```
# Check GPU
!nvidia-smi

```

```
Tue May 20 19:37:55 2025
+----------------------------------------------------------------------------
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Ver
|-----------------------------------------+------------------------+--------
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volati
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Ut
|                                         |                        |
|=========================================+========================+========
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |
| N/A   48C    P0             27W /   70W |     102MiB /  15360MiB |      0
|                                         |                        |
+-----------------------------------------+------------------------+--------

+----------------------------------------------------------------------------
| Processes:
|  GPU   GI   CI        PID   Type   Process name
|        ID   ID
|============================================================================
+----------------------------------------------------------------------------
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files 6 files

dog54.jpeg(image/jpeg) - 19075 bytes, last modified: n/a - 100% done
dog53.jpeg(image/jpeg) - 14693 bytes, last modified: n/a - 100% done
dog52.jpeg(image/jpeg) - 18096 bytes, last modified: n/a - 100% done
dog51.jpeg(image/jpeg) - 17665 bytes, last modified: n/a - 100% done
dog50.jpeg(image/jpeg) - 16623 bytes, last modified: n/a - 100% done
dog49.jpeg(image/jpeg) - 18057 bytes, last modified: n/a - 100% done

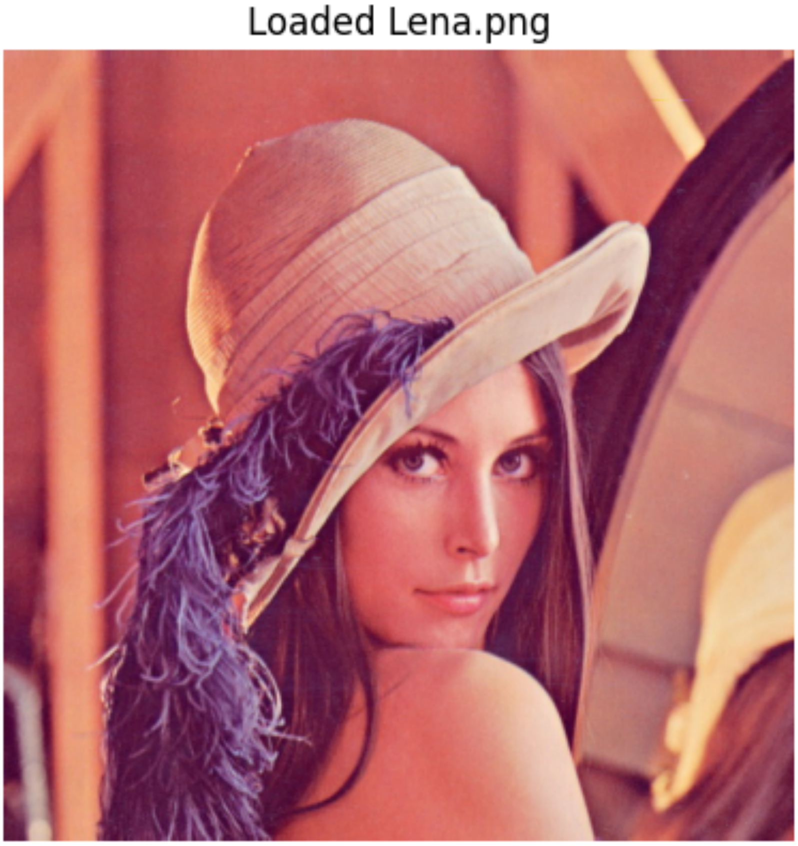
```
Saving dog54.jpeg to dog54 (2).jpeg
Saving dog53.jpeg to dog53 (2).jpeg
Saving dog52.jpeg to dog52 (2).jpeg
Saving dog51.jpeg to dog51 (2).jpeg
Saving dog50.jpeg to dog50 (2).jpeg
Saving dog49.jpeg to dog49 (2).jpeg
```

