CZ4003 Computer Vision Project

Wilson Thurman Teng (U1820540H)

Imports

In [1]:
```
import os

import cv2
import numpy as np
import plotly.graph_objects as go
from matplotlib import pyplot as plt

try:
    from PIL import Image
except ImportError:
    import Image

import pytesseract

np.seterr(divide='ignore', invalid='ignore')

# Tesseract-OCR executable
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Set paths
SAMPLE_1 = os.path.join('images', 'sample01.png')
SAMPLE_1_GROUND_TRUTH = os.path.join('out', 'correct_1.txt')
SAMPLE_1_PREDICTION = os.path.join('out', 'prediction_1.txt')
SAMPLE_1_ORIGINAL_PREDICTION = os.path.join('out', 'original_prediction_1.txt')

SAMPLE_2 = 'images/sample02.png'
SAMPLE_2_GROUND_TRUTH = os.path.join('out', 'correct_2.txt')
SAMPLE_2_PREDICTION = os.path.join('out', 'prediction_2.txt')
SAMPLE_2_ORIGINAL_PREDICTION = os.path.join('out', 'original_prediction_2.txt')
```

Utils

In [2]:
```
DPI = (300, 300)

def indent(string, num_spaces, addTripleQuotes=True):
    if addTripleQuotes:
        string = "'''\n" + string + "\n'''\n"
    return '\n'.join([((' ' * num_spaces) + line) for line in string.split("\n")])

def imshow1(img, title="", figsize=(14,14)):
    plt.figure(figsize=figsize)
    plt.imshow(img, cmap = 'gray')
    plt.title(title);

def imshow2(img, title="", figsize=(10,10)):
    plt.figure(figsize=figsize)
    plt.imshow(img, cmap = 'gray')
    plt.title(title);

def generate_image_dpi(path, dest, dpi=DPI):
    os.makedirs(os.path.dirname(dest), exist_ok=True)
    img = Image.open(path)
    img.save(dest, dpi=dpi)
    return Image.open(dest)

def intensity_hotspot(img, i_low, i_high, figsize=(12,12)):
    img = np.array(img)
    thres = np.ones(img.shape).astype(np.uint8)
    thres[np.where(img >= i_low) and np.where(img <= i_high)] = 0
    plt.figure(figsize=figsize)
    plt.imshow(thres, cmap = 'gray')
    return thres
```

Metrics

In [3]:
```
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def get_jaccard_sim(query, document):
    query = query.split()
    document = document.split()
    intersection = set(query).intersection(set(document))
    union = set(query).union(set(document))
```

```
    print(f"Jaccard errors:\n{sorted(union - intersection)}\n")
    if len(union) == 0:
        print('No similar words')
        return 0
    return len(intersection)/len(union)

def get_cosine_sim(query, document):
    text = [query, document]
    count_vec = CountVectorizer()
    count_vec.fit(text)
    arr = count_vec.transform(text).toarray()
    vector1, vector2 = arr
    return cosine_similarity(np.array([vector1]), np.array([vector2]))[0][0]

def get_levenshtein_sim(query, document):
    size_x = len(query) + 1
    size_y = len(document) + 1
    matrix = np.zeros((size_x, size_y), dtype=int)
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if query[x-1] == document[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1] + 1,
                    matrix[x, y-1] + 1
                )

    distance =  matrix[size_x - 1, size_y - 1]
    return 1 - (distance / (size_x + size_y))
```

In [4]:

```
import string
whitelist = string.ascii_lowercase + string.ascii_uppercase + ":-.,!\\\"\\\' "
print(f"[Whitelist]: {whitelist}")
```

```
[Whitelist]: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ:-.,!\"\'
```

In [5]:
```python
# custom_config = r'-c tessedit_char_blacklist=\©_|[] textord_space_size_is_variable=1 preserve_interword_spaces=1 --psm 3 --oem 3
custom_config = f'-c tessedit_char_whitelist="{whitelist}" --psm 3 --oem 3'

def get_metric_score(img, ground_truth_pth):
    img_str = pytesseract.image_to_string(img, config=custom_config).strip()
    print(f"{'=' * 18} Results {'=' * 18}")
    print(f"\n{indent(img_str, 3)}")
    with open(ground_truth_pth, 'r', encoding="utf8") as f:
        ground_truth = f.read().strip()

        jaccard_sim = get_jaccard_sim(img_str, ground_truth)
        cosine_sim = get_cosine_sim(img_str, ground_truth)
        levenshtein_sim = get_levenshtein_sim(img_str, ground_truth)

        print(f"-> Jaccard Similarity:    \t{jaccard_sim:.5f}")
        print(f"-> Cosine Similarity:     \t{cosine_sim:.5f}")
        print(f"-> Levenshtein Similarity:\t{levenshtein_sim:.5f}")

        avg_score = (jaccard_sim + cosine_sim + levenshtein_sim) / 3
        print(f"\n-> Average Score: {avg_score:.5f}")

    return avg_score, [jaccard_sim, cosine_sim, levenshtein_sim], img_str
```

