**Polarization Angle Computation:**

To compute the polarization angle from a camera with pixels corresponding to polarization orientations of 90 o, 0 o, 45 o, and135 o, we start by computing the Stokes parameters:

S1=I0 - I90

S2 =I45 - I135

Where:

* I0​: Intensity for 0 o polarization.
* I90: Intensity for 90 o polarization.
* I45 ​: Intensity for 45 o polarization.
* I135: Intensity for 135 o polarization.

The polarization angle, θ is then calculated as:

θ = 0.5 \* arctan2(S2, S1)

**Normalize the Polarization Angle:**

Arctan2 (from CUDA math library “math\_function.h”) computes the angle in the correct quadrant, in a range [-π, π]. Since the output image is an 8-bit grayscale, we need to rescale the result before displaying it. The computed θ lies in the range [−π/2,π/2]. To map this range to pixel intensities [0,255] we normalize as follows:

I=(θ+π/2)/π \* 255

* A black pixel corresponds to a polarization angle of −π/2.
* A white pixel corresponds to a polarization angle of π/2.
* Shades of grey represent angles in between.

**FFT Computation:**

Before computing the FFT we need to start from computing the intensity, from the camera pixels which correspond to polarization orientations of 90 o, 0 o, 45 o, and135 o

I = (I0 + I90 + I45 + I135)/2

The function returns a matrix of doubles cropping the imaged to a squared image with side = min(width, length)

The input of the FFT (the intensity matrix) is real, thus cufftExecR2C is used. The output, a matrix (N x N/2+1), is instead complex. To display the result, the magnitude is computed and the output rescale so that the highest value pixel (representing the DC contribution) does not mask other frequencies.

**Results Interpretation:**

Polarization angle: To demonstrate the correct functioning of this section of the code we used two images taken by a polarized camera:

1. **First Image**:
   * The camera is straight, with the laptop's screen polarized at 90 o and the second screen at 0 o.
   * The laptop appears either black (−π/2) or white (π/2).
   * The second screen appears grey (0).
2. **Second Image**:
   * The camera is rotated by 45 o.
   * The laptop's light now results in a π/4 polarization, mapped to light grey.
   * The second screen's light results in a polarization of −π/4 mapped to dark grey.

FFT: To demonstrate the correct functioning of this section of the code we also added a simple image of a circle and one of straight lines. Then we crop the two images of the screens in different format to verify that the cropping mechanism in the function was working correctly and weather the computation was faster with images with dimensions equal to a power of 2. Here the output window:

