Contents

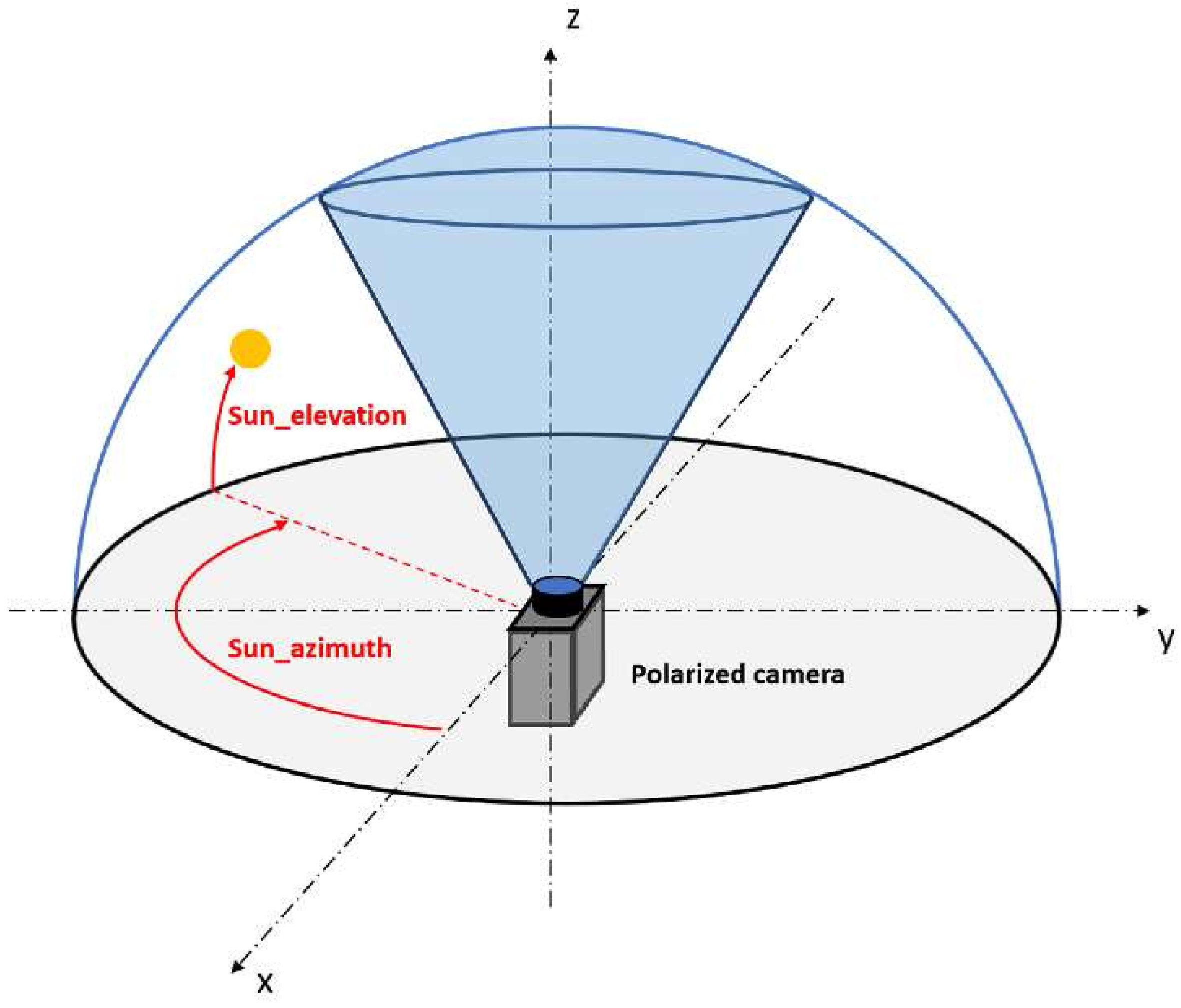
- SETUP
- OPTICS SIMULATION INPUTS
- OPTICS SIMULATION
- PARAMETERS FOR FUTURE RESULTS' PRINTS
- TILTED FIELD INPUTS
- TILTED FIELD COMPUTATION
- RAYLEIGHT'S MODEL INPUTS
- RAYLEIGHT'S MODEL SIMULATION
- PRINT RAYLEIGH'S MODEL SIMULATION RESULTS
- BERRY'S MODEL INPUTS
- BERRY'S MODEL SIMULATION
- PRINT BERRY'S MODEL SIMULATION RESULTS
- SKY RADIANCE SIMULATION INPUTS
- SKY RADIANCE SIMULATION
- PRINT SKY RADIANCE SIMULATION RESULTS
- MICRO POLARIZER SIMULATION INPUTS
- MICRO POLARIZER SIMULATION
- PRINT MICRO POLARIZER SIMULATION RESULTS
- SENSOR'S SIMULATION INPUTS
- SENSOR'S SIMULATION
- PRINT SENSOR SIMULATION RESULTS
- DATA PROCESSING
- COMPARISON WITH OUTDOOR REAL CAMERA CAPTURE

```
clear all;
close all;
```

SETUP

```
figure()
imshow('Setup.png')
title('Simulator s canonical setup');
```

Simulator s canonical setup



OPTICS SIMULATION INPUTS

```
% here we compute the visual field of our sensor, for a camera pointing
% toward true vertical axis (up, zenith axis)

% "particules_azimuth_matrix_deg" and "particules_elevation_matrix_deg" are
% the coordinate in degree, of the atmosphere particles conjugated with the
% sensor's pixels.
% azimuth is expressed in x,y camera image plane, seen from above.
% xyz is direct frame, z is up true vertical axis.
% seen from outputed camera image, y axis goes up on the image, x axis goes
% left on the image


%pixel size in micrometers
pixel_size=3.45;
%focal length in millimeters
f=8;
%sensor high in pixels
sensor_rows=2048;
%sensor length in pixels
sensor_cols=2448;

%conjugation type
r_conj='r0';
```

OPTICS SIMULATION

```
[particules_azimuth_matrix_rad,particules_elevation_matrix_rad,X_coordinate_pixels,Y_coordinate_pixels]...
    = Simu_Optical_Conjugation( pixel_size, sensor_cols, sensor_rows, f, r_conj,0,0);
```