Processing

In [6]:
```python
def rotate_image(img, angle):
    if angle == 0:
        return img
    image_center = tuple(np.array(img.shape[1::-1]) / 2)
    rot_matrix = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    rotated = cv2.warpAffine(img, rot_matrix, img.shape[1::-1],
                             flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_REPLICATE)
    return rotated

def invert_thres(binary_img):
    return (~(binary_img*255) / 255).astype('uint8')

def contrast_stretching(img):
    img = np.array(img)
    min_ = img.min()
    max_ = img.max()
```

```
    img = np.subtract(img, min_)
    img = np.multiply(img, 255 / (max_ - min_))

    return img.astype(np.uint8)

def thresholding(img, low):
    assert low < 256 and low > -1
#     img = img.convert('L')
    img = np.array(img)
    thres = np.zeros(img.shape).astype(np.uint8)
    thres[np.where(img > low)] = 1
    return thres

def otsu_thresholding(img):
    matrix = np.product(img.shape)
    variance_list = []
    hist, bin_edges = np.histogram(img, bins=256, range=(0,256))

    for temp_intensity in np.arange(1, 255, 1):
        thres = thresholding(img, temp_intensity)

        # Computing Weights, w
        w_a = np.sum(hist[:temp_intensity]) / float(matrix)
        w_b = np.sum(hist[temp_intensity:]) / float(matrix)

        # Computing Variances, sigma
        sigma_a = np.var(img[np.where(thres == 0)])
        sigma_b = np.var(img[np.where(thres == 1)])

        # Appending variance of Li to variance_list
        variance_list.append(w_a * sigma_a + w_b * sigma_b)

    L_optimal = np.nanargmin(variance_list)
    otsu_img = thresholding(img, L_optimal)

    return otsu_img, L_optimal

def get_skew_angle(img):
    otsu, otsu_thres = otsu_thresholding(img)

    coords = np.column_stack(np.where(otsu == 0))
    angle = cv2.minAreaRect(coords)[-1]
```

```
    if angle < -45:
        return -(90 + angle)
    else:
        return -angle
```

Sample01.png

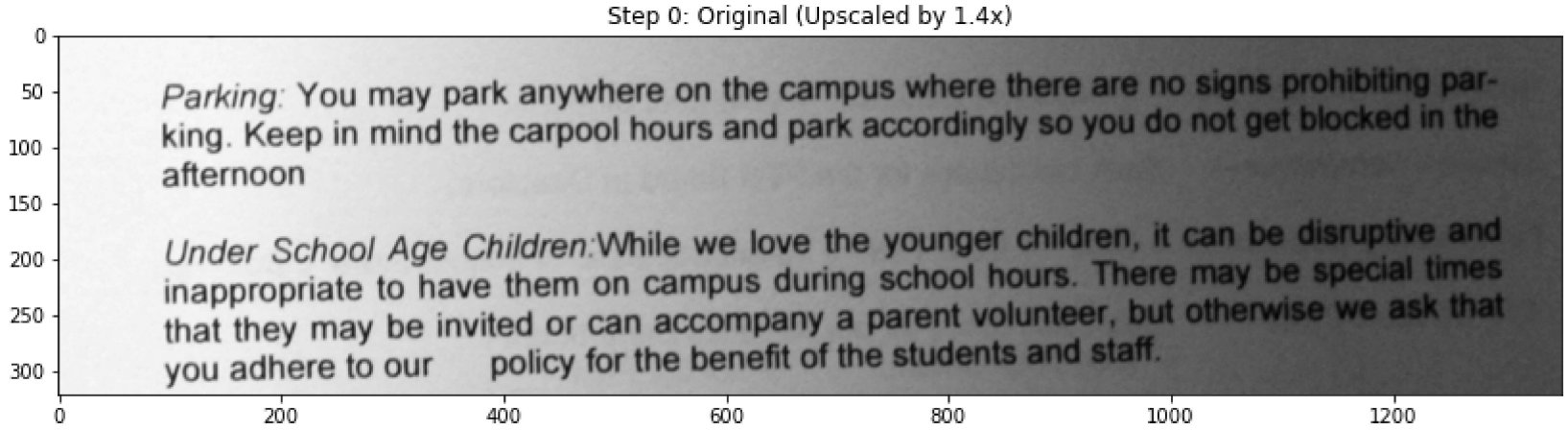
In [7]:
```
# Original image results
_, _, results_str1 = get_metric_score(SAMPLE_1, SAMPLE_1_GROUND_TRUTH);
with open(SAMPLE_1_ORIGINAL_PREDICTION, "w") as text_file:
    text_file.write(results_str1)
```