```
import os

# List all files to confirm it's there
print(os.listdir())

```

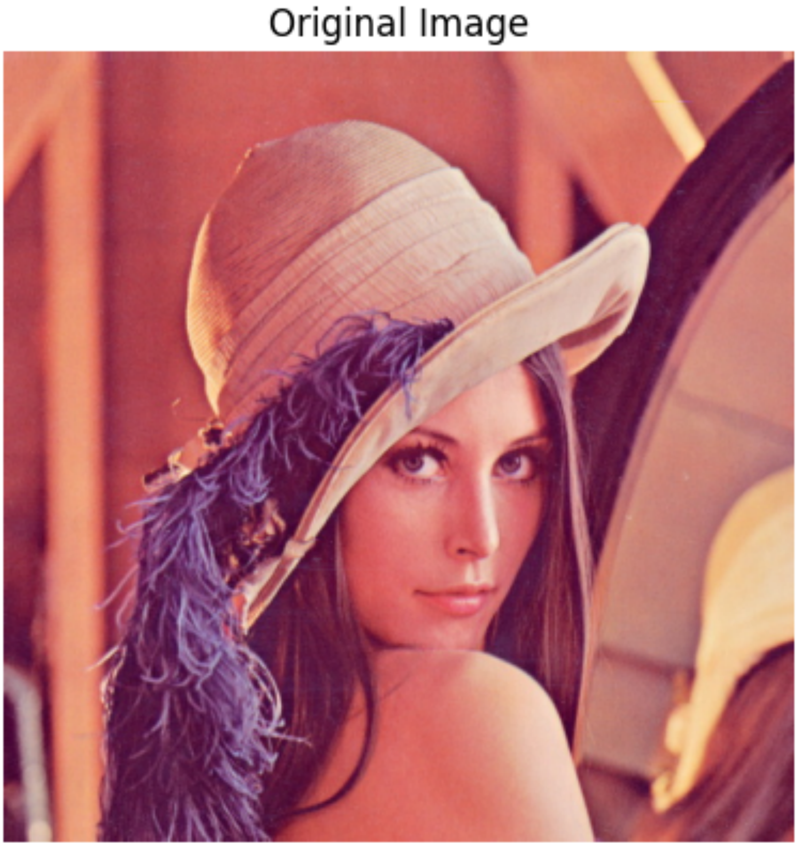
```
['.config', 'dog52.jpeg', 'rotate.cu', 'dog50.jpeg', 'dog36.jpeg', 'dog50 (
# Download Lena image from OpenCV GitHub
!wget https://raw.githubusercontent.com/opencv/opencv/master/samples/data/l

```

```
--2025-05-20 19:40:43--  https://raw.githubusercontent.com/opencv/opencv/ma
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199
HTTP request sent, awaiting response... 200 OK
Length: 91814 (90K) [image/jpeg]
Saving to: ‘Lena.png’

Lena.png            100%[===================>]  89.66K  --.-KB/s    in 0.01

2025-05-20 19:40:44 (8.99 MB/s) - ‘Lena.png’ saved [91814/91814]
```

```python
import cv2
from matplotlib import pyplot as plt

# Load the image
img = cv2.imread("Lena.png")

# Check if image is read correctly
if img is None:
    print("Failed to load image.")
else:
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.title("Loaded Lena.png")
    plt.axis('off')
    plt.show()

```



```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Read the uploaded image
img = cv2.imread("Lena.png")

# Check if the image was loaded successfully
if img is None:
    print("Error: Could not load image 'Lena.png'. Please ensure the file i
else:
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.title("Original Image")
    plt.axis('off')
    plt.show()
```



```
%%writefile rotate.cu
#include <cuda_runtime.h>

__global__ void rotateKernel(unsigned char* input, unsigned char* output, i
    int x = blockIdx.x * blockDim.x + threadIdx.x; // column
    int y = blockIdx.y * blockDim.y + threadIdx.y; // row

    float radians = angle * 3.14159265 / 180.0;
    int xc = width / 2;
    int yc = height / 2;

    if (x < width && y < height) {
        int tx = x - xc;
        int ty = y - yc;

        int srcX = cos(radians) * tx + sin(radians) * ty + xc;
        int srcY = -sin(radians) * tx + cos(radians) * ty + yc;

        for (int c = 0; c < 3; c++) {
            if (srcX >= 0 && srcX < width && srcY >= 0 && srcY < height) {
                output[(y * width + x) * 3 + c] = input[(srcY * width + src
            } else {
                output[(y * width + x) * 3 + c] = 255; // white background
            }
        }
    }
}

```

```
Overwriting rotate.cu
```

```
!nvcc -o rotate rotate.cu

```

```
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/11/../../../x86_64-linux-gnu/Scr
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

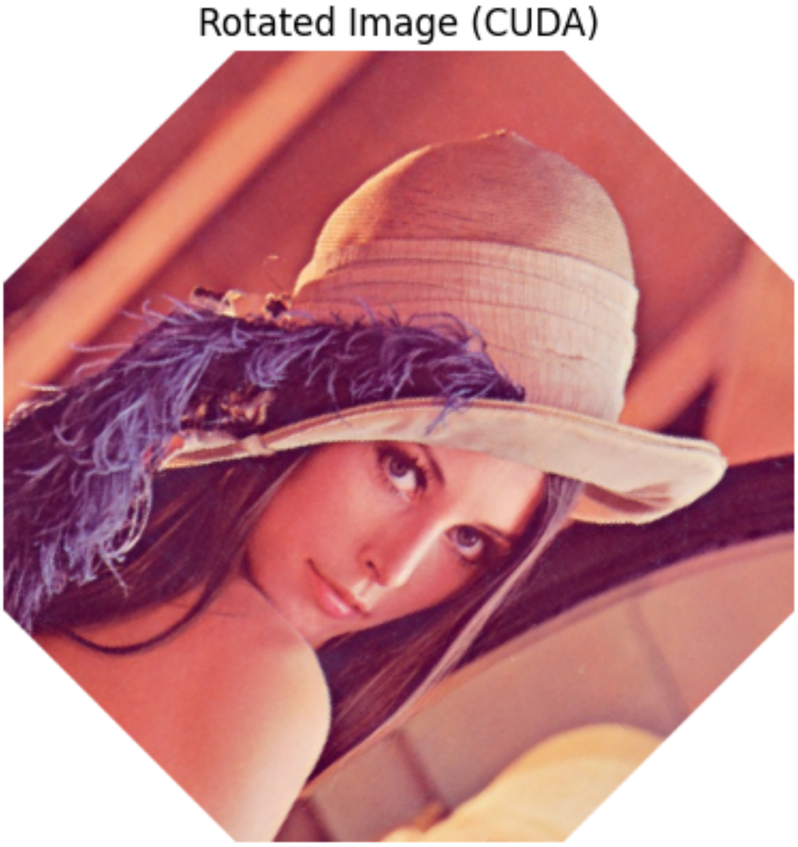
```
!pip install pycuda
```

```
Collecting pycuda
  Downloading pycuda-2025.1.tar.gz (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 39.5 MB/s eta 0:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pytools>=2011.2 (from pycuda)
  Downloading pytools-2025.1.5-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: platformdirs>=2.2.0 in /usr/local/lib/python
Requirement already satisfied: mako in /usr/lib/python3/dist-packages (from
Collecting siphash24>=1.6 (from pytools>=2011.2->pycuda)
  Downloading siphash24-1.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014
Requirement already satisfied: typing-extensions>=4.5 in /usr/local/lib/pyt
Downloading pytools-2025.1.5-py3-none-any.whl (93 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 93.6/93.6 kB 9.6 MB/s eta 0:00:
Downloading siphash24-1.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.6/105.6 kB 10.3 MB/s eta 0:
Building wheels for collected packages: pycuda
5
77
6
9
22
77
88
99
99
7
  Building wheel for pycuda (pyproject.toml) ... done
  Created wheel for pycuda: filename=pycuda-2025.1-cp311-cp311-linux_x86_64
  Stored in directory: /root/.cache/pip/wheels/77/7e/6c/d2d1451ea6424cdc3d6
Successfully built pycuda
Installing collected packages: siphash24, pytools, pycuda
Successfully installed pycuda-2025.1 pytools-2025.1.5 siphash24-1.7
```

```
# Install pycuda
!pip install pycuda

import pycuda.driver as cuda
import pycuda.autoinit
import numpy as np
import cv2
from matplotlib import pyplot as plt
from pycuda.compiler import SourceModule

# Image preprocessing
img = cv2.imread("Lena.png")

# Check if the image was loaded successfully before proceeding
if img is None:
    print("Error: Could not load image 'Lena.png'. Please ensure the file is
else:
```

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (512, 512))
height, width, channels = img.shape
angle = 45

# Flatten image to 1D
img_flat = img.astype(np.uint8).ravel()
output_img = np.zeros_like(img_flat)

# Allocate device memory
input_gpu = cuda.mem_alloc(img_flat.nbytes)
output_gpu = cuda.mem_alloc(output_img.nbytes)

cuda.memcpy_htod(input_gpu, img_flat)

# Run the compiled binary - Note: This line seems redundant as you are a
# compiling and running the kernel through pycuda later. You might want
# remove this unless it serves a specific purpose not evident from the c
# !./rotate

# Define and compile the CUDA kernel
cuda_code = """
#include <cuda_runtime.h>
#include <math.h> // Include math.h for cos and sin

__global__ void rotateKernel(unsigned char* input, unsigned char* output
    int x = blockIdx.x * blockDim.x + threadIdx.x; // column
    int y = blockIdx.y * blockDim.y + threadIdx.y; // row

    float radians = angle * 3.14159265 / 180.0;
    int xc = width / 2;
    int yc = height / 2;

    if (x < width && y < height) {
        int tx = x - xc;
        int ty = y - yc;

        // Ensure floating point calculations for rotation
        float srcX_float = cos(radians) * tx + sin(radians) * ty + xc;
        float srcY_float = -sin(radians) * tx + cos(radians) * ty + yc;

        // Use nearest neighbor interpolation for simplicity
        int srcX = round(srcX_float);
        int srcY = round(srcY_float);

        for (int c = 0; c < 3; c++) {
            if (srcX >= 0 && srcX < width && srcY >= 0 && srcY < height)
                output[(y * width + x) * 3 + c] = input[(srcY * width +
            } else {
                output[(y * width + x) * 3 + c] = 255; // white backgrou
            }
        }
    }
}
"""
mod = SourceModule(cuda_code)
rotateKernel = mod.get_function("rotateKernel")

# Launch kernel
block = (16, 16, 1)
grid = ((width + block[0] - 1) // block[0], (height + block[1] - 1) // b
rotateKernel(input_gpu, output_gpu, np.int32(width), np.int32(height), r

cuda.memcpy_dtoh(output_img, output_gpu)
rotated = output_img.reshape((height, width, 3))

plt.imshow(rotated)
plt.title("Rotated Image (CUDA)")
plt.axis('off')
plt.show()

# Free device memory
input_gpu.free()
output_gpu.free()
```

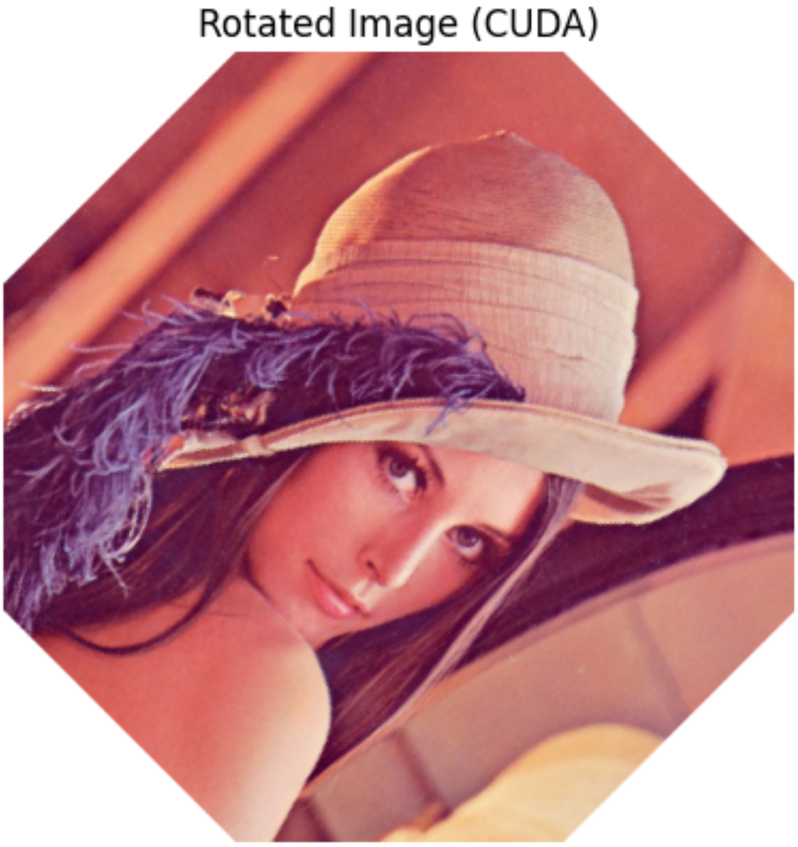
```
Requirement already satisfied: pycuda in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pytools>=2011.2 in /usr/local/lib/python3.11
Requirement already satisfied: platformdirs>=2.2.0 in /usr/local/lib/python
Requirement already satisfied: mako in /usr/lib/python3/dist-packages (from
Requirement already satisfied: siphash24>=1.6 in /usr/local/lib/python3.11/
Requirement already satisfied: typing-extensions>=4.5 in /usr/local/lib/pyt
```



Rotated Image (CUDA)

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt
from pycuda.compiler import SourceModule

# Read the uploaded image
img = cv2.imread("Lena.png")

# Check if the image was loaded successfully
if img is None:
    print("Error: Could not load image 'Lena.png'. Please ensure the file i
else:
    # Rest of your code for processing the image
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (512, 512))
    height, width, channels = img.shape
    angle = 45

    # Flatten image to 1D
    img_flat = img.astype(np.uint8).ravel()
    output_img = np.zeros_like(img_flat)
    # Allocate device memory
    input_gpu = cuda.mem_alloc(img_flat.nbytes)
    output_gpu = cuda.mem_alloc(output_img.nbytes)

    cuda.memcpy_htod(input_gpu, img_flat)

    # Define and compile the CUDA kernel
    cuda_code = """
    #include <cuda_runtime.h>
    #include <math.h> // Include math.h for cos and sin

    __global__ void rotateKernel(unsigned char* input, unsigned char* outpu
        int x = blockIdx.x * blockDim.x + threadIdx.x; // column
        int y = blockIdx.y * blockDim.y + threadIdx.y; // row

        float radians = angle * 3.14159265 / 180.0;
        int xc = width / 2;
        int yc = height / 2;

        if (x < width && y < height) {
            int tx = x - xc;
            int ty = y - yc;

            // Ensure floating point calculations for rotation
            float srcX_float = cos(radians) * tx + sin(radians) * ty + xc;
            float srcY_float = -sin(radians) * tx + cos(radians) * ty + yc;

            // Use nearest neighbor interpolation for simplicity
            int srcX = round(srcX_float);
            int srcY = round(srcY_float);

            for (int c = 0; c < 3; c++) {
                if (srcX >= 0 && srcX < width && srcY >= 0 && srcY < height
                    output[(y * width + x) * 3 + c] = input[(srcY * width +
                } else {
                    output[(y * width + x) * 3 + c] = 255; // white backgr
                }
            }
        }
    }
    """
    mod = SourceModule(cuda_code)
    rotateKernel = mod.get_function("rotateKernel")

    # Launch kernel
    block = (16, 16, 1)
    grid = ((width + block[0] - 1) // block[0], (height + block[1] - 1) //
    rotateKernel(input_gpu, output_gpu, np.int32(width), np.int32(height),

    cuda.memcpy_dtoh(output_img, output_gpu)
    rotated = output_img.reshape((height, width, 3))

    plt.imshow(rotated)
    plt.title("Rotated Image (CUDA)")
    plt.axis('off')
    plt.show()

    # Free device memory
    input_gpu.free()
    output_gpu.free()
```

```
/usr/local/lib/python3.11/dist-packages/google/colab/_variable_inspector.py
  globals().clear()
```



```
Start coding or generate with AI.
```