PARAMETERS FOR FUTURE RESULTS' PRINTS

```
x_pixel_mesh=ones(size(Y_coordinate_pixels))'*X_coordinate_pixels;
y_pixel_mesh=(Y_coordinate_pixels')*ones(size(X_coordinate_pixels));
```

TILTED FIELD INPUTS

```
% here we compute

rotation_axis_aziluth_deg=0;

rotation_angle_deg=0;

rotation_axis_aziluth_rad=(pi/180)*rotation_axis_aziluth_deg;
rotation_angle_rad=(pi/180)*rotation_angle_deg;
```

TILTED FIELD COMPUTATION

```
[ particules_azimuth_matrix_rad , particules_elevation_matrix_rad ] = Zenital_tilt( particules_azimuth_matrix_rad, ...
    particules_elevation_matrix_rad, rotation_axis_aziluth_rad, rotation_angle_rad);
```

RAYLEIGHT'S MODEL INPUTS

```
% sun azimuth is expressed in x,y camera image plane, seen from above.
% xyz is direct frame, z is up true vertical axis.
% seen from outputed camera image, y axis goes up on the image, x axis goes
% left on the image
sun_azimuth_deg = 84;

% sun elevation, in degree, from the ground (0) to the vertical z axis (or
% zenith (90)

sun_elevation_deg = 0;%10.47;

% maximum DoLP observable in sky, 0.75 for usual clear sky
DoLP_Max=0.3;
```

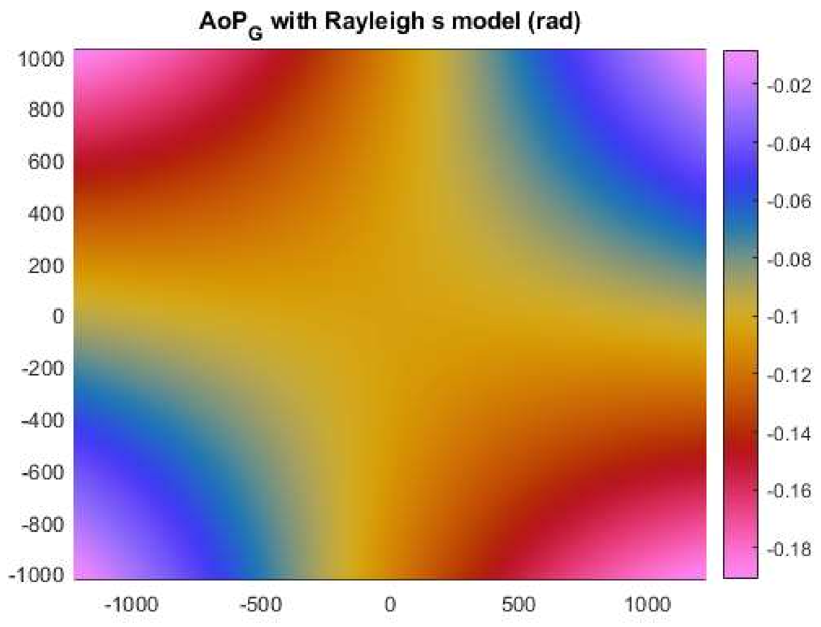
RAYLEIGHT'S MODEL SIMULATION

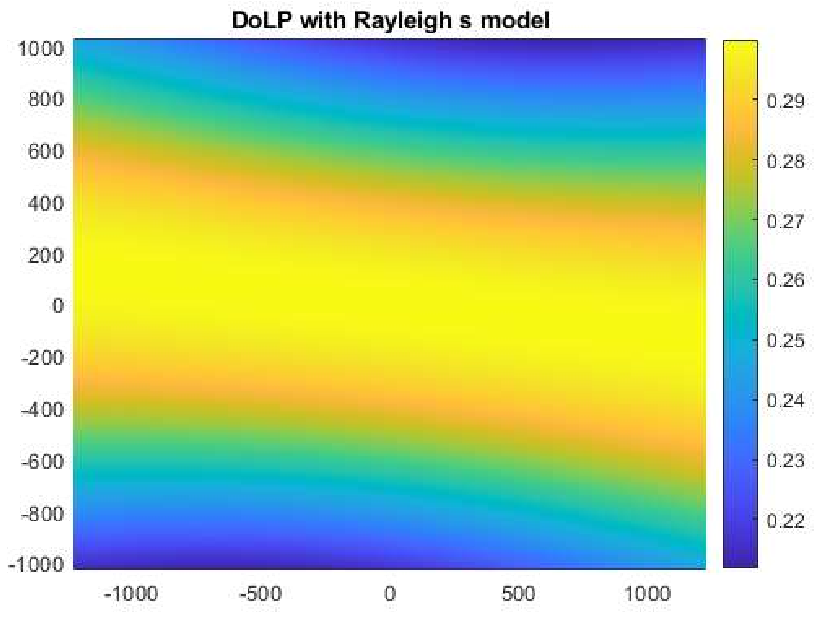
```
sun_elevation_rad = sun_elevation_deg*pi/180;
sun_azimuth_rad = sun_azimuth_deg*pi/180;

[ AoP_Matrix_Global_rad_Rayleigh, DoLP_Matrix_Rayleigh ] = Simu_Rayleigh( sun_elevation_rad,...
sun_azimuth_rad, particules_elevation_matrix_rad, particules_azimuth_matrix_rad, DoLP_Max );
```

PRINT RAYLEIGH'S MODEL SIMULATION RESULTS

```
%Prepare colors for figures
map = cmap('C1');

figure();
h=pcolor(x_pixel_mesh,y_pixel_mesh,AoP_Matrix_Global_rad_Rayleigh);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('AoP_G with Rayleigh s model (rad)');

figure();
```

```
h=pcolor(x_pixel_mesh,y_pixel_mesh,DoLP_Matrix_Rayleigh);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('DoLP with Rayleigh s model');
```





BERRY'S MODEL INPUTS

```
% sun azimuth is expressed in x,y camera image plane, seen from above.
% xyz is direct frame, z is up true vertical axis.
% seen from outputed camera image, y axis goes up on the image, x axis goes
% left on the image

sun_azimuth_deg; %same as Rayleigh's model input in this example

% sun elevation, in degree, from the ground (0) to the vertical z axis (or
% zenith (90)
```