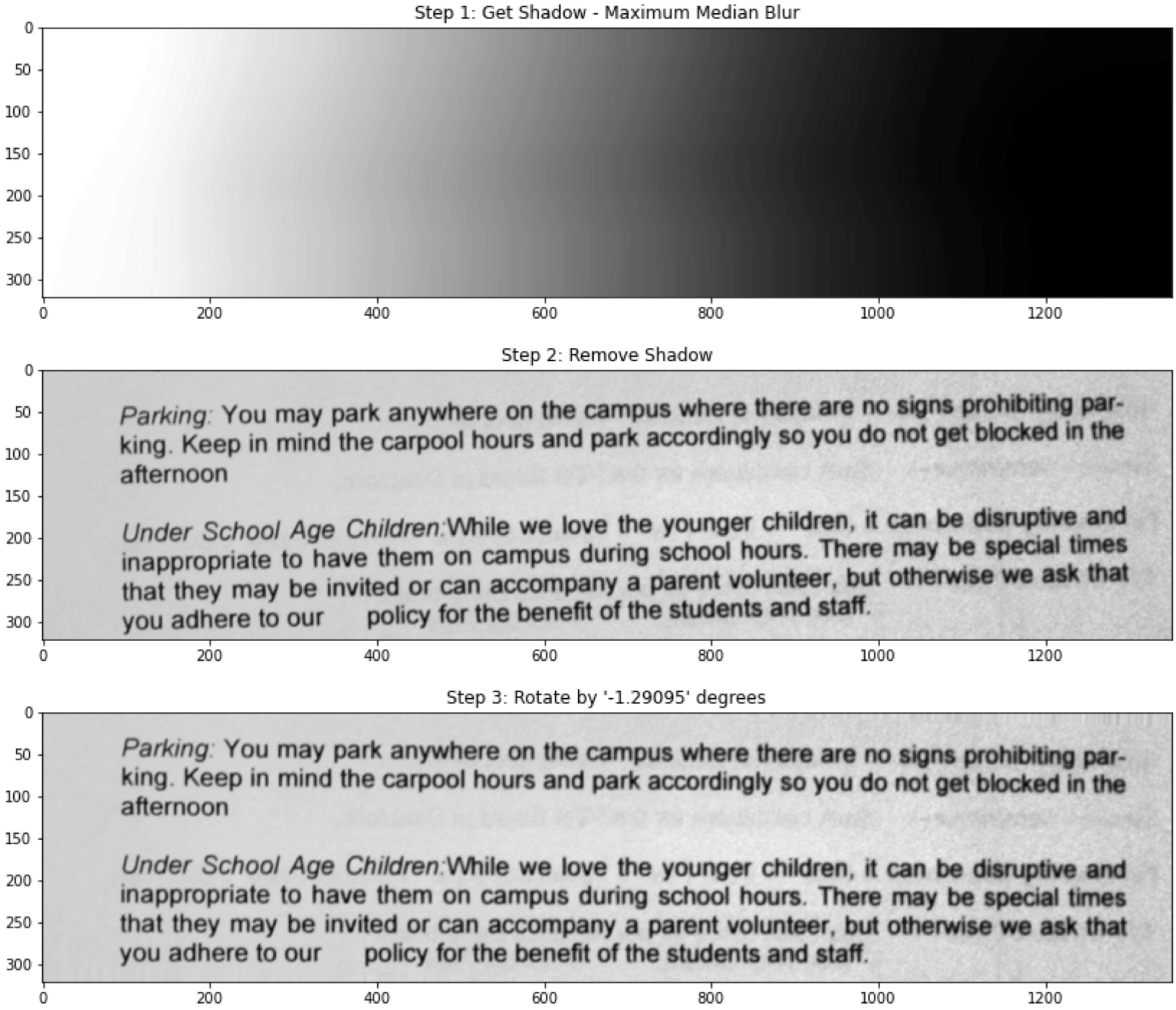
```
================== Results ==================

   '''
   Parking: You may park anywhere on the ce
   king. Keep in mind the carpool hours and park
   afternoon

   Under School Age Children:While we love
   inappropriate to have them on campus d
   that they may be invited or can accompany :
   you adhere to our policy for the benefit of
   '''

Jaccard errors:
[':', 'There', 'a', 'accordingly', 'are', 'ask', 'blocked', 'but', 'ce', 'children,', 'd', 'disruptive', 'do', 'during', 'get', 'h
ours.', 'it', 'no', 'not', 'otherwise', 'par-', 'parent', 'prohibiting', 'school', 'signs', 'so', 'special', 'staff.', 'students',
'there', 'times', 'volunteer,', 'where', 'younger']

-> Jaccard Similarity:          0.54054
-> Cosine Similarity:           0.84615
-> Levenshtein Similarity:      0.67647

-> Average Score: 0.68772
```

In [8]:
```
img = generate_image_dpi(SAMPLE_1, os.path.join("processed", f"sample01_dpi_{DPI[0]}.png"), dpi=DPI)
print(f"Format: \t{img.format}\nSize: \t\t{img.size}\nChannels: \t{img.mode}")

fig1 = go.Figure()

img = img.convert('L')
img = np.array(img)
img = cv2.resize(img, None, fx=1.4, fy=1.4, interpolation=cv2.INTER_CUBIC)
imshow1(img, 'Step 0: Original (Upscaled by 1.4x)')
```

```
fig1.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 0: Original"))

# Remove shadow
median_blur_max = cv2.medianBlur(img, 301)
imshow1(median_blur_max, f'Step 1: Get Shadow - Maximum Median Blur')
fig1.add_trace(go.Histogram(x=median_blur_max.ravel(), nbinsx=256, name="Step 1: Get shadow"))

shadow_removed = img / median_blur_max
new_img = contrast_stretching(shadow_removed)
img = new_img.astype('uint8')
imshow1(img, f'Step 2: Remove Shadow')
fig1.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 2: Remove Shadow"))

# Rotate image
angle = get_skew_angle(img)
if (angle != 0):
    print(f"Angle to rotate: {angle}")
    img = rotate_image(img, angle)
    imshow1(img, f"Step 3: Rotate by '{angle:.5f}' degrees")
    fig1.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 3: Rotate Image"))
else:
    print(f"Skew angle is 0, rotation unneccessary.")

im = Image.fromarray(img)
im.save(os.path.join("processed", f"sample01_processed.png"), dpi=DPI)
```

