In [1]:
```
# import libraries
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

TASK #01

Enhancing and Analyzing Medical Image Quality: You are a junior researcher at a medical imaging lab, and your team has been working on improving the quality of medical images for accurate diagnosis and analysis. The lab has received a set of grayscale X-ray images from a hospital, and your task is to enhance the images using various techniques and analyze their impact on medical diagnosis. The images suffer from poor contrast and visibility issues, making it challenging for doctors to identify subtle details. You need to perform a series of operations to enhance the images and demonstrate their effectiveness.

1) Load and Display: Start by loading and displaying one of the X-ray images using OpenCV to get a sense of the image quality.

2) Contrast Enhancement: Apply histogram equalization to enhance the contrast and visibility of the X-ray image.Compare the enhanced image with the original and note any improvements in terms of details and contrast.

3) Color Mapping: Convert the enhanced grayscale image to a color-coded heatmap using the "jet" colormap. Visualize how this color mapping can help doctors identify intensity variations more effectively.

4) Color Balance: The X-ray images might have color casts due to variations in lighting during capture. Apply a color balance adjustment to remove the casts and restore the true color representation.

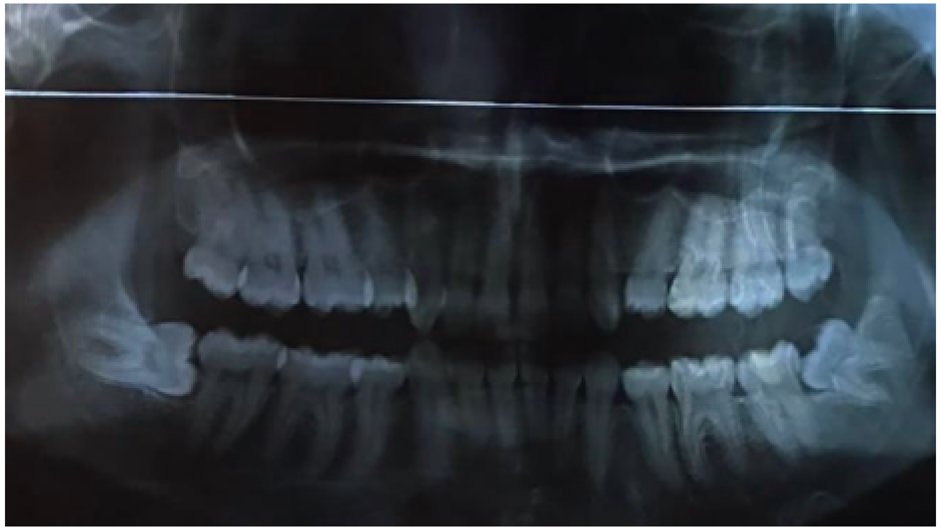
5) Color Filtering: Since X-ray images may exhibit specific color patterns due to different tissue densities, filter out a specific color range that corresponds to a particular density. Visualize the filtered image to highlight specific areas of interest.

6) Logarithmic Transformation: Apply a logarithmic transformation to enhance the visibility of darker areas in the image, which might contain critical details.

7) Power-Law Transformation: Use a power-law transformation to fine-tune the contrast of the enhanced image, further improving details' visibility.

In [2]:
```
#1 Load and display
xray = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\3\task1_xray.jpeg')
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(xray, cmap='gray')
plt.imshow(xray)
```

Out[2]:
```
<matplotlib.image.AxesImage at 0x1bcbc494eb0>
```

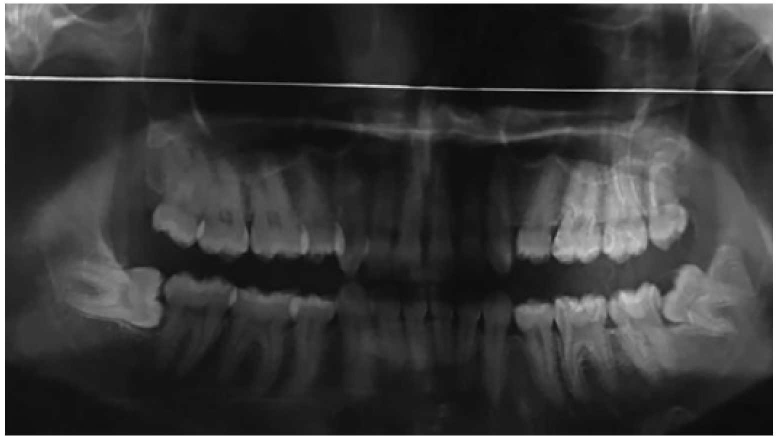


In [3]:
```
#2 Contrast Enhancement  and comparison
gray_xray = cv2.cvtColor(xray, cv2.COLOR_BGR2GRAY)
equalized_xray = cv2.equalizeHist(gray_xray)
```

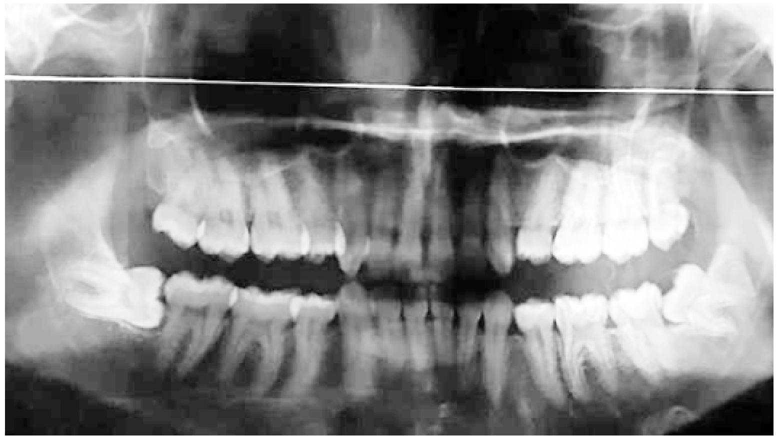
In [4]:

```
plt.figure(figsize=(10, 5), dpi = 150)
plt.subplot(121), plt.imshow(cv2.cvtColor(gray_xray, cv2.COLOR_BGR2RGB)), plt.title('Original Gray X-ray')
plt.axis('off')
plt.subplot(122), plt.imshow(equalized_xray, cmap='gray'), plt.title('Enhanced X-ray')
plt.axis('off')
plt.show()
```

Original Gray X-ray

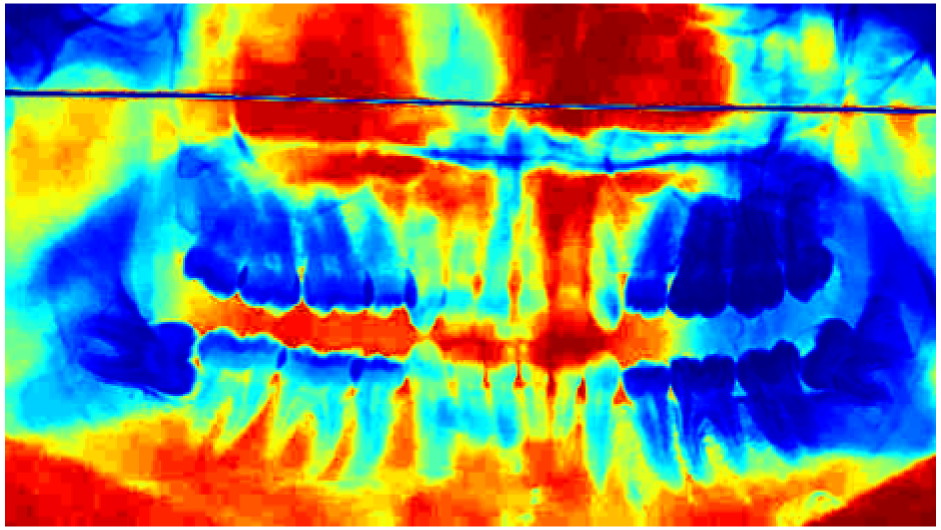


Enhanced X-ray



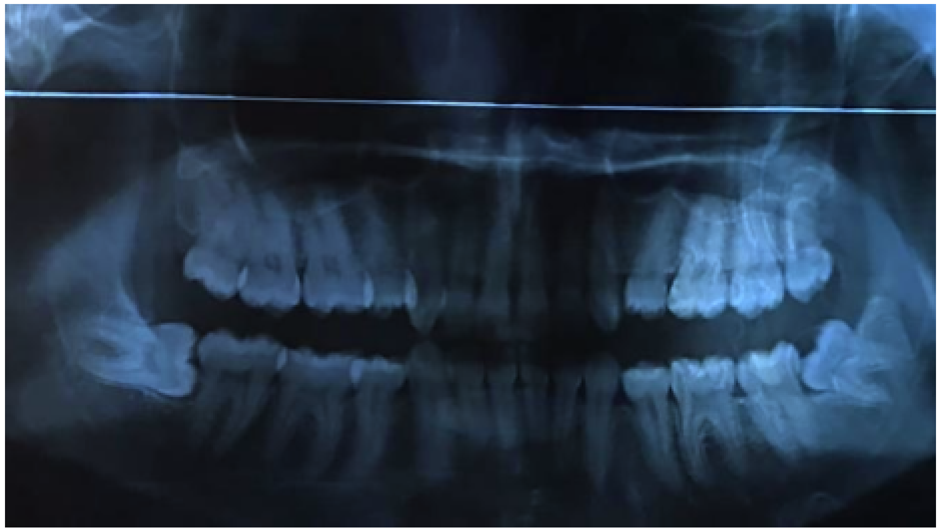
In [5]:
```python
#3 Color Mapping jet
jet_xray = cv2.applyColorMap(equalized_xray, cv2.COLORMAP_JET)
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(jet_xray)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x1bcbe5c6f40>
```



In [6]:
```
#4 Color Balance Adjustment
blue_scale = 0.8
green_scale = 1.0
red_scale = 1.2
adjusted_image = np.clip(xray * [blue_scale, green_scale, red_scale], 0, 255).astype(np.uint8)
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(adjusted_image)
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x1bcbe52b5e0>
```

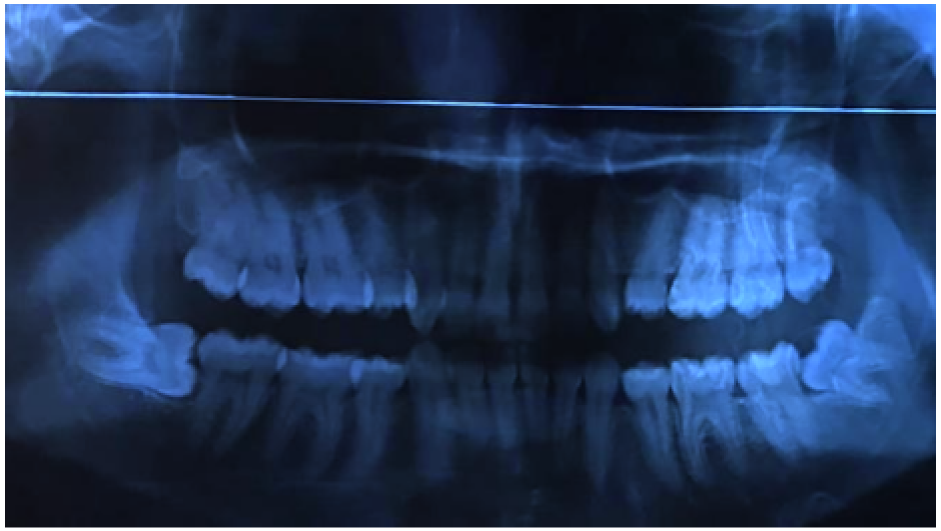


In [7]:

```python
#5 color filtering
# Color Filtering: Applying point processing on color channels can selectively emphasize or suppress specific
#in an image.
blue_channel, green_channel, red_channel = cv2.split(xray)
alpha = 1.5
beta = 0.7
adjusted_red = np.clip(red_channel * alpha, 0, 255).astype(np.uint8)
adjusted_blue = np.clip(blue_channel * beta, 0, 255).astype(np.uint8)

filtered_xray = cv2.merge((adjusted_blue, green_channel, adjusted_red))
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(filtered_xray)
```

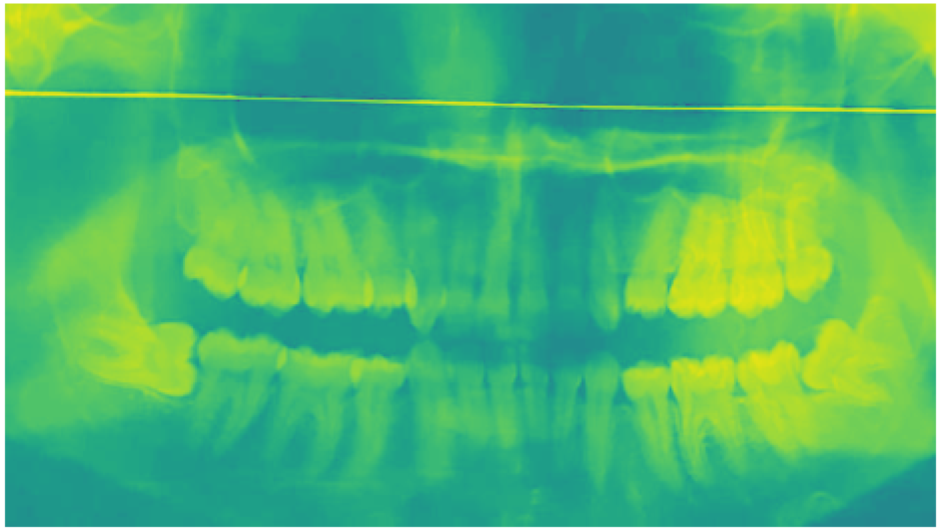
Out[7]:

```
<matplotlib.image.AxesImage at 0x1bcbe52c100>
```



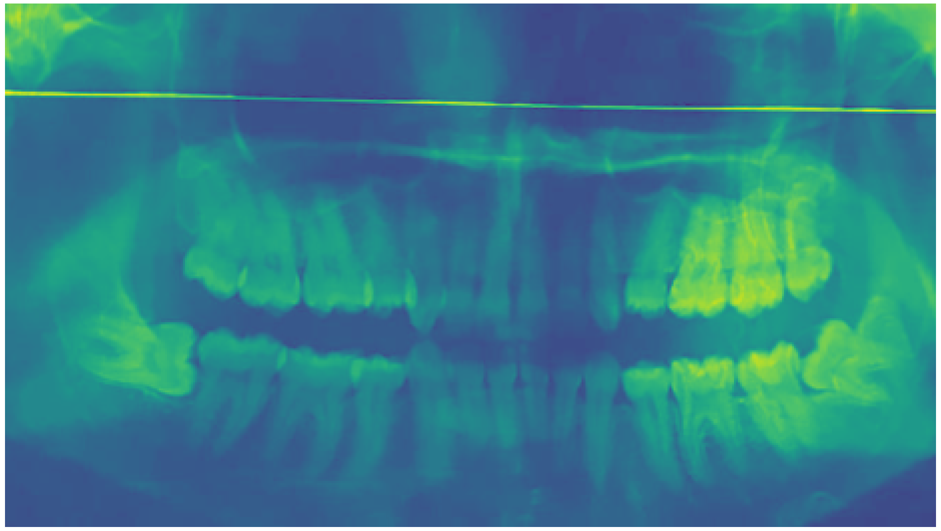
In [8]:
```python
#6 Logarithmic Transformation
c = 255 / np.log(1 + np.max(gray_xray))
log_transformed_image = c * np.log(1 + gray_xray)
# New pixel value     = c *    log(1 + old pixel value)

# Convert to uint8 type for display
log_transformed_image = np.array(log_transformed_image, dtype=np.uint8)

plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(log_transformed_image)
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x1bcbe6a29a0>
```



In [9]:
```python
#7 Power-Law Transformation
c = 1
gamma = 0.5
gamma_corrected_image = c * np.power(gray_xray, gamma)
#New pixel value      = c * (old pixel value ^ γ)

#normalization of pixels in range [0,255]
gamma_corrected_image = np.clip(gamma_corrected_image * (255 / np.max(gamma_corrected_image)), 0,255).astype(
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(gamma_corrected_image)
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x1bcbe6ffaf0>
```



TASK #02

Enhancing Multi-Modal Medical Image Fusion for Precise Diagnostics

You are now a senior researcher at a cutting-edge medical imaging institute. Your team is tasked with improving diagnostic accuracy by fusing multiple imaging modalities—X-ray and MRI scans—into a single enhanced image. This combined approach can provide comprehensive insights into both anatomical structures and tissue properties. However, the integration is complex due to varying contrasts and spatial resolutions in the modalities. Your goal is to apply the techniques you've learned to perform fusion and enhance visibility for doctors.

1) Load Modalities: Load an X-ray and an MRI image, both capturing the same anatomical region, using OpenCV.

2) Histogram Equalization for Each Modality: Apply histogram equalization separately to the X-ray and MRI images to enhance their respective contrasts.

3) Color Mapping and Fusion: Convert the enhanced X-ray and MRI images to colored heatmaps using "jet" colormaps. Overlay the colored MRI heatmap onto the X-ray heatmap, creating a fused image.

4) Multi-Modal Weighted Fusion: Adjust the weighting of the X-ray and MRI images during fusion to enhance features from each modality optimally. Use more weight for the modality with higher resolution, and less weight for the other.

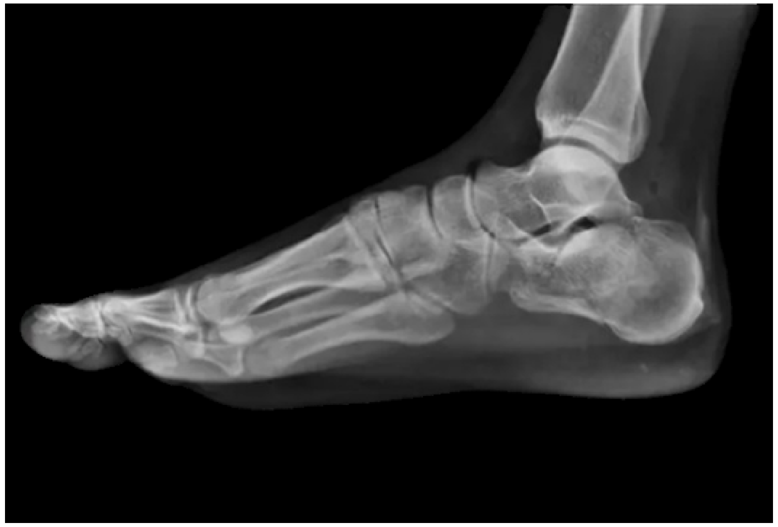
5) Logarithmic and Power-Law Transformations: Apply logarithmic and power-law transformations to the fused image to improve visibility of subtle structures and enhance contrast.

6) Comparative Analysis: Compare the original X-ray and MRI images with the fused image to evaluate the diagnostic benefits of the multi-modal fusion approach.

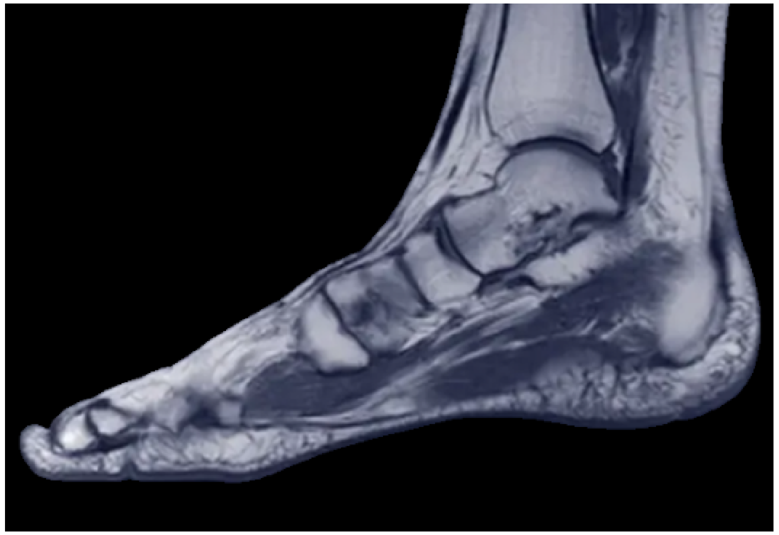
In [10]:

```python
#1) Load Modalities: Load an X-ray and an MRI image, both capturing the same anatomical region, using OpenCV.
xray = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\3\xray.png')
mri = cv2.imread(r'D:\FastSemesters\semester7\Computer Vision\lab\3\mri.png')
plt.figure(figsize=(10, 5), dpi = 150)
plt.subplot(121), plt.imshow(cv2.cvtColor(xray, cv2.COLOR_BGR2RGB)), plt.title('Foot Xray')
plt.axis('off')
plt.subplot(122), plt.imshow(cv2.cvtColor(mri, cv2.COLOR_BGR2RGB)), plt.title('Foot MRI')
plt.axis('off')
plt.show()
```

Foot Xray

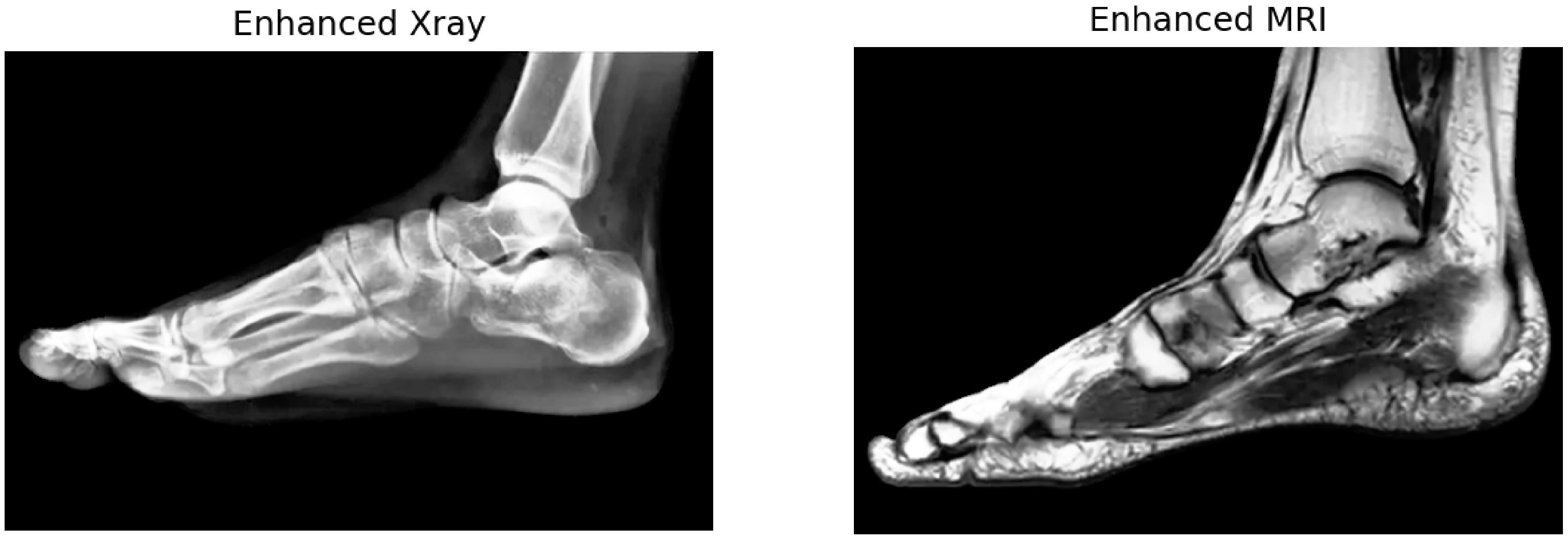


Foot MRI



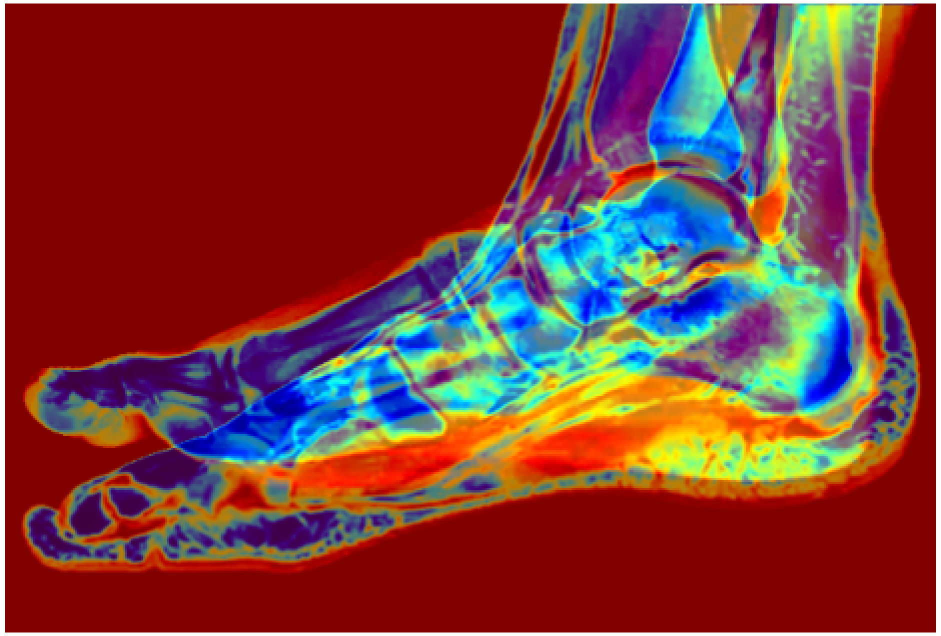
In [11]:

```
#2) Histogram Equalization for Each Modality: Apply histogram equalization separately to the X-ray and MRI im
#   enhance their respective contrasts.

gray_xray = cv2.cvtColor(xray, cv2.COLOR_BGR2GRAY)
equalized_xray = cv2.equalizeHist(gray_xray)
gray_mri = cv2.cvtColor(mri, cv2.COLOR_BGR2GRAY)
equalized_mri = cv2.equalizeHist(gray_mri)

plt.figure(figsize=(10, 5), dpi = 150)
plt.subplot(121), plt.imshow(equalized_xray, cmap='gray'), plt.title('Enhanced Xray')
plt.axis('off')
plt.subplot(122), plt.imshow(equalized_mri, cmap='gray'), plt.title('Enhanced MRI')
plt.axis('off')
plt.show()
```



In [12]:
```
#3) Color Mapping and Fusion: Convert the enhanced X-ray and MRI images to colored heatmaps using "jet" color
#   Overlay the colored MRI heatmap onto the X-ray heatmap, creating a fused image.

jet_xray = cv2.applyColorMap(equalized_xray, cv2.COLORMAP_JET)
jet_mri = cv2.applyColorMap(equalized_mri, cv2.COLORMAP_JET)
jet_mri = cv2.resize(jet_mri, (jet_xray.shape[1], jet_xray.shape[0]))
opacity = 0.5
fused_image = cv2.addWeighted(jet_xray, 1 - opacity, jet_mri, opacity, 0)
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(fused_image)
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x1bcbe827b20>
```

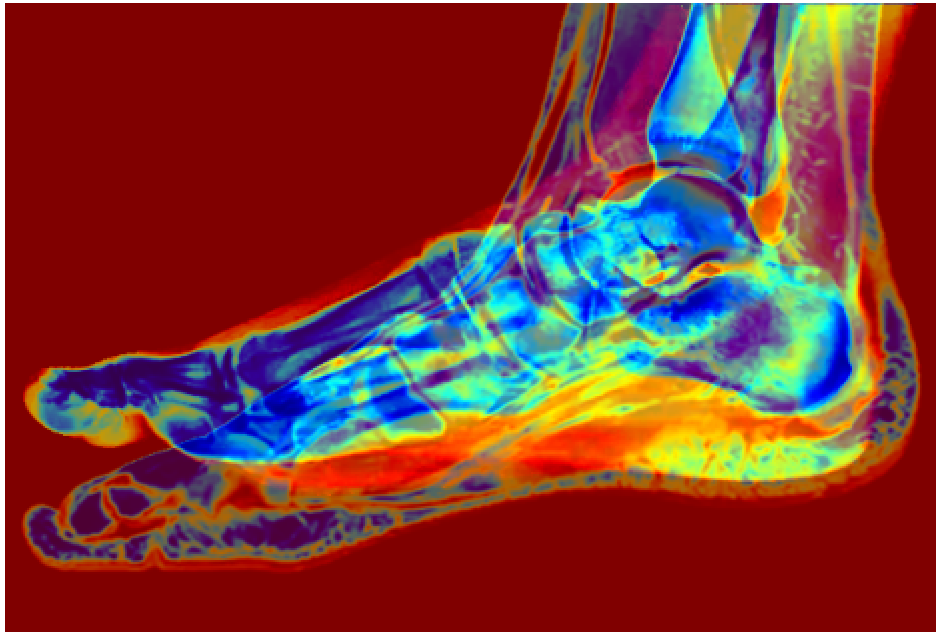


In [13]:

```python
#4) Multi-Modal Weighted Fusion: Adjust the weighting of the X-ray and MRI images during fusion to enhance fe
#  from each modality optimally. Use more weight for the modality with higher resolution, and less weight for

# Multi-modal fusion with weighted overlay
weight_xray = 0.6
weight_mri = 0.4
fused_image = cv2.addWeighted(jet_xray, weight_xray, jet_mri, weight_mri, 0)
plt.figure(figsize=(4, 3), dpi=150)
plt.axis('off')
plt.imshow(fused_image)
```

Out[13]:

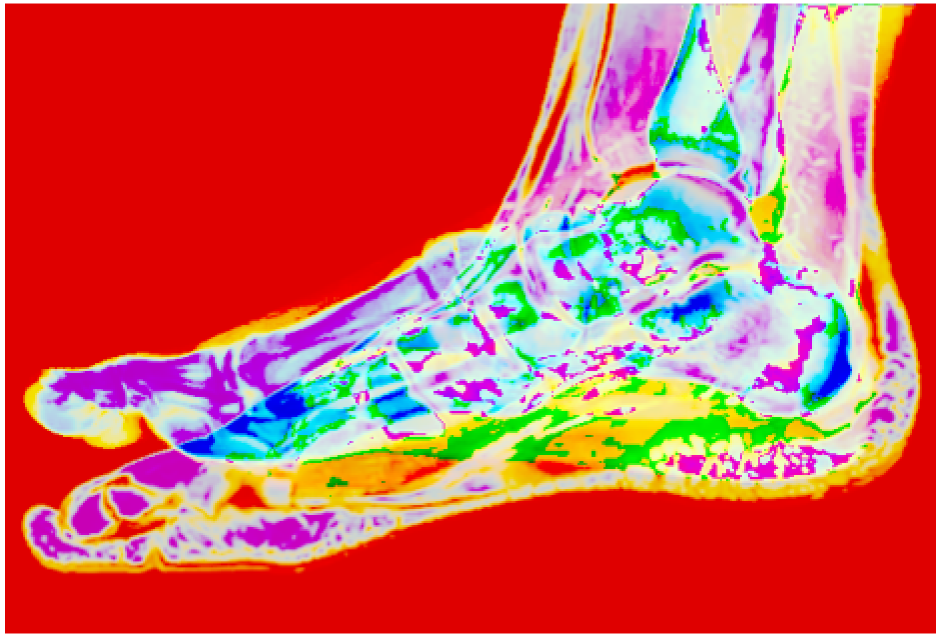
```
<matplotlib.image.AxesImage at 0x1bcbfc40130>
```



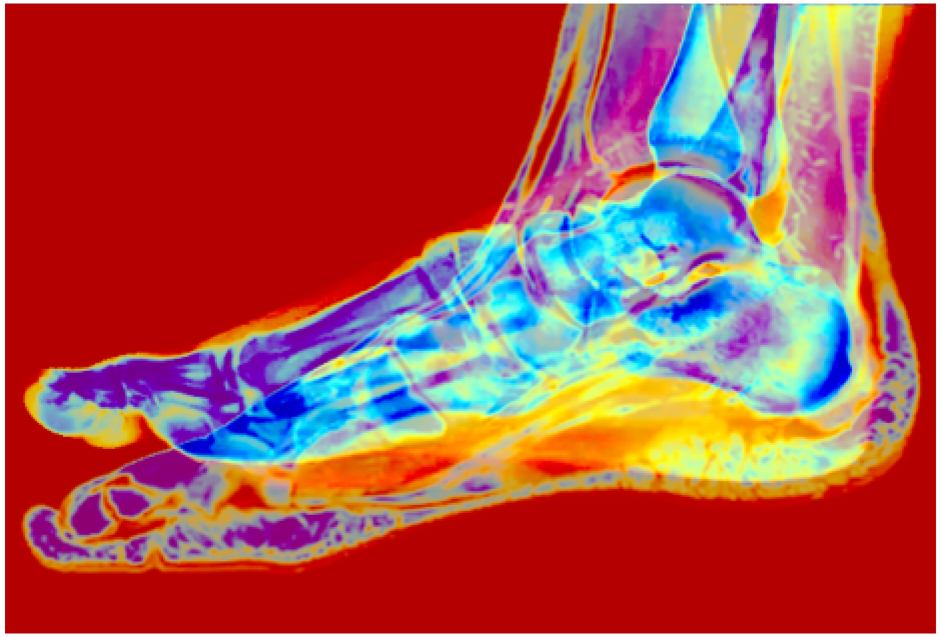
In [14]:
```
#5) Logarithmic and Power-Law Transformations: Apply logarithmic and power-law transformations to the fused i
#   improve visibility of subtle structures and enhance contrast.

#logarithmic Transformation
c = 255 / np.log(1 + np.max(fused_image))
log_transformed_image = c * np.log(1 + fused_image)
# New pixel value     = c *    log(1 + old pixel value)

# Convert to uint8 type for display
log_transformed_image = np.array(log_transformed_image, dtype=np.uint8)

plt.figure(figsize=(4, 3), dpi=150) , plt.title('logarithmically transformed')
plt.axis('off')
plt.imshow(log_transformed_image)

c = 1
gamma = 0.5
gamma_corrected_image = c * np.power(fused_image, gamma)
#New pixel value      = c * (old pixel value ^ γ)

#normalization of pixels in range [0,255]
gamma_corrected_image = np.clip(gamma_corrected_image * (255 / np.max(gamma_corrected_image)), 0,255).astype(
plt.figure(figsize=(4, 3), dpi=150) ,plt.title('power law transformed')
plt.axis('off')
plt.imshow(gamma_corrected_image)
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x1bcbfbb5fd0>
```

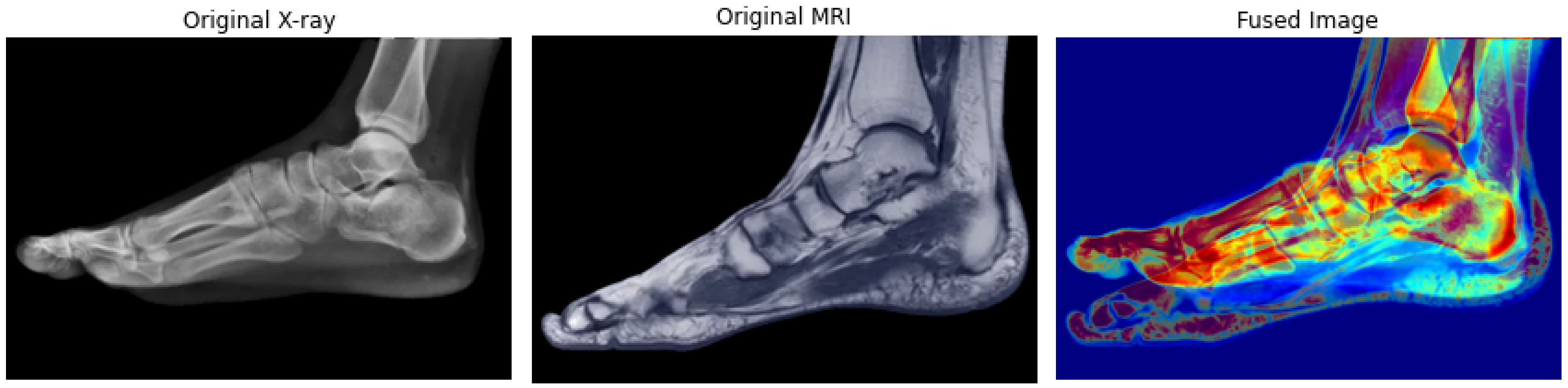
logarithmically transformed



power law transformed



In [15]:
```python
#6) Comparative Analysis: Compare the original X-ray and MRI images with the fused image to evaluate the diag
#   benefits of the multi-modal fusion approach.

# subplots for comparative analysis
# since the images were taken from a slightly different angle, the fusion is a bit messed. But overall enhanc
#analysis easier
plt.figure(figsize=(12, 4))
plt.subplot(131)
plt.title('Original X-ray')
plt.imshow(cv2.cvtColor(xray, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(132)
plt.title('Original MRI')
plt.imshow(cv2.cvtColor(mri, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(133)
plt.title('Fused Image')
plt.imshow(cv2.cvtColor(fused_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.tight_layout()
plt.show()

```



task #03

Real-Time Video Enhancement and Analysis In this task, you will capture a live video stream from your camera and apply a series of image enhancement operations to each frame in real-time. You will then display the original video stream along with individual frames enhanced using various techniques. This will allow you to observe the immediate effects of each operation on the video feed.

Video Capture Setup: Initialize the camera for video capture using OpenCV.

Configure the camera settings (resolution, frame rate, etc.) if needed.

Real-Time Video Processing Loop: Start a loop to continuously capture video frames from the camera.

For each captured frame:

Apply histogram equalization for contrast enhancement.

Convert the frame to a colored heatmap using the "jet" colormap.

Apply color balance adjustment to correct any color casts.

Apply a logarithmic transformation to enhance visibility in dark regions.

Apply a power-law transformation for fine-tuning contrast.

Display the original live video stream.

Display individual frames with each operation applied side by side for comparison.

In [18]:

```python
#run on your device
camera = cv2.VideoCapture(0)
if not camera.isOpened():
    print("error: could not open camera")
    exit()

cv2.namedWindow("Live Camera", cv2.WINDOW_NORMAL)

while True:
    ret, frame = camera.read()
    if not ret:
        print("Error: Could not read frame.")
        break

    #histogram equalization
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    equalized_frame = cv2.equalizeHist(gray_frame)

    #colored heatmap using the "jet" colormap
    jet_frame = cv2.applyColorMap(equalized_frame, cv2.COLORMAP_JET)

    #color balance adjustment
    adjusted_image = np.clip(jet_frame * [0.8, 1.0, 1.2], 0, 255).astype(np.uint8)

    #logarithmic transformation
    log_transformed_frame = np.log1p(adjusted_image.astype(np.float32))

    #power-law transformation
    gamma = 0.5
    gamma_corrected_frame = np.power(log_transformed_frame, gamma)

    cv2.imshow("Original Video", frame)
    cv2.imshow("Equalized Frame", equalized_frame)
    cv2.imshow("Adjusted Image", adjusted_image)
    cv2.imshow("Jet Colormap", jet_frame)
    cv2.imshow("Log Transform", log_transformed_frame)
    cv2.imshow("Gamma Corrected", gamma_corrected_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

```

```
camera.release()
cv2.destroyAllWindows()
```

In [ ]:

```

```