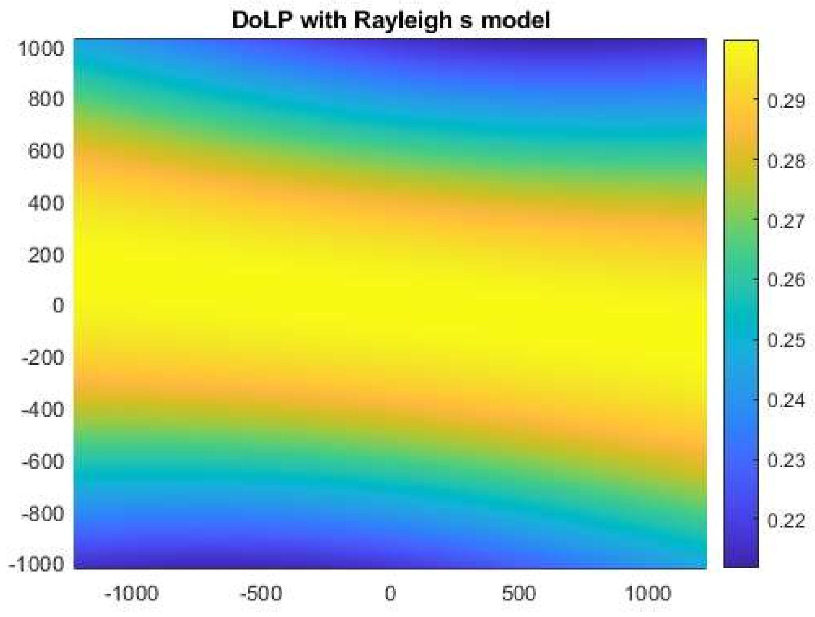
```
sun_elevation_deg; %same as Rayleigh's model input in this example

% maximum DoP observable in sky, 0.75 for usual clear sky
DoLP_Max; %same as Rayleigh's model input in this example

% angle between Babinet and Brewster points in degrees:
delta_deg = 140;
```

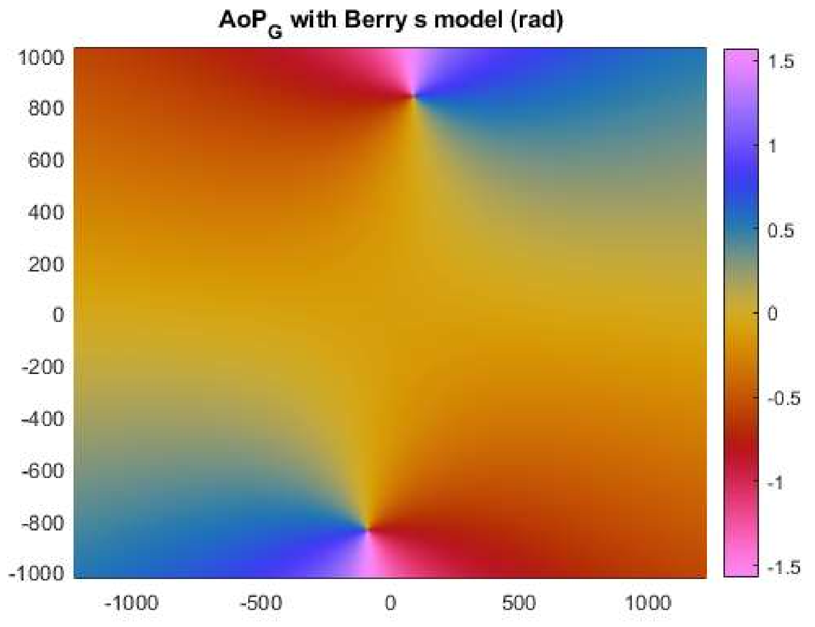
BERRY'S MODEL SIMULATION

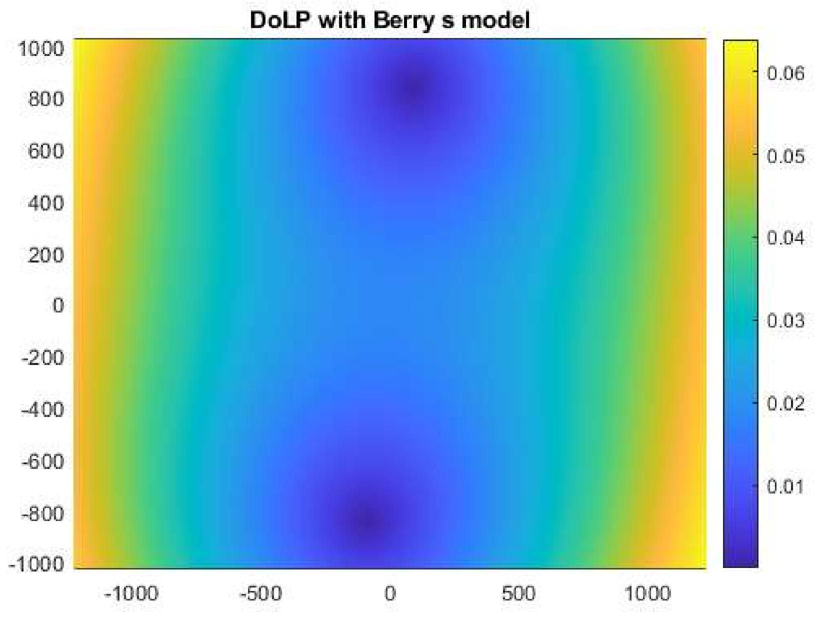
```
sun_elevation_rad = sun_elevation_deg*pi/180;
sun_azimuth_rad = sun_azimuth_deg*pi/180;

delta_rad = delta_deg*pi/180;

[ AoP_Matrix_Global_rad_Berry, DoLP_Matrix_Berry ] = Simu_Berry( sun_elevation_rad,...
sun_azimuth_rad, particules_elevation_matrix_rad, particules_azimuth_matrix_rad, delta_rad,...
DoLP_Max );
```