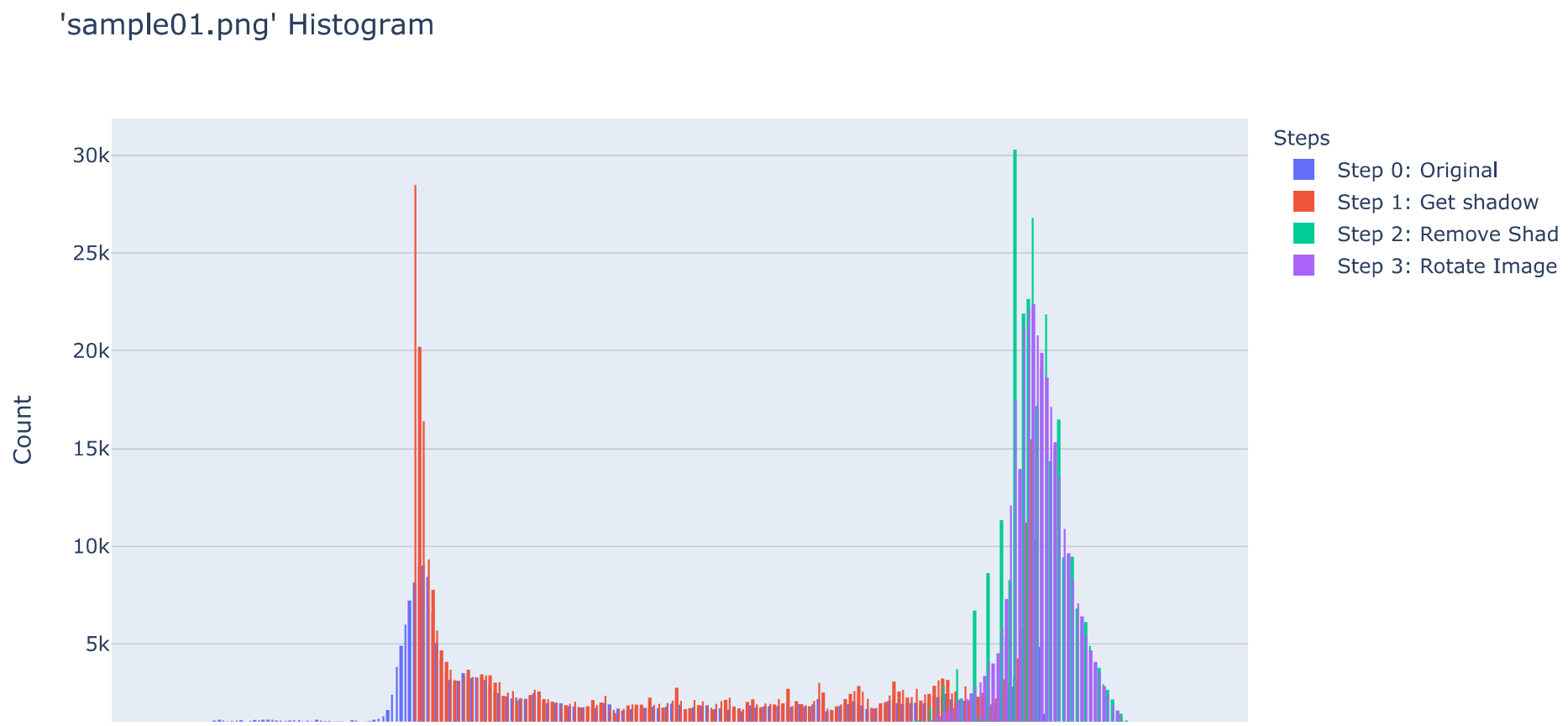
```
Format:         PNG
Size:           (965, 229)
Channels:       RGBA
Angle to rotate: -1.29095458984375
```

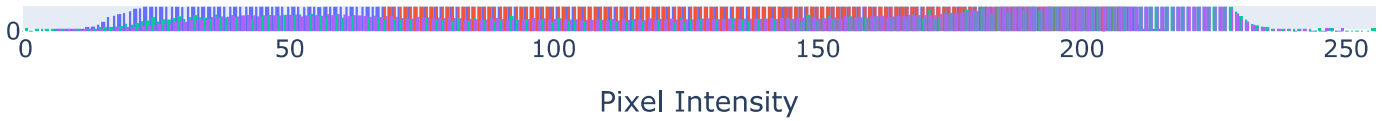



Step 1: Get Shadow - Maximum Median Blur
Step 2: Remove Shadow
Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon
Under School Age Children:While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.
Step 3: Rotate by '-1.29095' degrees
Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon
Under School Age Children:While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

In [9]:

```
# Histogram Plots for the above processed images
# Commented to keep within Github 100mb space limits.
# Uncomment to see interactive histogram plot. Alternatively, refer to 'OCR_Project.html'

fig1.update_layout(
    title="'sample01.png' Histogram",
    xaxis_title="Pixel Intensity",
    yaxis_title="Count",
    legend_title="Steps",
    autosize=True,
    width=975,
    height=550
)
fig1.show()
```





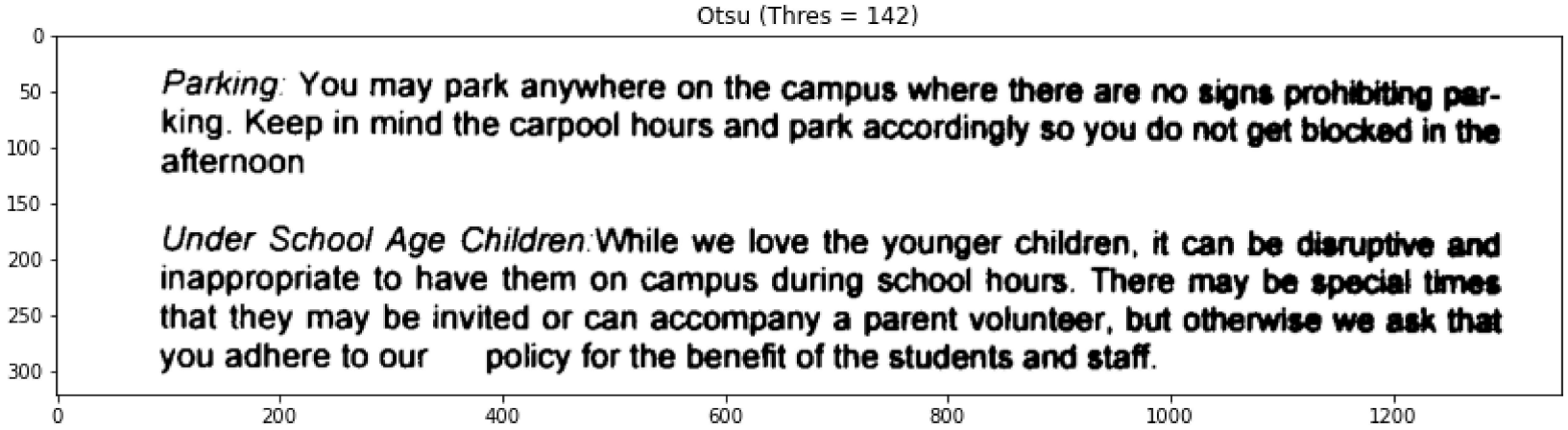
In [10]:
```
_, _, results_str1 = get_metric_score(img, SAMPLE_1_GROUND_TRUTH);
with open(SAMPLE_1_PREDICTION, "w") as text_file:
    text_file.write(results_str1)
```

```
================== Results ==================

   '''
   Parking: You may park anywhere on the campus where there are no signs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   afternoon

   Under School Age Children:While we love the younger children, it can be disruptive and
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
[]

-> Jaccard Similarity:          1.00000
-> Cosine Similarity:           1.00000
-> Levenshtein Similarity:      1.00000

-> Average Score: 1.00000
```

In [11]:
```
otsu_img, otsu_thres = otsu_thresholding(img);
imshow1(otsu_img, f'Otsu (Thres = {otsu_thres})')

get_metric_score(otsu_img, SAMPLE_1_GROUND_TRUTH);
```

```
C:\Users\wilso\AppData\Roaming\Python\Python36\site-packages\numpy\core\fromnumeric.py:3367: RuntimeWarning:

Degrees of freedom <= 0 for slice

================== Results ==================

   '''
```

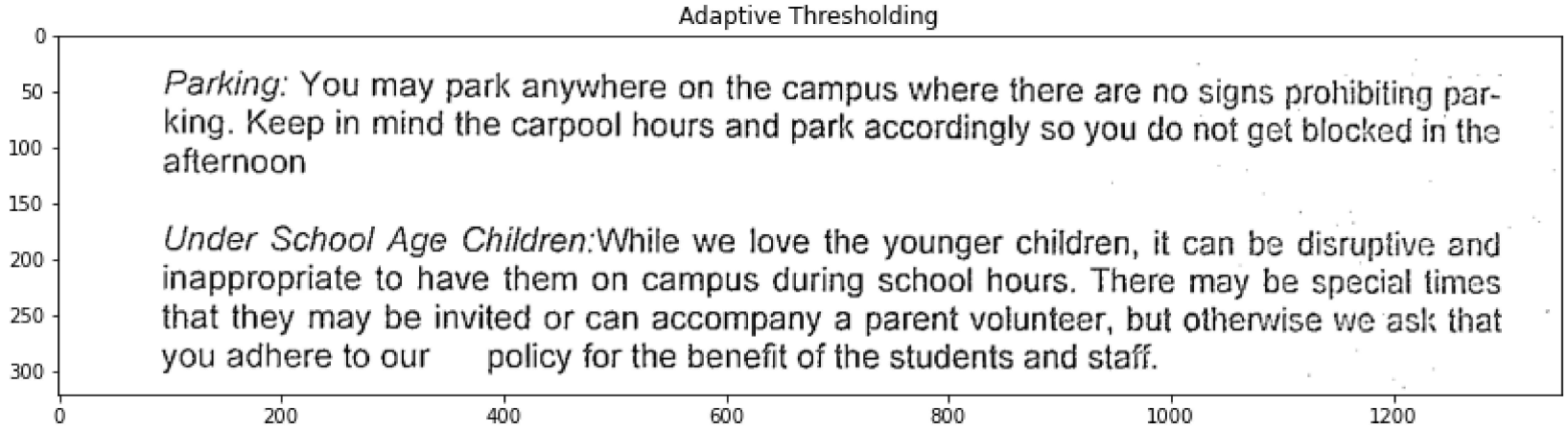
```
   Parking You may park anywhere on the campus where there are no igNs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   alternoon

   Under School Age Children While we love the younger children, it can be disruptive and
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
['Children', 'Children:While', 'Parking', 'Parking:', 'While', 'afternoon', 'alternoon', 'igNs', 'signs']

-> Jaccard Similarity:          0.88158
-> Cosine Similarity:           0.98817
-> Levenshtein Similarity:      0.99516

-> Average Score: 0.95497
```



In [12]:
```
result = cv2.adaptiveThreshold(img, 255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,5,11)
imshow1(result, 'Adaptive Thresholding')
get_metric_score(result, SAMPLE_1_GROUND_TRUTH);
```

```
================== Results ==================

   '''
   Parking: You may park anywhere on the campus where there are no signs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   afternoon

   Under School Age Children:While we love the younger children, it can be disruptive and
```

```
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
[]

-> Jaccard Similarity:          1.00000
-> Cosine Similarity:           1.00000
-> Levenshtein Similarity:      1.00000

-> Average Score: 1.00000
```



Sample02.png

In [13]:
```
# Original image results
_, _, results_str2 = get_metric_score(SAMPLE_2, SAMPLE_2_GROUND_TRUTH);
with open(SAMPLE_2_ORIGINAL_PREDICTION, "w") as text_file:
    text_file.write(results_str2)
```

```
================== Results ==================

   '''
   Sonnet for Lena
   '''

Jaccard errors:
["'Damn", 'Alas!', 'And', 'But', 'Colthurst', 'Crays', 'First', 'Hard', 'I', "I'll", 'If', 'It', 'Lena,', 'O', 'Thirteen', 'Thoma
s', 'VQ', 'Your', 'a', 'all', 'and', 'are', 'beauty', 'belong', 'cheeks', 'compress.', 'contains', 'cosines.', 'could', 'dear', 'd
```

