```
import cv2
import numpy as np
from keras.applications import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.models import Model

# Load the two face images
image_path1 = "/content/drive/MyDrive/Datasets/face_img_1.jpg"
image_path2 = "/content/drive/MyDrive/Datasets/face_img_2.jpg"

img1 = cv2.imread(image_path1)
img2 = cv2.imread(image_path2)

# Preprocess the images for feature extraction
img1 = cv2.resize(img1, (224, 224))
img2 = cv2.resize(img2, (224, 224))

img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

# Load the VGG16 model with pre-trained weights (excluding the top classification layers)
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Create a new model that includes only the feature extraction layers of VGG16
feature_extractor = Model(inputs=base_model.input, outputs=base_model.get_layer('block3_conv3').output)

# Preprocess the images for the VGG16 model
img1 = preprocess_input(img1)
img2 = preprocess_input(img2)

# Extract features from both images
features1 = feature_extractor.predict(np.expand_dims(img1, axis=0))
features2 = feature_extractor.predict(np.expand_dims(img2, axis=0))

# Combine the features (element-wise averaging)
fused_features = (features1 + features2) / 2
```

```
# You now have the fused features that represent the same person's face from different directions.

# You can use these features for various tasks, such as face recognition or classification.
```

```
1/1 [==============================] - 0s 428ms/step
1/1 [==============================] - 0s 386ms/step
```

```
base_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
def visualize_feature_maps_enlarged(feature_maps, upsample_factor=4):
    num_features = feature_maps.shape[-1]
    print(num_features)
    for i in range(min(num_features,10)):
        feature_map = feature_maps[0, :, :, i]

        # Upsample the feature map
        upsampled_feature_map = cv2.resize(feature_map, None, fx=upsample_factor, fy=upsample_factor, interpolation=cv2.INTER_LINEAR)

        plt.figure(figsize=(8, 8))
        plt.imshow(upsampled_feature_map, cmap='viridis')
        plt.axis('off')
        plt.show()

# Visualize enlarged feature maps for Image 1 (you can also do this for Image 2)
visualize_feature_maps_enlarged(features1, upsample_factor=8)
```

```
256
```