PRINT BERRY'S MODEL SIMULATION RESULTS

```
%print on pixel map

x_pixel_mesh=ones(size(Y_coordinate_pixels))'*X_coordinate_pixels;
y_pixel_mesh=(Y_coordinate_pixels')*ones(size(X_coordinate_pixels));

%Prepare color for figures
map = cmap('C1');

figure();
h=pcolor(x_pixel_mesh,y_pixel_mesh,AoP_Matrix_Global_rad_Berry);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('AoP_G with Berry s model (rad)');

figure();
h=pcolor(x_pixel_mesh,y_pixel_mesh,DoLP_Matrix_Berry);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('DoLP with Berry s model');
```


DoLP with Rayleigh s model
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
-1000
-500
0
500
1000
0.29
0.28
0.27
0.26
0.25
0.24
0.23
0.22


AoP$_G$ with Berry s model (rad)
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
-1000
-500
0
500
1000
1.5
1
0.5
0
-0.5
-1
-1.5



SKY RADIANCE SIMULATION INPUTS

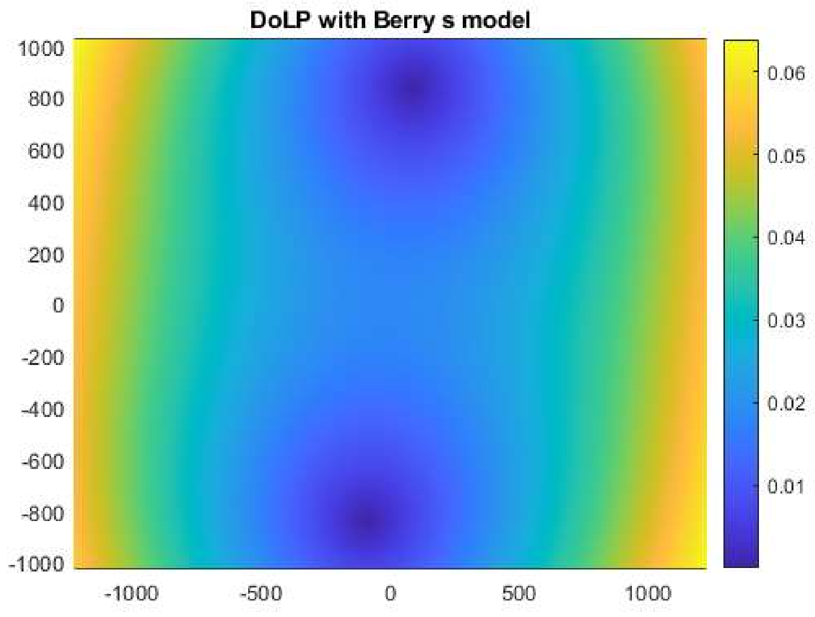
```
% sun azimuth is expressed in x,y camera image plane, seen from above.
% xyz is direct frame, z is up true vertical axis.
% seen from outputed camera image, y axis goes up on the image, x axis goes
% left on the image
sun_azimuth_deg; %same as Rayleigh's model input in this example

% sun elevation, in degree, from the ground (0) to the vertical z axis (or
% zenith (90)

sun_elevation_deg; %same as Rayleigh's model input in this example

% CIE table sky model number, refere to the Article
%"Analysis of vertical sky components under various CIE standard
% general skies" from D.H.W. Li, C. Li, S.W. Lou, E.K.W. Tsang and J.C. Lam
% wrote in 2015
CIE_Sky_number = 9; %integer between 1 and 15
```

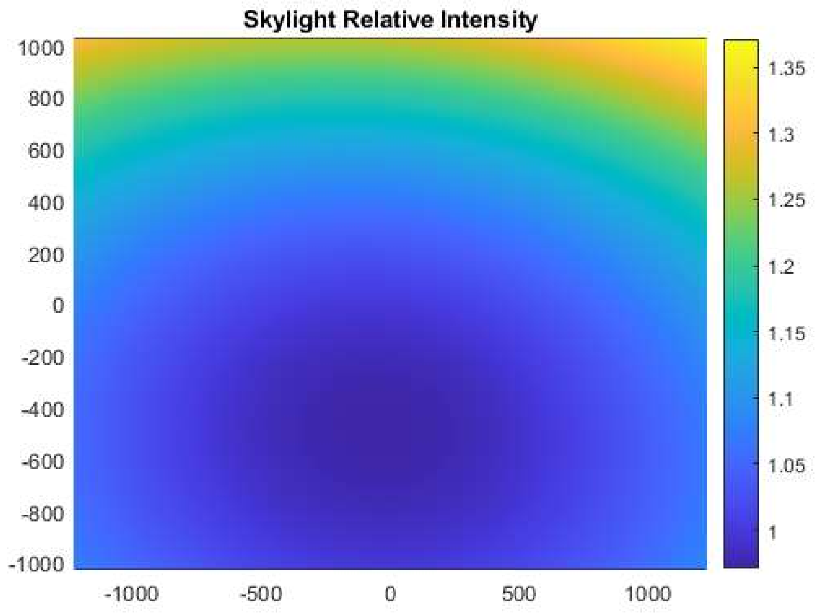
SKY RADIANCE SIMULATION

```
[ Skylight_Relative_Intensity ] = Simu_Sky_Intensity_CIE( sun_azimuth_rad,...
    sun_elevation_rad, particules_azimuth_matrix_rad,...
    particules_elevation_matrix_rad, CIE_Sky_number );
```

PRINT SKY RADIANCE SIMULATION RESULTS

```
figure();
h=pcolor(x_pixel_mesh,y_pixel_mesh,Skylight_Relative_Intensity);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Skylight Relative Intensity');
```





MICRO POLARIZER SIMULATION INPUTS

```
% The sensor called in this simulation  is based on sony IMX250MZR sensor
% used in FLIR BFS-U3-51S5P-C camera.
% It is a sensor with a micro-polarizer array, each block of 2 by 2 pixels
% possess 4 type of linear micro-polarizer oriented at 0,45,90 and 135
% degrees.

% Here we compute micro-polarizer array effects

% we already set size in OPITCS SIMULATION INPUTS :
sensor_rows;
sensor_cols;

% each polarizer is not realy at its supposed orientation, here we put the
% mechanical tolerance in polarizer orientation in degrees
tolerance_deg=1;
```