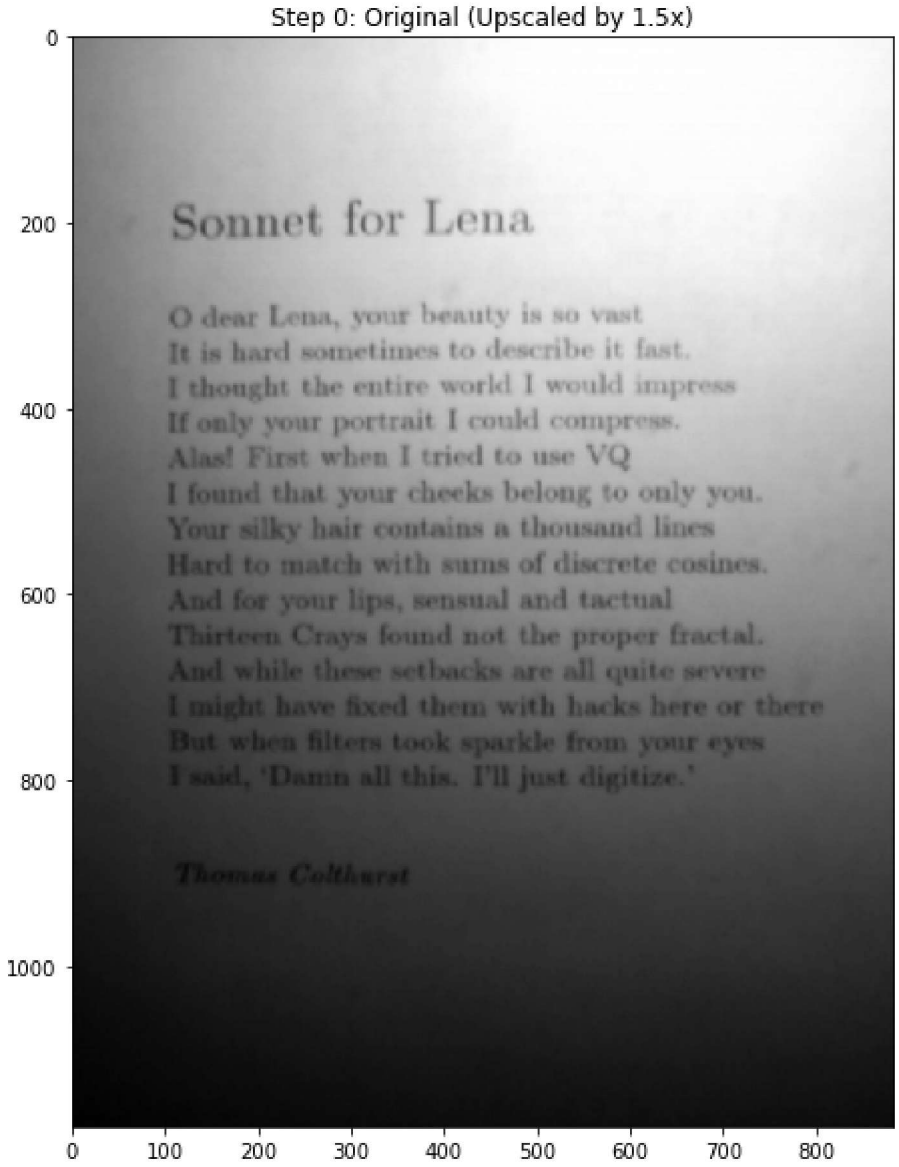
```
escribe', 'digitize."', 'discrete', 'entire', 'eyes', 'fast.', 'filters', 'fixed', 'found', 'fractal.', 'from', 'hacks', 'hair',
'hard', 'have', 'here', 'impress', 'is', 'it', 'just', 'lines', 'lips,', 'match', 'might', 'not', 'of', 'only', 'or', 'portrait',
'proper', 'quite', 'said,', 'sensual', 'setbacks', 'severe', 'silky', 'so', 'sometimes', 'sparkle', 'sums', 'tactual', 'that', 'th
e', 'them', 'there', 'these', 'this.', 'thought', 'thousand', 'to', 'took', 'tried', 'use', 'vast', 'when', 'while', 'with', 'worl
d', 'would', 'you.', 'your']

-> Jaccard Similarity:          0.03191
-> Cosine Similarity:           0.21698
-> Levenshtein Similarity:      0.04931

-> Average Score: 0.09940
```

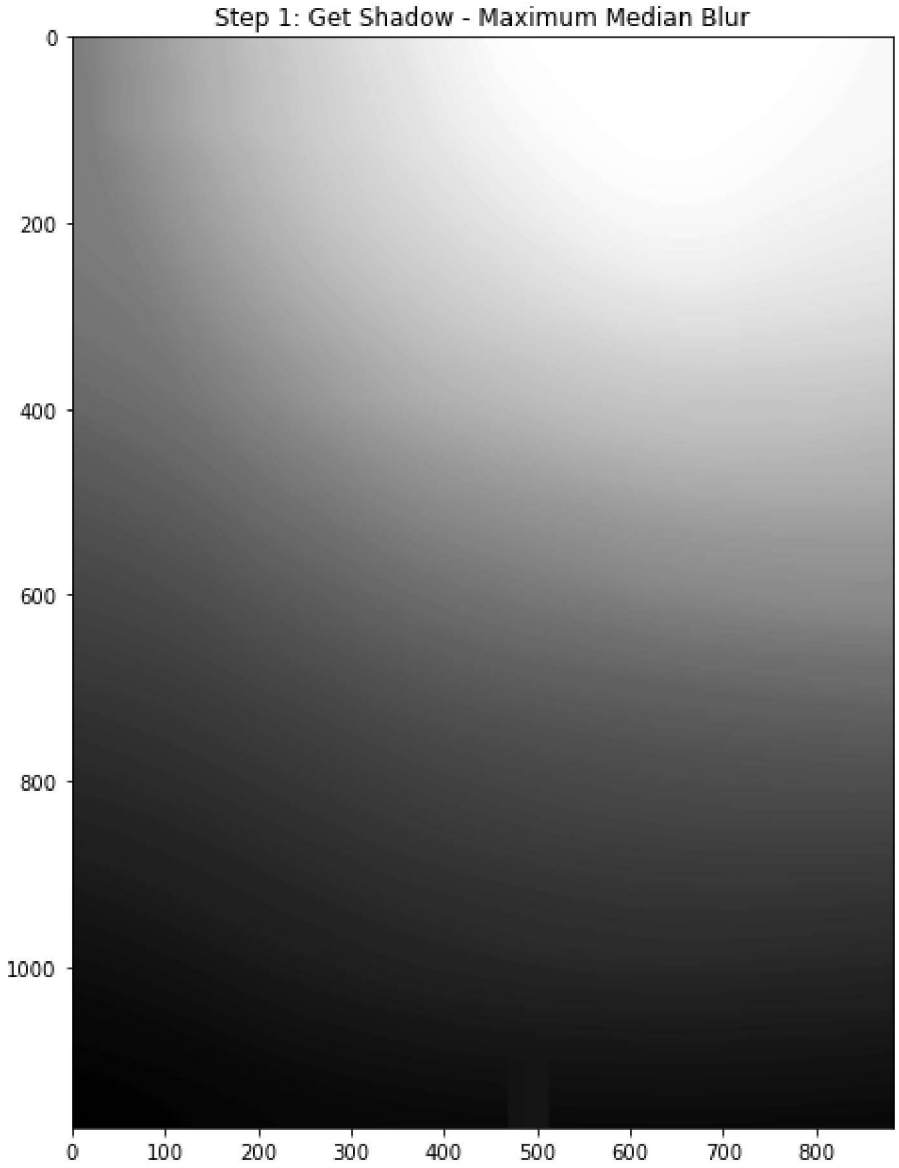
In [14]:

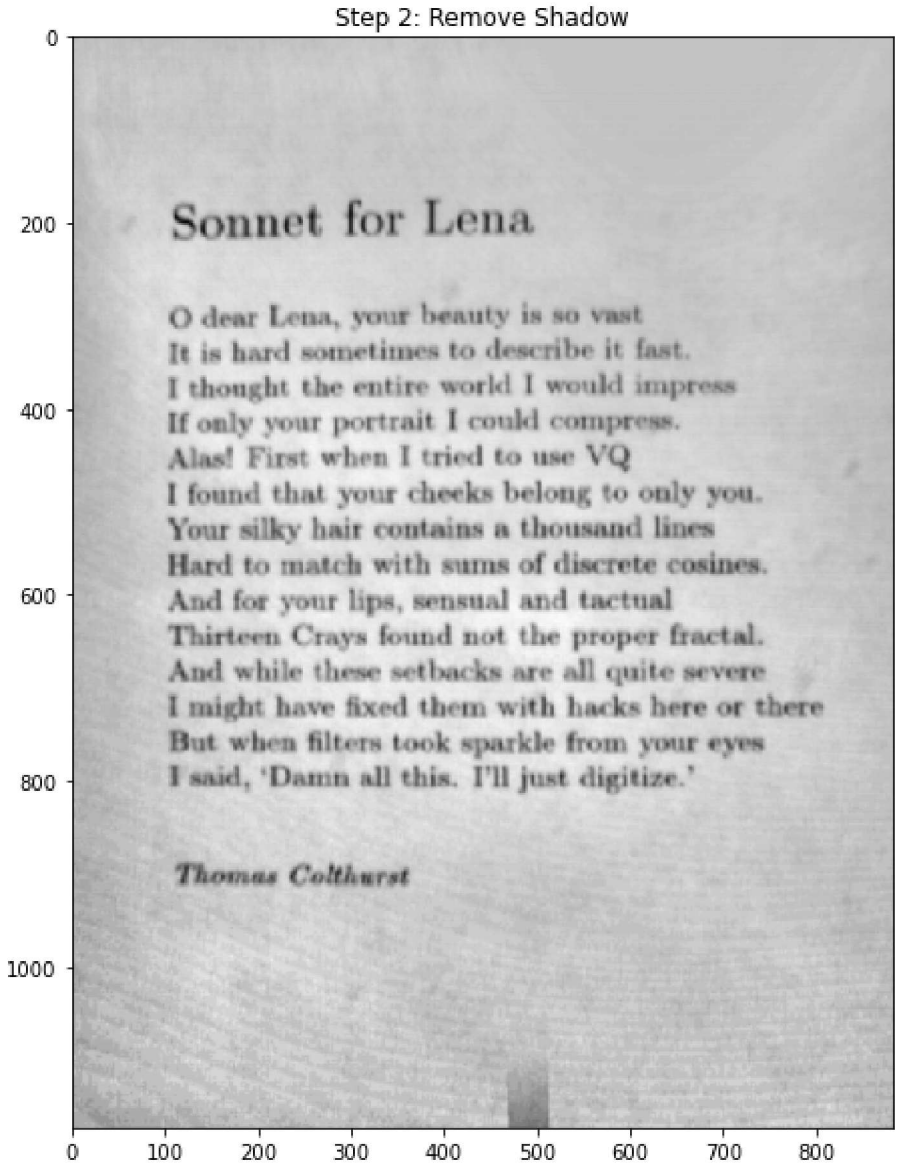
```python
img = generate_image_dpi(SAMPLE_2, os.path.join("processed", f"sample02_dpi_{DPI[0]}.png"), dpi=DPI)
print(f"Format: \t{img.format}\nSize: \t\t{img.size}\nChannels: \t{img.mode}")

fig2 = go.Figure()

img = img.convert('L')
img = np.array(img)
img = cv2.resize(img, None, fx=1.5, fy=1.5, interpolation=cv2.INTER_CUBIC)
imshow2(img, 'Step 0: Original (Upscaled by 1.5x)')
fig2.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 0: Original"))

# Remove shadow
median_blur_max = cv2.medianBlur(img, 339)
imshow2(median_blur_max, f'Step 1: Get Shadow - Maximum Median Blur')
fig2.add_trace(go.Histogram(x=median_blur_max.ravel(), nbinsx=256, name="Step 1: Get shadow"))

shadow_removed = img / median_blur_max
new_img = contrast_stretching(shadow_removed)
img = new_img.astype('uint8')
imshow2(img, f'Step 2: Remove Shadow')
fig2.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 2: Remove Shadow"))

# Rotate image
angle = get_skew_angle(img)
if (angle != 0):
    print(f"Angle to rotate: {angle}")
    img = rotate_image(img, angle)
    imshow2(img, f"Step 3: Rotate by '{angle:.5f}' degrees")
    fig2.add_trace(go.Histogram(x=img.ravel(), nbinsx=256, name="Step 3: Rotate Image"))
else:
    print(f"Skew angle is 0, rotation unneccessary.")

# Save processed image
```

```
im = Image.fromarray(img)
im.save(os.path.join("processed", f"sample02_processed.png"), dpi=DPI)
```

```
Format:         PNG
Size:           (589, 782)
Channels:       LA
Skew angle is 0, rotation unneccessary.
```



