UNSHARP MASKING:

เป็นเทคนิคการทำให้ภาพคมชัดขึ้นโดยการลบส่วนที่เบลอออกจากภาพต้นฉบับ

ขั้นตอน:

1. เบลอภาพต้นฉบับ: blur = Gaussian_blur(original)

2. สร้าง mask (unsharp mask): mask = original - blur

3. เพิ่ม mask กลับเข้าไปในภาพ: sharpened = original + mask

สูตร:

$$g(x,y) = f(x,y) + [f(x,y) - \bar{f}(x,y)]$$

โดย:

- $f(x,y)$ = ภาพต้นฉบับ
- $\bar{f}(x,y)$ = ภาพที่เบลอ
- $[f(x,y) - \bar{f}(x,y)]$ = unsharp mask

HIGH-BOOST FILTERING:

เป็นการขยายแนวคิดของ Unsharp Masking โดยเพิ่มค่า k (amplification factor)

สูตร:

$$g(x,y) = k \cdot f(x,y) - \bar{f}(x,y)$$

หรือเขียนได้อีกแบบ:

$$g(x,y) = (k-1) \cdot f(x,y) + [f(x,y) - \bar{f}(x,y)]$$

โดย:

- $k = 1$: เท่ากับ unsharp masking ปกติ
- $k > 1$: เพิ่มความคมชัดมากขึ้น (high-boost)
- $k < 1$: ลดความคมชัด

เทคนิคที่ใช้:
1. Unsharp Masking (amount = 1.0)
2. High-boost Filtering (k = 1.0, 1.5, 2.0, 2.5)

Parameters:
- Gaussian kernel size: {kernel_size}×{kernel_size}
- Sigma (σ): {sigma}

ผลลัพธ์:
Unsharp masking ให้ผลการ sharpen แบบธรรมชาติ
High-boost filtering (k > 1) เพิ่มความคมชัดมากขึ้นตามค่า k
ค่า k ที่สูงเกินไปอาจทำให้เกิด artifacts และ amplify noise

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from datetime import datetime


image_path = 'datasets/mega_space_molly.jpg'

img = cv2.imread(image_path)
if img is None:
    print(f"❌ Error: ไม่สามารถอ่านภาพได้ {image_path}")
    exit(1)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

def apply_unsharp_masking(image, kernel_size=5, sigma=1.0, amount=1.0):
    """
    Unsharp Masking

    Parameters:
    - image: ภาพต้นฉบับ (RGB)
    - kernel_size: ขนาด Gaussian kernel (ต้องเป็นเลขคี่)
    - sigma: ค่า standard deviation ของ Gaussian
    - amount: ความแรงของการ sharpen (โดยทั่วไป = 1.0 สำหรับ unsharp masking)

    Returns:
    - sharpened: ภาพที่ทำ unsharp masking แล้ว
    - mask: unsharp mask
    - blurred: ภาพที่เบลอ
    """
    # แปลงเป็น float เพื่อการคำนวณ
    image_float = image.astype(np.float32)

    # Step 1: เบลอภาพด้วย Gaussian blur
    blurred = cv2.GaussianBlur(image_float, (kernel_size, kernel_size), sigma)

    # Step 2: สร้าง unsharp mask = original - blurred
    mask = image_float - blurred

    # Step 3: เพิ่ม mask กลับเข้าไปในภาพ
    # sharpened = original + amount * mask
    sharpened = image_float + amount * mask

    # Clip ค่าให้อยู่ในช่วง [0, 255]
    sharpened = np.clip(sharpened, 0, 255).astype(np.uint8)

    return sharpened, mask, blurred.astype(np.uint8)


def apply_highboost_filtering(image, kernel_size=5, sigma=1.0, k=1.5):
    """
    High-boost Filtering

    Parameters:
    - image: ภาพต้นฉบับ (RGB)
    - kernel_size: ขนาด Gaussian kernel
    - sigma: ค่า standard deviation ของ Gaussian
    - k: amplification factor (k > 1 สำหรับ high-boost)

    Returns:
    - enhanced: ภาพที่ทำ high-boost filtering แล้ว

    สูตร: g(x,y) = k·f(x,y) - f̄(x,y)
    """
    # แปลงเป็น float
    image_float = image.astype(np.float32)

    # เบลอภาพ
    blurred = cv2.GaussianBlur(image_float, (kernel_size, kernel_size), sigma)

    # High-boost filtering: k*original - blurred
    enhanced = k * image_float - blurred

    # Clip ค่าให้อยู่ในช่วง [0, 255]
    enhanced = np.clip(enhanced, 0, 255).astype(np.uint8)

    return enhanced
```

```python
# ========================================
# PROCESSING
# ========================================

print("\n" + "="*80)
print("กำลังประมวลผล...")
print("="*80)

# Parameters
kernel_size = 5
sigma = 1.0

# 1. Unsharp Masking (amount = 1.0)
print("\n1. Unsharp Masking (amount = 1.0)...")
unsharp_result, unsharp_mask, blurred_img = apply_unsharp_masking(
    img_rgb, kernel_size=kernel_size, sigma=sigma, amount=1.0
)

# 2. High-boost Filtering with different k values
k_values = [1.0, 1.5, 2.0, 2.5]
highboost_results = {}

print("\n2. High-boost Filtering:")
for k in k_values:
    print(f"   - k = {k}")
    result = apply_highboost_filtering(img_rgb, kernel_size=kernel_size, sigma=sigma, k=k)
    highboost_results[k] = result

print("\n✓ ประมวลผลเสร็จสมบูรณ์!")
```

```python
# ========================================
# VISUALIZATION
# ========================================

print("\n" + "="*80)
print("กำลังสร้างกราฟเปรียบเทียบ...")
print("="*80)

# Figure 1: Unsharp Masking Process
fig1, axes1 = plt.subplots(2, 2, figsize=(14, 12))
fig1.suptitle('Unsharp Masking Process', fontsize=16, fontweight='bold')

# Original
axes1[0, 0].imshow(img_rgb)
axes1[0, 0].set_title('Original Image', fontsize=12, fontweight='bold')
axes1[0, 0].axis('off')

# Blurred
axes1[0, 1].imshow(blurred_img)
axes1[0, 1].set_title(f'Blurred (Gaussian {kernel_size}x{kernel_size}, σ={sigma})', fontsize=12)
axes1[0, 1].axis('off')

# Unsharp Mask (แสดงเป็น grayscale และปรับ contrast)
mask_display = unsharp_mask.astype(np.float32)
mask_display = (mask_display - mask_display.min()) / (mask_display.max() - mask_display.min())
axes1[1, 0].imshow(mask_display)
axes1[1, 0].set_title('Unsharp Mask (Original - Blurred)', fontsize=12)
axes1[1, 0].axis('off')

# Sharpened Result
axes1[1, 1].imshow(unsharp_result)
axes1[1, 1].set_title('Sharpened (Unsharp Masking)', fontsize=12, fontweight='bold')
axes1[1, 1].axis('off')

plt.tight_layout()

# Figure 2: High-boost Filtering Comparison
fig2, axes2 = plt.subplots(2, 3, figsize=(16, 11))
fig2.suptitle('High-boost Filtering - Effect of k Parameter', fontsize=16, fontweight='bold')

# Original
axes2[0, 0].imshow(img_rgb)
axes2[0, 0].set_title('Original Image', fontsize=12, fontweight='bold')
axes2[0, 0].axis('off')

# Unsharp Masking (k=1.0 equivalent)
axes2[0, 1].imshow(unsharp_result)
axes2[0, 1].set_title('Unsharp Masking\n(k = 1.0)', fontsize=12)
axes2[0, 1].axis('off')

# High-boost k=1.5
axes2[0, 2].imshow(highboost_results[1.5])
axes2[0, 2].set_title('High-boost Filtering\n(k = 1.5)', fontsize=12, fontweight='bold')
axes2[0, 2].axis('off')

# High-boost k=2.0
axes2[1, 0].imshow(highboost_results[2.0])
axes2[1, 0].set_title('High-boost Filtering\n(k = 2.0)', fontsize=12, fontweight='bold')
axes2[1, 0].axis('off')

# High-boost k=2.5
axes2[1, 1].imshow(highboost_results[2.5])
axes2[1, 1].set_title('High-boost Filtering\n(k = 2.5)', fontsize=12, fontweight='bold')
axes2[1, 1].axis('off')

# Comparison text
comparison_text = f"""
PARAMETERS:
• Kernel size: {kernel_size}x{kernel_size}
• Sigma (σ): {sigma}

OBSERVATION:
• k = 1.0: Standard unsharp masking
• k = 1.5: Moderate sharpening
• k = 2.0: Strong sharpening
• k = 2.5: Very strong sharpening

Note: Higher k values amplify
edges and details more, but may
also amplify noise.
"""
axes2[1, 2].text(0.1, 0.5, comparison_text, fontsize=10,
                 verticalalignment='center', family='monospace',
                 bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.5))
axes2[1, 2].axis('off')

plt.tight_layout()

# Save figures
fig1.savefig('output/task4_unsharp_masking_process.png', dpi=150, bbox_inches='tight')
fig2.savefig('output/task4_highboost_comparison.png', dpi=150, bbox_inches='tight')

print("✓ บันทึกกราฟ:")
print("  - output/task4_unsharp_masking_process.png")
print("  - output/task4_highboost_comparison.png")
```

Detailed Comparison: Original vs Enhanced

Original

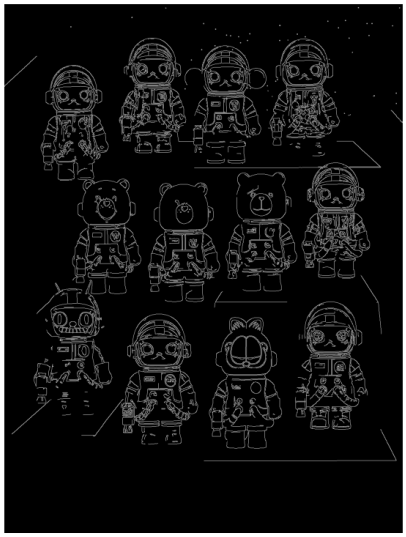
Unsharp Masking (k=1.0)

High-boost (k=1.5)

High-boost (k=2.0)

High-boost (k=2.5)

Edge Detection (Original)



High-boost Filtering - Effect of k Parameter

Original Image

Unsharp Masking
(k = 1.0)

High-boost Filtering
(k = 1.5)

High-boost Filtering
(k = 2.0)

High-boost Filtering
(k = 2.5)

PARAMETERS:
• Kernel size: 5x5
• Sigma (σ): 1.0

OBSERVATION:
• k = 1.0: Standard unsharp masking
• k = 1.5: Moderate sharpening
• k = 2.0: Strong sharpening
• k = 2.5: Very strong sharpening

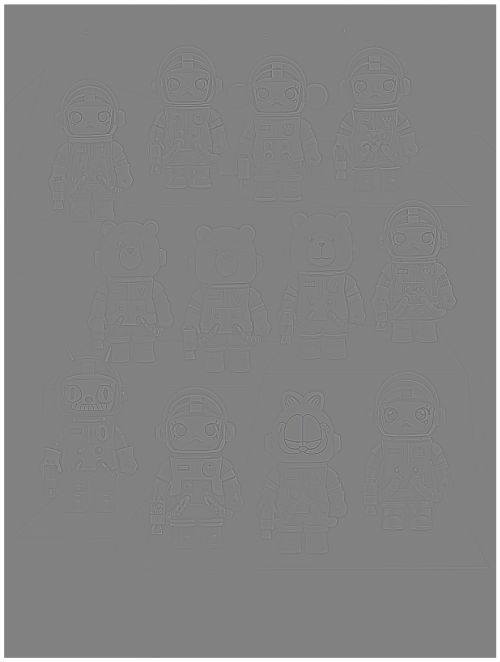
Note: Higher k values amplify edges and details more, but may also amplify noise.

Unsharp Masking Process

Original Image

Blurred (Gaussian 5x5, σ=1.0)

Unsharp Mask (Original - Blurred)



Sharpened (Unsharp Masking)