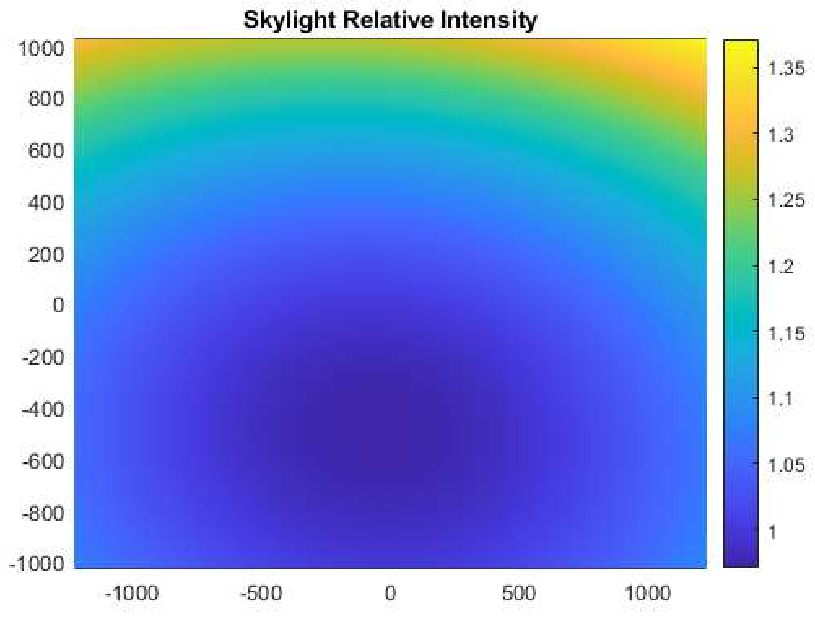
```
% If T1 is the intensity transmitance for an incident ray totaly linearly
% polarized along transmission axis and T2 is the intensity transmittance
% for an incident ray totally linearly polarized at 90 degrees from
% transmission axis, then the polarizer efficiency is (T1-T2)/(T1+T2).

polarizer_efficiency=0.99;
```

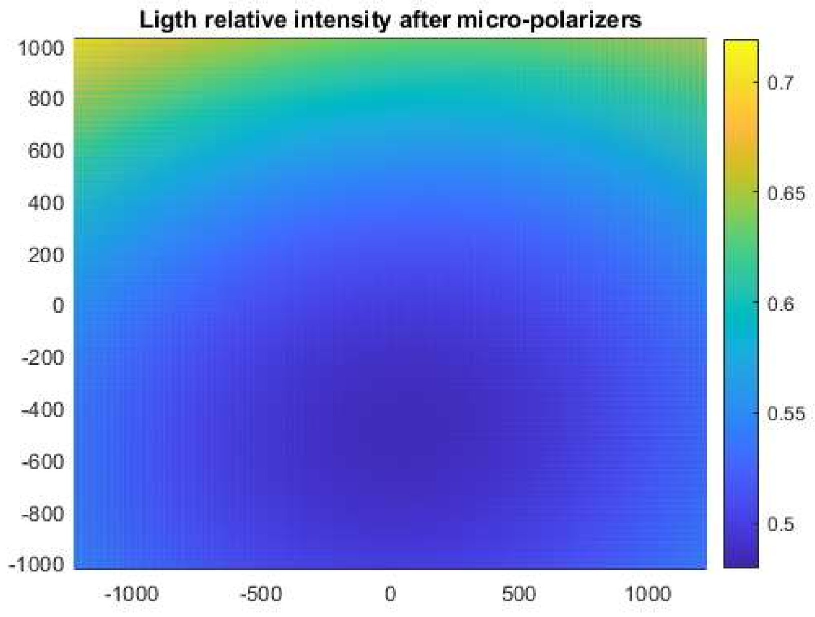
MICRO POLARIZER SIMULATION

```
tolerance_rad=tolerance_deg*pi/180;

% We chose to use results from Berry's model for this example:
Intensity_on_pixels=Simu_Micro_Polarizers(Skylight_Relative_Intensity,AoP_Matrix_Global_rad_Berry,...
    DoLP_Matrix_Berry,tolerance_rad,polarizer_efficiency);
```

PRINT MICRO POLARIZER SIMULATION RESULTS

```
%here the figure is set as returned image
figure();
h=pcolor(-x_pixel_mesh,y_pixel_mesh,Intensity_on_pixels);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Ligth relative intensity after micro-polarizers');
```





SENSOR'S SIMULATION INPUTS

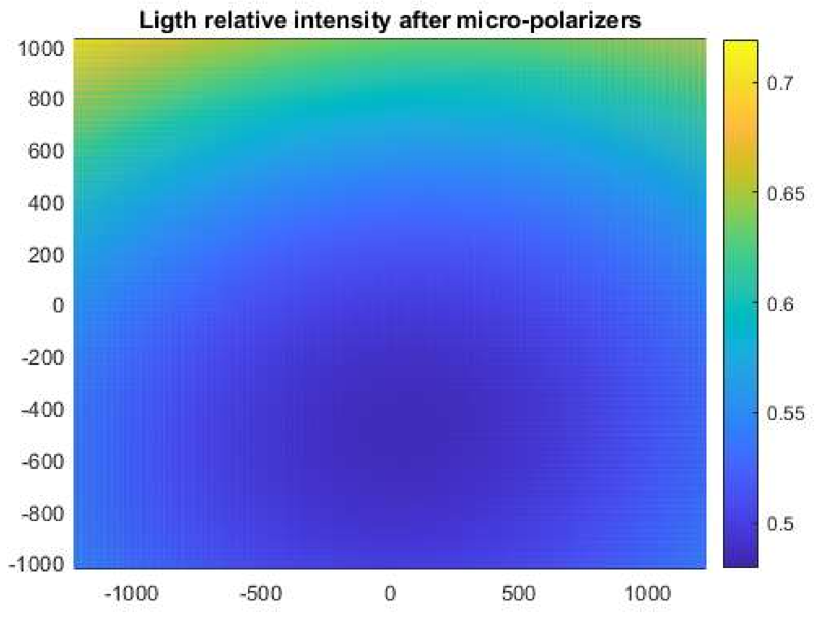
```
% This part deals with pixels simulation

% Here we enter the saturation ratio
% It is the ratio between the sensor irradiance saturation value and the
% maximum relative irradiance comming on sensor's pixels.
pixel_saturation = 1.6;% <1 if saturated, >1 if not saturated

% Here we enter the number of output bits for grayscale
number_of_bits=12;

% Here we enter the per pixel gaussian noise Signal to Noise Ratio
SNR=50;    %no units, no log scale
```

