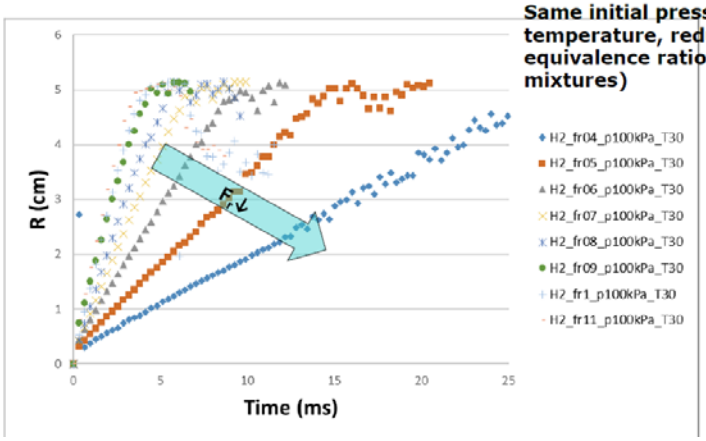
Results: Cellularity Density vs Flame Radius



Fr = 0.6
p = 1 bar
T = 40ºC
Hydrogen

12

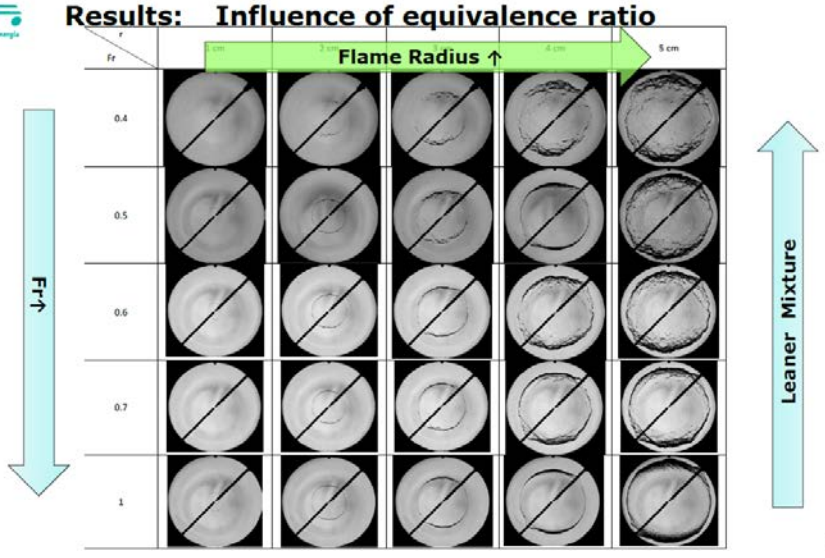
Results: Influence of equivalence ratio
Flame Front Radius vs Time



Same initial pressure and temperature, reduction of equivalence ratio (leaner mixtures)

