{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 1000

},

"id": "6N0OXo4THz5b",

"outputId": "cd6fbbcb-d9bc-4574-b210-72535b68843c"

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1200x500 with 2 Axes>"

],

"image/png": "\n"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 800x500 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1200x500 with 2 Axes>"

],

"image/png": "\n"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1200x500 with 2 Axes>"

],

"image/png": "\n"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1000x500 with 3 Axes>"

],

"image/png": "\n"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1200x500 with 2 Axes>"

],

"image/png": "\n"

},

"metadata": {}

}

],

"source": [

"import cv2\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"\n",

"def compute\_histogram(image, title):\n",

" \"\"\"Computes and plots histogram for grayscale image\"\"\"\n",

" hist = cv2.calcHist([image], [0], None, [256], [0, 256])\n",

" hist\_prob = hist / hist.sum() # Probability distribution\n",

"\n",

" plt.figure(figsize=(12, 5))\n",

"\n",

" # Method 1: Histogram with pixel count\n",

" plt.subplot(1, 2, 1)\n",

" plt.plot(hist, color='black')\n",

" plt.title(f'{title} - Histogram (Pixel Count)')\n",

" plt.xlabel('Intensity Values')\n",

" plt.ylabel('Number of Pixels')\n",

"\n",

" # Method 2: Histogram with probability\n",

" plt.subplot(1, 2, 2)\n",

" plt.plot(hist\_prob, color='red')\n",

" plt.title(f'{title} - Histogram (Probability)')\n",

" plt.xlabel('Intensity Values')\n",

" plt.ylabel('Probability')\n",

"\n",

" plt.show()\n",

"\n",

"def process\_color\_image(image):\n",

" \"\"\"Computes histograms for each channel in a color image.\"\"\"\n",

" colors = ('b', 'g', 'r')\n",

" plt.figure(figsize=(8, 5))\n",

"\n",

" for i, color in enumerate(colors):\n",

" hist = cv2.calcHist([image], [i], None, [256], [0, 256])\n",

" plt.plot(hist, color=color, label=f'{color.upper()} Channel')\n",

"\n",

" plt.title('Color Image Histogram')\n",

" plt.xlabel('Intensity Values')\n",

" plt.ylabel('Number of Pixels')\n",

" plt.legend()\n",

" plt.show()\n",

"\n",

"def histogram\_equalization(image):\n",

" \"\"\"Enhances image contrast using histogram equalization.\"\"\"\n",

" equalized = cv2.equalizeHist(image)\n",

" return equalized\n",

"\n",

"def main():\n",

" # Load color image\n",

" image\_color = cv2.imread('/content/Lena\_Image.png') # Replace with your image path\n",

" image\_color = cv2.cvtColor(image\_color, cv2.COLOR\_BGR2RGB)\n",

"\n",

" # Convert to grayscale\n",

" image\_gray = cv2.cvtColor(image\_color, cv2.COLOR\_RGB2GRAY)\n",

"\n",

" # Compute and visualize histograms\n",

" compute\_histogram(image\_gray, 'Grayscale Image')\n",

" process\_color\_image(image\_color)\n",

"\n",

" # Bright and dark image comparison\n",

" bright\_image = cv2.convertScaleAbs(image\_gray, alpha=1.5, beta=50)\n",

" dark\_image = cv2.convertScaleAbs(image\_gray, alpha=0.5, beta=-50)\n",

"\n",

" compute\_histogram(bright\_image, 'Bright Image')\n",

" compute\_histogram(dark\_image, 'Dark Image')\n",

"\n",

" # Histogram Equalization\n",

" equalized\_image = histogram\_equalization(image\_gray)\n",

"\n",

" # Display results\n",

" plt.figure(figsize=(10, 5))\n",

" plt.subplot(1, 3, 1)\n",

" plt.imshow(image\_gray, cmap='gray')\n",

" plt.title('Original Grayscale')\n",

"\n",

" plt.subplot(1, 3, 2)\n",

" plt.imshow(equalized\_image, cmap='gray')\n",

" plt.title('Equalized Image')\n",

"\n",

" plt.subplot(1, 3, 3)\n",

" plt.imshow(image\_color)\n",

" plt.title('Original Color Image')\n",

"\n",

" plt.show()\n",

"\n",

" compute\_histogram(equalized\_image, 'Equalized Image')\n",

"\n",

"if \_\_name\_\_ == \"\_\_main\_\_\":\n",

" main()\n"

]

},

{

"cell\_type": "code",

"source": [

"import cv2\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"\n",

"def compute\_fft(image):\n",

" \"\"\"Computes the FFT and magnitude spectrum of an image.\"\"\"\n",

" dft = np.fft.fft2(image)\n",

" dft\_shift = np.fft.fftshift(dft)\n",

" magnitude\_spectrum = 20 \* np.log(np.abs(dft\_shift) + 1)\n",

"\n",

" return dft, dft\_shift, magnitude\_spectrum\n",

"\n",

"def compute\_ifft(dft):\n",

" \"\"\"Computes the Inverse FFT to reconstruct the original image.\"\"\"\n",

" dft\_shift = np.fft.ifftshift(dft)\n",

" img\_reconstructed = np.fft.ifft2(dft\_shift)\n",

" img\_reconstructed = np.abs(img\_reconstructed)\n",

"\n",

" return img\_reconstructed\n",

"\n",

"def rotate\_image(image, angle):\n",

" \"\"\"Rotates the image by a given angle.\"\"\"\n",

" (h, w) = image.shape[:2]\n",

" center = (w // 2, h // 2)\n",

" M = cv2.getRotationMatrix2D(center, angle, 1.0)\n",

" rotated = cv2.warpAffine(image, M, (w, h))\n",

"\n",

" return rotated\n",

"\n",

"def main():\n",

" # Load grayscale image\n",

" image = cv2.imread('/content/Flower.jpg', cv2.IMREAD\_GRAYSCALE) # Replace with your image path\n",

"\n",

" # Compute FFT and display magnitude spectrum\n",

" dft, dft\_shift, magnitude\_spectrum = compute\_fft(image)\n",

"\n",

" # Compute IFFT to reconstruct image\n",

" reconstructed\_image = compute\_ifft(dft\_shift)\n",

"\n",

" # Rotate image by 45 degrees and compute its FFT\n",

" rotated\_image = rotate\_image(image, 45)\n",

" \_, \_, rotated\_magnitude\_spectrum = compute\_fft(rotated\_image)\n",

"\n",

" # Display images and results\n",

" plt.figure(figsize=(12, 6))\n",

"\n",

" plt.subplot(2, 3, 1)\n",

" plt.imshow(image, cmap='gray')\n",

" plt.title('Original Image')\n",

"\n",

" plt.subplot(2, 3, 2)\n",

" plt.imshow(magnitude\_spectrum, cmap='gray')\n",

" plt.title('Magnitude Spectrum')\n",

"\n",

" plt.subplot(2, 3, 3)\n",

" plt.imshow(reconstructed\_image, cmap='gray')\n",

" plt.title('Reconstructed Image')\n",

"\n",

" plt.subplot(2, 3, 4)\n",

" plt.imshow(rotated\_image, cmap='gray')\n",

" plt.title('Rotated Image (45°)')\n",

"\n",

" plt.subplot(2, 3, 5)\n",

" plt.imshow(rotated\_magnitude\_spectrum, cmap='gray')\n",

" plt.title('Magnitude Spectrum (Rotated)')\n",

"\n",

" plt.show()\n",

"\n",

"if \_\_name\_\_ == \"\_\_main\_\_\":\n",

" main()\n"

],

"metadata": {

"id": "tfTo0\_RnJvXu",

"outputId": "4d6d0a15-24df-4d3c-915d-69b1c09aaaec",

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 517

}

},

"execution\_count": null,

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 1200x600 with 5 Axes>"

],

"image/png": "\n"

},

"metadata": {}

}

]

}

]

}