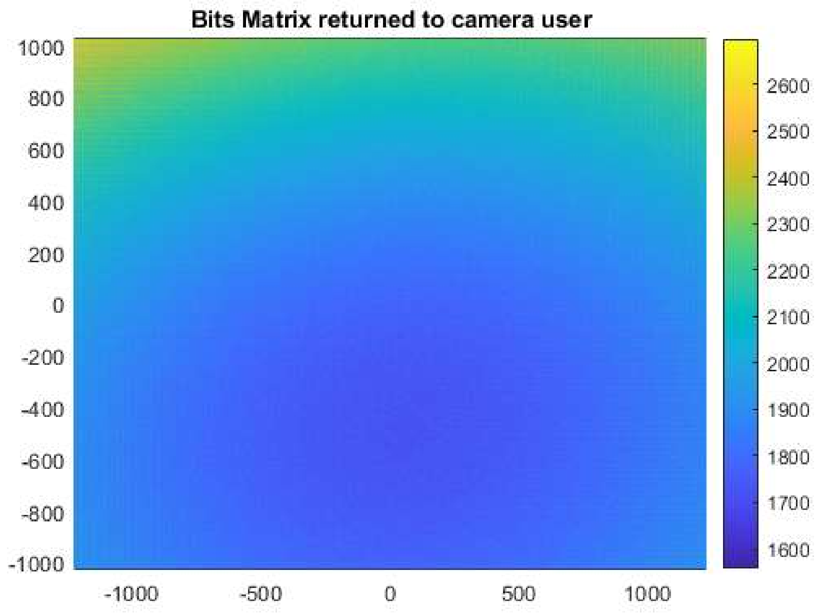
SENSOR'S SIMULATION

```
[Bits_Matrix] = Simu_Sensor( Intensity_on_pixels,...
    pixel_saturation,number_of_bits, SNR);
```

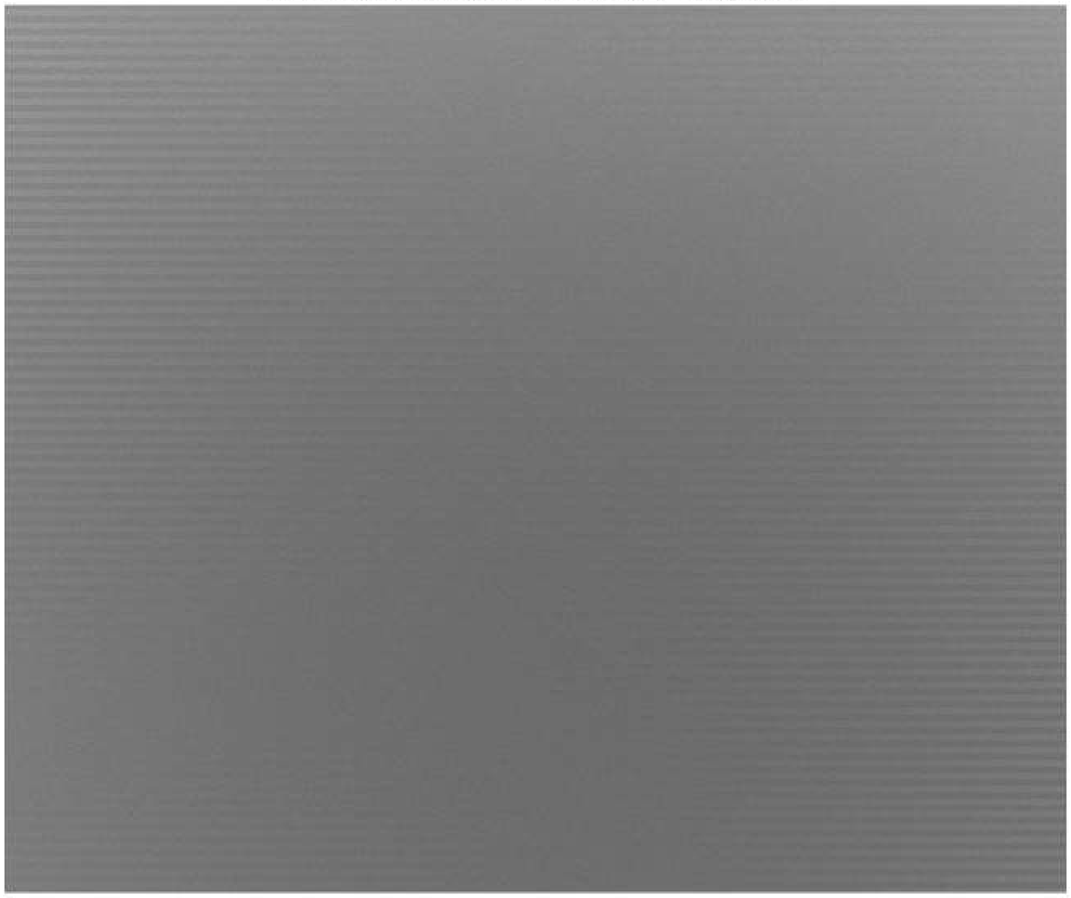
PRINT SENSOR SIMULATION RESULTS

```
%here the figure is set as returned image
figure();
h=pcolor(-x_pixel_mesh,y_pixel_mesh,Bits_Matrix);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Bits Matrix returned to camera user');

%transform grayscale image in 8 bits grayscale image
Bits_Matrix_8B = uint8(floor(255*(1/(2^number_of_bits-1))*double(Bits_Matrix)));
%here the figure is set as returned image
figure();
imshow(Bits_Matrix_8B)
title('Bits Matrix returned to camera user - grayscale');
```


Ligth relative intensity after micro-polarizers
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
-1000
-500
0
500
1000
0.7
0.65
0.6
0.55
0.5


Bits Matrix returned to camera user
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
-1000
-500
0
500
1000
2600
2500
2400
2300
2200
2100
2000
1900
1800
1700
1600

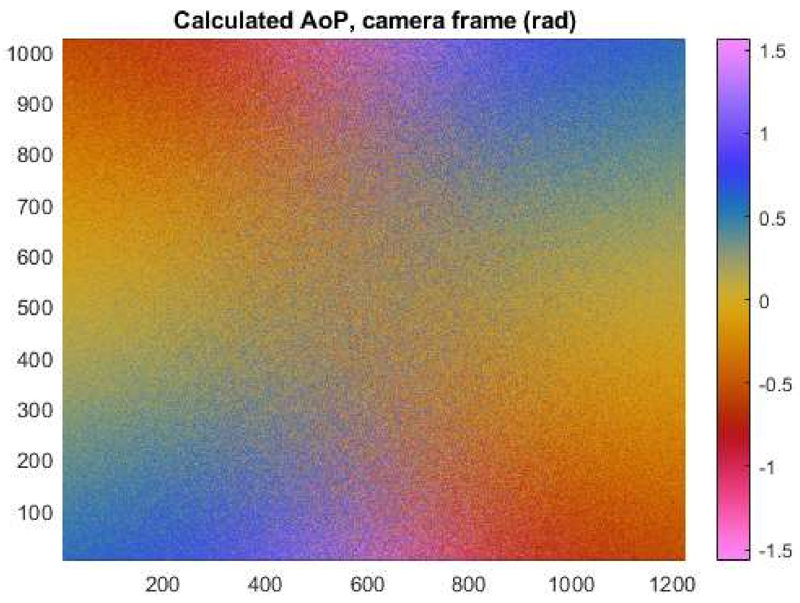
Bits Matrix returned to camera user - grayscale

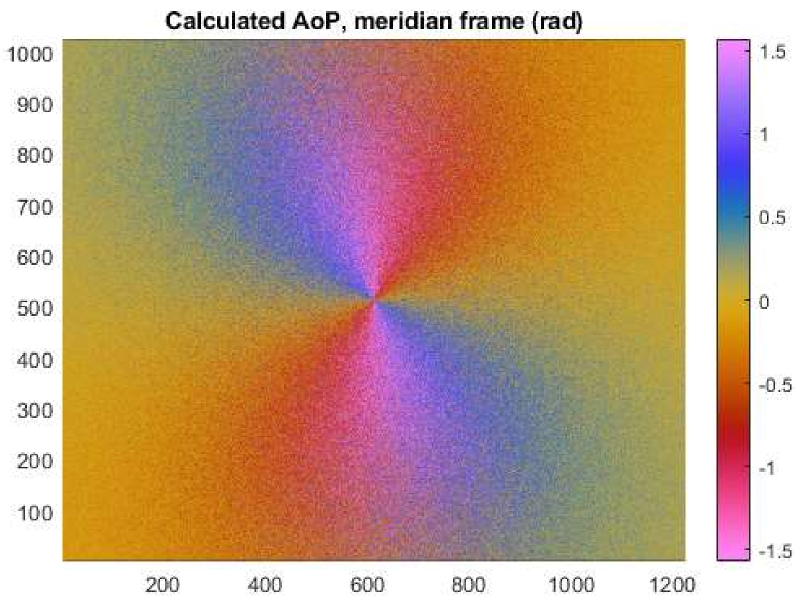


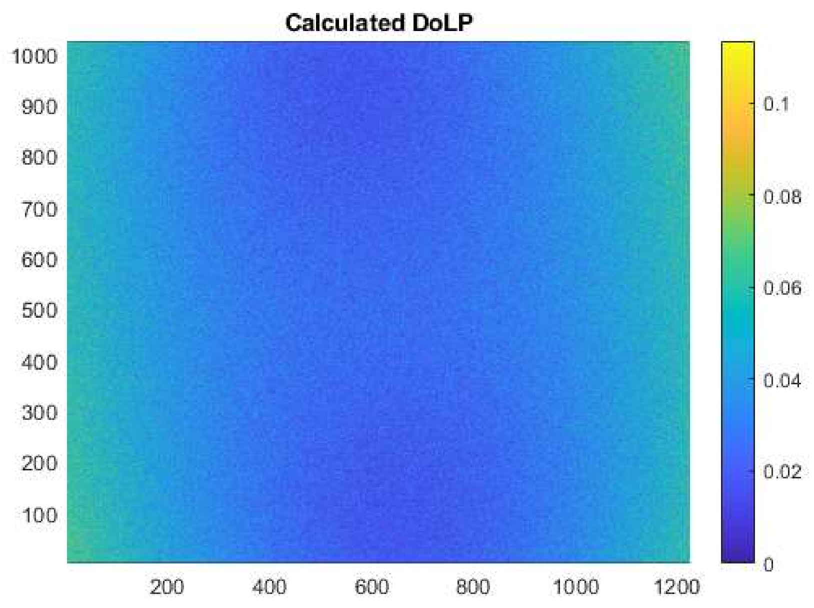
DATA PROCESSING

```
%This part is not a real part of the simulator.
% It is just about to deal with camera output and compute polarization
% measurment, just like we would do with a real camera.

[AoP_data_processing_imframe, AoP_data_processing_meridianframe,...
    DoLP_data_processing] = Simu_Data_Processing(Bits_Matrix);

[rows_print,cols_print]=size(DoLP_data_processing);

X_mesh_print= ones(rows_print,1) * (1:1:cols_print);
Y_mesh_print=(rows_print:-1:1)' * ones(1,cols_print);


map = cmap('C1');

figure();
h=pcolor(X_mesh_print, Y_mesh_print, AoP_data_processing_imframe);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Calculated AoP, camera frame (rad)');

figure();
h=pcolor(X_mesh_print, Y_mesh_print, AoP_data_processing_meridianframe);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Calculated AoP, meridian frame (rad)');
```

```
figure();
h=pcolor(X_mesh_print, Y_mesh_print, DoLP_data_processing);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Calculated DoLP');
```







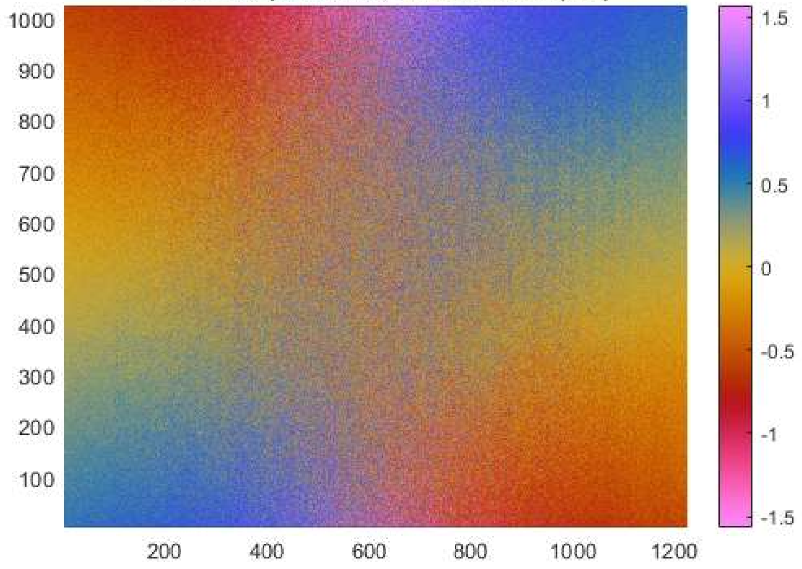
COMPARISON WITH OUTDOOR REAL CAMERA CAPTURE

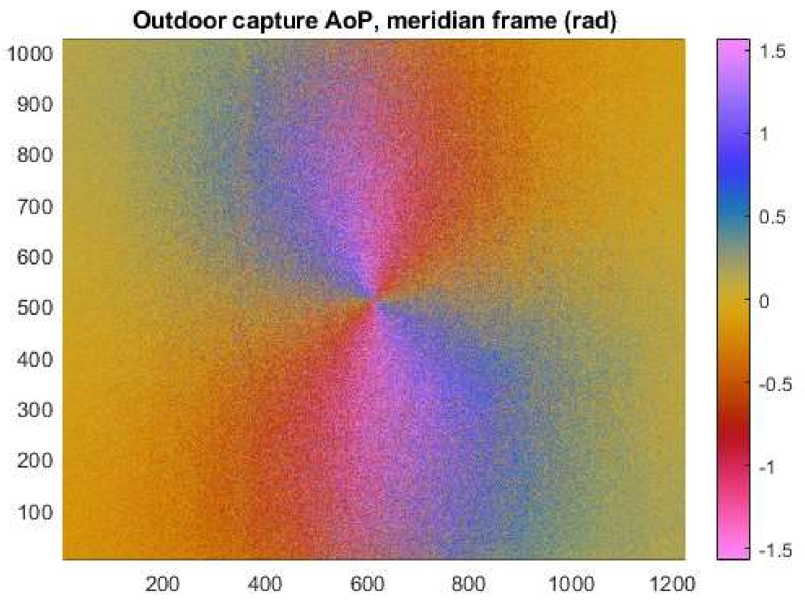
```
%This part is not a real part of the simulator.
%Here we compare our simulation with real outdoor capture
%To do so we use the same data treatment for outdoor output camera data
%than for simulated output camera data

Camera_capture=imread('TEST_overcast_sky.tiff');

Camera_capture_8B=uint8(floor((255/(-1+2^16))*Camera_capture));

figure();
imshow(Camera_capture_8B);
title('Outdoor capture');

MaxCapture=max(max(Camera_capture));

% print outside capture results
Camera_capture_double = double(Camera_capture);
[AoP_expe_imframe, AoP_expe_meridianframe,DoLP_expe] = Simu_Data_Processing(Camera_capture_double);

[rows_print_cam,cols_print_cam]=size(DoLP_expe);
X_mesh_print_cam= ones(rows_print_cam,1) * (1:1:cols_print_cam);
Y_mesh_print_cam=(rows_print_cam:-1:1)' * ones(1,cols_print_cam);


map = cmap('C1');

figure();
h=pcolor(X_mesh_print_cam, Y_mesh_print_cam, AoP_expe_imframe);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Outdoor capture AoP, camera frame (rad)');

figure();
h=pcolor(X_mesh_print_cam, Y_mesh_print_cam, AoP_expe_meridianframe);
colormap(map);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Outdoor capture AoP, meridian frame (rad)');
```

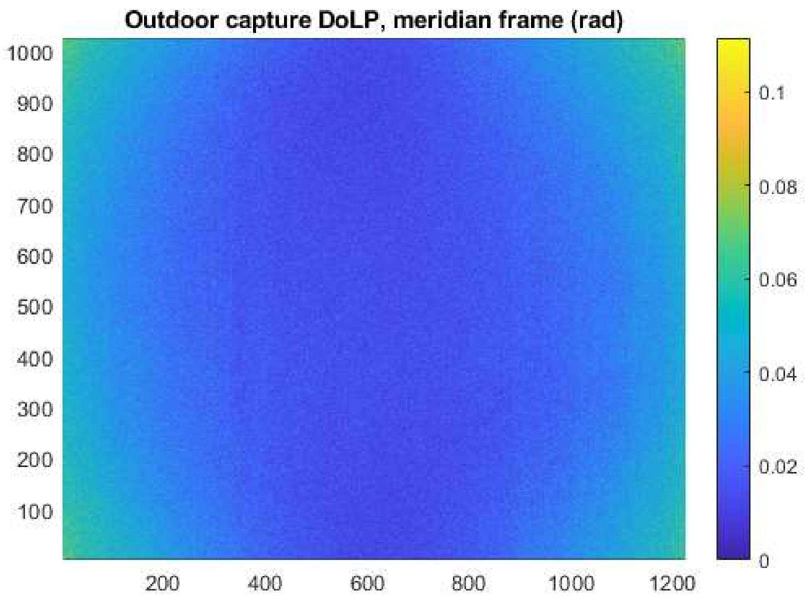
```
figure();
h=pcolor(X_mesh_print_cam, Y_mesh_print_cam, DoLP_expe);
set(h, 'EdgeColor', 'none');
colorbar;
axis image;
title('Outdoor capture DoLP, meridian frame (rad)');
```

Outdoor capture

Outdoor capture AoP, camera frame (rad)







Published with MATLAB® R2022b