МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Звіт

до лабораторної роботи №4

з предмету Комп’ютерне бачення та аналіз зображень

Роботу виконала:

Мерцало Ірина Ігорівна,

студентка групи ПМІМ-11

Львів – 2022

Contour detection

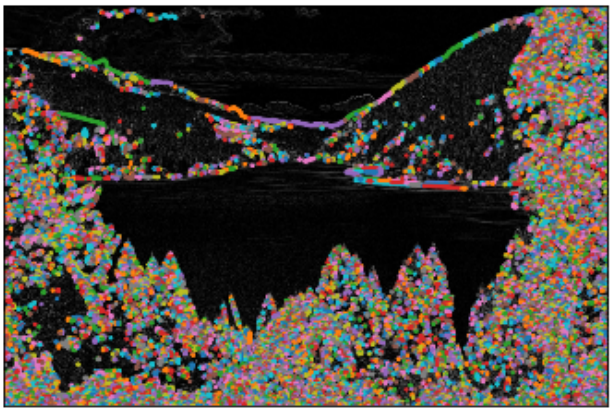
Вихідне зображення:

mountains.jpg

Виконала визначення контурів (обводить межі для кожного об’єкту і формує замкнутий цикл; кожен замкнутий цикл на зображенні являє собою контур). Для цього відкрила зображення, перетворила його у градації сірого, застосувала собелеве визначення країв:

In [2]:
```
from skimage import measure
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.filters import sobel
import matplotlib.pyplot as plt

#Read an image
img = imread('mountains.jpg')
#Convert the image to grayscale
img_gray = rgb2gray(img)
#Find edges in the image
img_edges = sobel(img_gray)
#Find contours in the image
contours = measure.find_contours(img_edges, 0.2)
# Display the image and plot all contours found
fig, ax = plt.subplots()
ax.imshow(img_edges, interpolation='nearest', cmap=plt.cm.gray)
for n, contour in enumerate(contours):
    ax.plot(contour[:, 1], contour[:, 0], linewidth=2)
ax.axis('image')
ax.set_xticks([])
ax.set_yticks([])
plt.show()
```



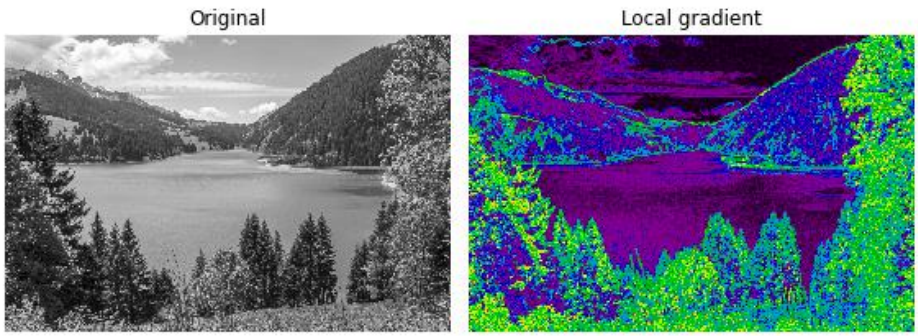
The Watershed algorithm

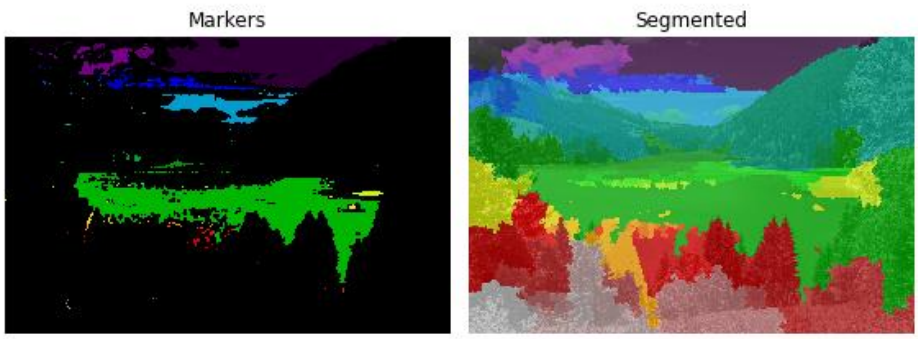
Застосувала вододільний алгоритм (знаходить локальні мінімуми - маркери, які позначають пирблизне рзташування об’єкта, присвоює їм унікальний колір, заповнює цим кольором навколо допоки не дійде до кольору іншого маркера):

In [24]:
```
from scipy import ndimage as ndi
from skimage.morphology import watershed, disk
from skimage.segmentation import watershed
from skimage import data
from skimage.io import imread
from skimage.filters import rank
from skimage.color import rgb2gray
from skimage.util import img_as_ubyte
import matplotlib.pyplot as plt

img = imread('mountains.jpg')
img_gray = rgb2gray(img)
image = img_as_ubyte(img_gray)
#Calculate the local gradients of the image
#and only select the points that have a
#gradient value of less than 20
markers = rank.gradient(image, disk(5)) < 20
markers = ndi.label(markers)[0]
gradient = rank.gradient(image, disk(2))
#Watershed Algorithm
labels = watershed(gradient, markers)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(8,8), sharex=True, sharey=True)
ax = axes.ravel()
ax[0].imshow(image,cmap=plt.cm.gray, interpolation='nearest')
ax[0].set_title("Original")
ax[1].imshow(gradient,cmap=plt.cm.nipy_spectral, interpolation='nearest')
ax[1].set_title("Local gradient")
ax[2].imshow(markers,cmap=plt.cm.nipy_spectral, interpolation='nearest')
ax[2].set_title("Markers")
ax[3].imshow(image,cmap=plt.cm.gray, interpolation='nearest')
ax[3].imshow(labels,cmap=plt.cm.nipy_spectral, interpolation='nearest',alpha=.7)
ax[3].set_title("Segmented")
for a in ax:
    a.axis('off')
fig.tight_layout()
plt.show()
```





Superpixels

Зобразила зображення у вигляді суперпікселів (алгоритм об’єднує сусідні пікселі схожих кольорів у кластери, які отримують назву суперпіксель):

In [19]:
```
from skimage import segmentation, color
from skimage.io import imread, imsave
from skimage.future import graph
from matplotlib import pyplot as plt

img = imread('mountains.jpg')
img_segments = segmentation.slic(img, compactness=20, n_segments=500)
superpixels = color.label2rgb(img_segments, img, kind='avg')

imsave('result.jpg', superpixels)
```

mountains.jpg

result.jpg

Normalized graph cut

Застосувала алгоритм нормалізованого розрізання графа (кожен піксель зображення сприймає як вузол, додатково є вузли які представляють об’єкти на зображенні; усі пікселі пов’язані з усіма сусідніми пікселями і кожен із вузлами об’єкта. Ітераційно зрізає ребра в графі, таким чином отримує підграфи, допоки це можливо. В результаті кожен піксель зображення буде пов’язаний з одним об’єктом):

In [30]:
```
from skimage import data, segmentation, color
from skimage.io import imread
from skimage import data
from skimage.future import graph
from matplotlib import pyplot as plt
img = imread('mountains.jpg')
img_segments = segmentation.slic(img, compactness=30, n_segments=200)
out1 = color.label2rgb(img_segments, img, kind='avg')
segment_graph = graph.rag_mean_color(img, img_segments, mode='similarity')
img_cuts = graph.cut_normalized(img_segments, segment_graph)
normalized_cut_segments = color.label2rgb(img_cuts, img, kind='avg')

imsave('result2.jpg', normalized_cut_segments)
```

mountains.jpg

result2.jpg