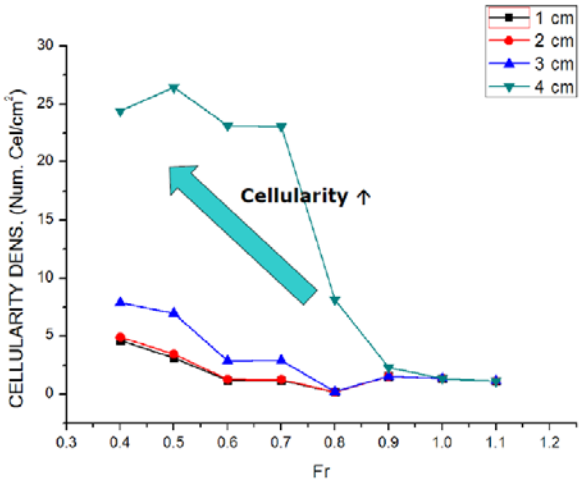
13

Results: Influence of equivalence ratio



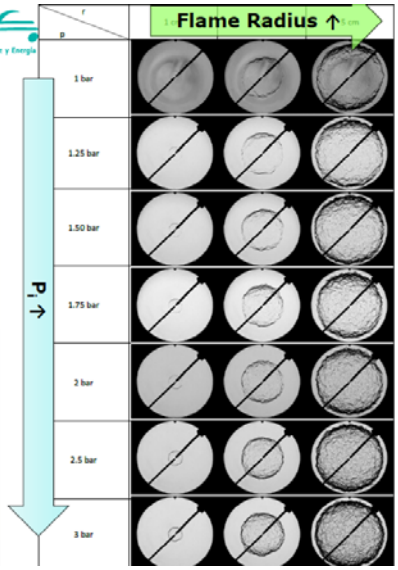
14

Results: Influence of equivalence ratio



15

Results: Influence of pressure

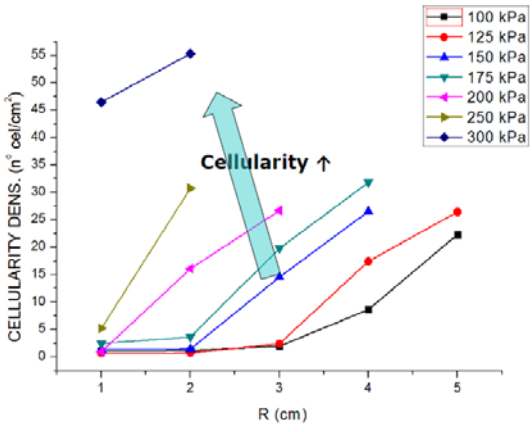


Fr = 0.5
p_i = 0.125 MPa
T_i = 40ºC
Hydrogen

Fr = 0.5
p_i = 0.3 MPa
T_i = 40ºC
Hydrogen

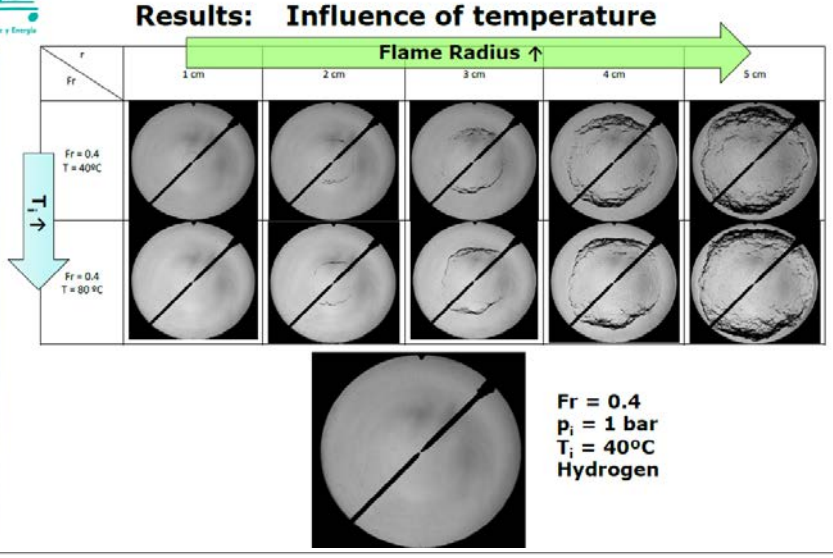
16

Results: Influence of pressure



17

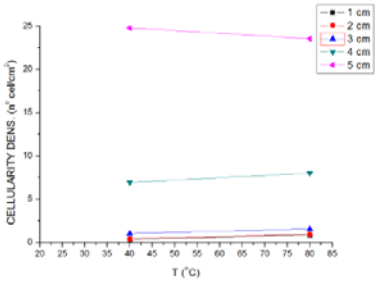
Results: Influence of temperature

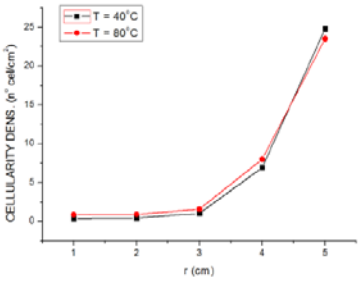


Fr = 0.4
p_i = 1 bar
T_i = 40ºC
Hydrogen

18

Results: Influence of temperature





19

4. Conclusions

- The combustion process of hydrogen-air has been studied using an **optical cylindrical chamber** with a Schlieren technique and **a high-speed camera**.
- An algorithm has been developed to characterize the **cellular structure of the hydrogen-air flames** by accounting for the number of cells per squares cm, under different experimental conditions.
- Results show that **hydrogen-air flames** present **a more cellular character** as:
  - Mixture is **leaner** (reduction of equivalence ratio)
  - Bomb **pressure increases**
  - Bomb **temperature increases** (although the explored range is short).

- The authors **acknowledge the support** of Regional Government of Castilla and León as Excellence Research Group GR203

20

Optical characterization of hydrogen-air laminar combustion under cellularity conditions

F.V. Tinaut*, M. Reyes, A. Melgar and M. Rodríguez

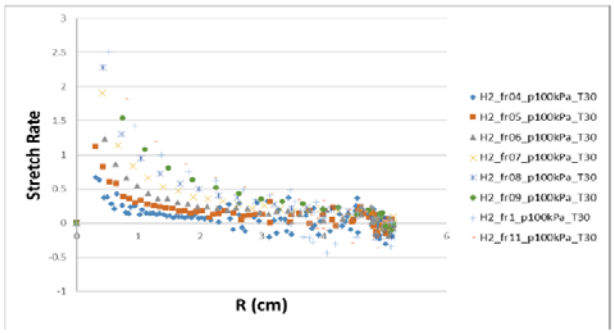
[1]MYER Group. University of Valladolid (Spain)
(*) tinaut@eii.uva.es

March 2018

Stretch effect investigation

K = Stretch Rate (1/s)
S_n = Stretched Flame Propagation Velocity

$$K = \frac{1}{A_f}\frac{dA_f}{dt} = \frac{2}{R_f}\frac{dR_f}{dt} = \frac{2}{r_f}S_n$$



22