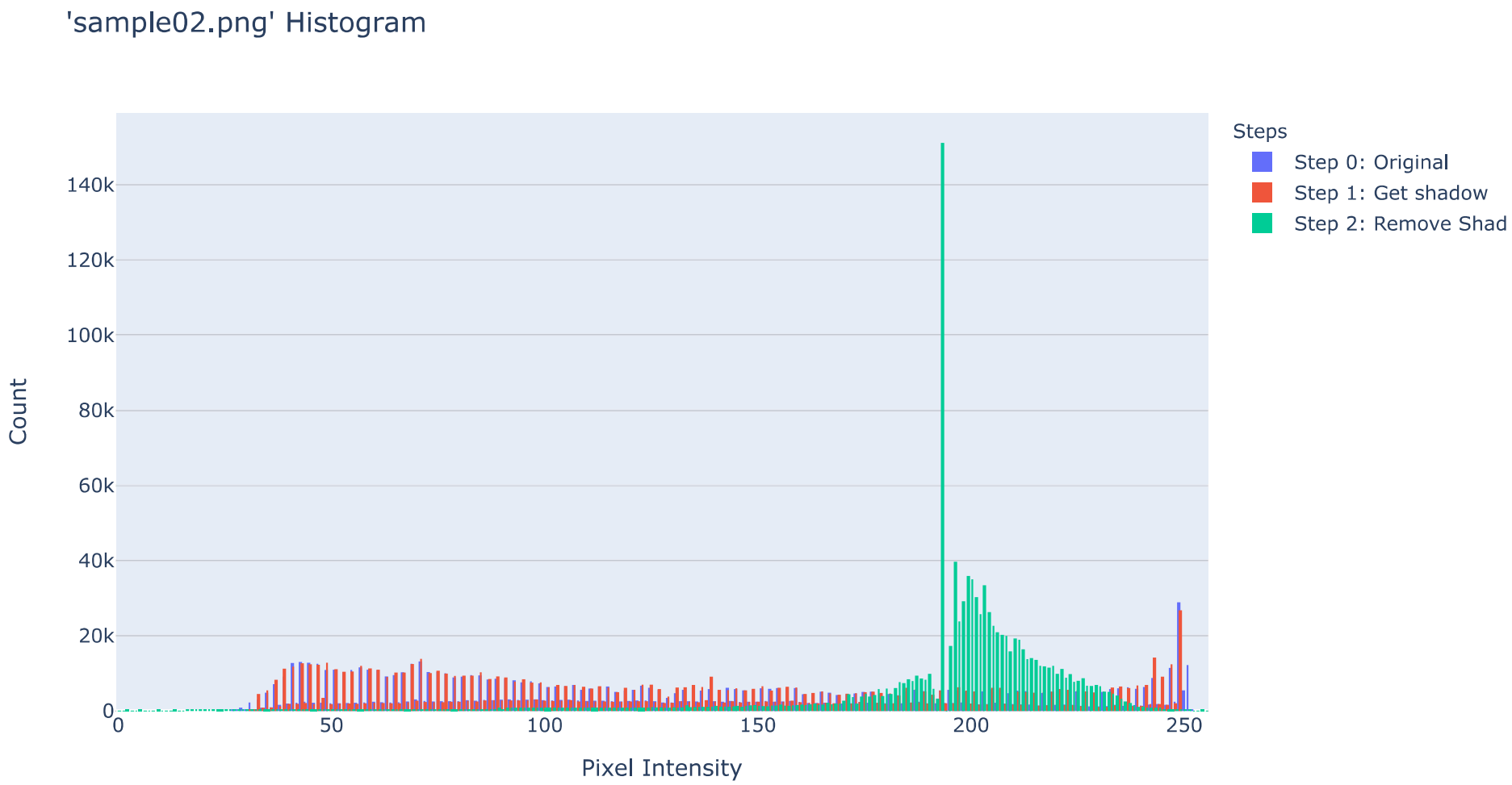

Step 1: Get Shadow - Maximum Median Blur
0
200
400
600
800
1000
0
100
200
300
400
500
600
700
800



In [15]:

```
# Histogram Plots for the above processed images
# Commented to keep within Github 100mb space limits.
# Uncomment to see interactive histogram plot. Alternatively, refer to 'OCR_Project.html'

fig2.update_layout(
    title="'sample02.png' Histogram",
```

```
    xaxis_title="Pixel Intensity",
    yaxis_title="Count",
    legend_title="Steps",
    autosize=True,
    width=975,
    height=550
)
fig2.show()
```



In [16]:
```
_, _, results_str2 = get_metric_score(img, SAMPLE_2_GROUND_TRUTH);
with open(SAMPLE_2_PREDICTION, "w") as text_file:
    text_file.write(results_str2)
```

```
================== Results ==================

   '''
   Sonnet for Lena

   O dear Lena, your beauty is so vast

   It is hard sometimes to describe it fast.

   I thought the entire world would impress
   If only your portrait could compress.

   Alas! First when I tried to use VQ

   I found that your cheeks belong to only you.
   Your silky hair contains a thousand lines
   Hard to match with sums of discrete cosines.
   And for your lips, sensual and tactual
   Thirteen Crays found not the proper fractal.
   And while these setbacks are all quite severe
   I might have fixed them with hacks here or there
   But when filters took sparkle from your eyes
   I said, Damn all this. I'll just digitize."

   Thomas Colthurst
   '''

Jaccard errors:
["'Damn", 'Damn']

-> Jaccard Similarity:          0.97895
-> Cosine Similarity:           1.00000
-> Levenshtein Similarity:      0.99209

-> Average Score: 0.99035
```

In [ ]: