In [2]:

```python
import zipfile
with zipfile.ZipFile(r"C:\Users\HP\Downloads\hand gesture.zip","r") as zip_ref:
    zip_ref.extractall("targetdir")
```

In [3]:

```python
import numpy as np # We'll be storing our data as numpy arrays
import os # For handling directories
from PIL import Image # For handling the images
import matplotlib.pyplot as plt
import matplotlib.image as mpimg # Plotting
```

In [10]:

```
%%time
lookup = dict()
reverselookup = dict()
count = 0
for j in os.listdir(r"D:\Gesture\leapGestRecog\00"):
    if not j.startswith('.'): # If running this code locally, this is to
                              # ensure you aren't reading in hidden folders
        lookup[j] = count
        reverselookup[count] = j

        count = count + 1
lookup
```

```
CPU times: total: 0 ns
Wall time: 1.07 ms


{'01_palm': 0,
 '02_l': 1,
 '03_fist': 2,
 '04_fist_moved': 3,
 '05_thumb': 4,
 '06_index': 5,
 '07_ok': 6,
 '08_palm_moved': 7,
 '09_c': 8,
 '10_down': 9}
```

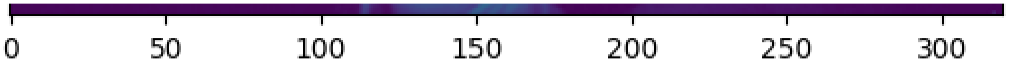
In [12]:

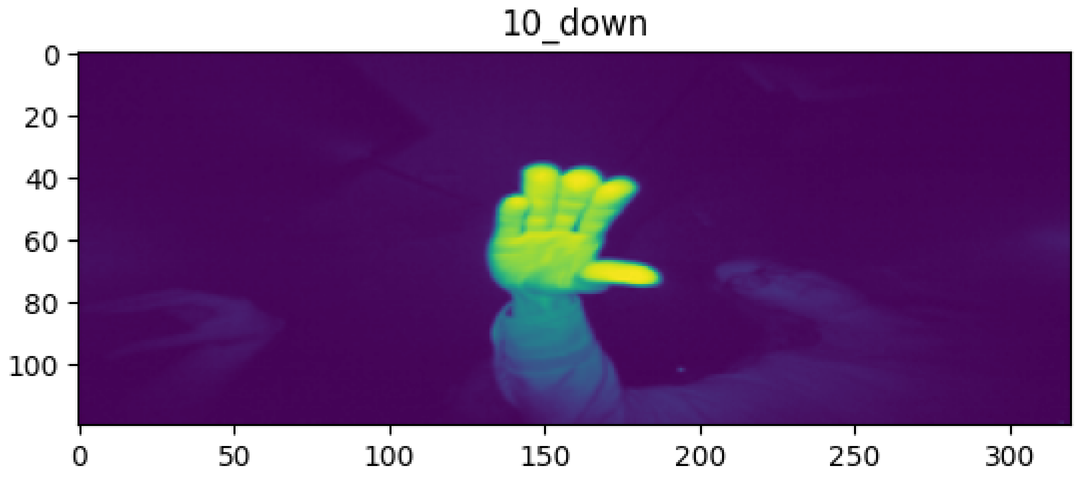
```
%%time
x_data = []
y_data = []
datacount = 0 # We'll use this to tally how many images are in our dataset
for i in range(0, 10): # Loop over the ten top-level folders
    for j in os.listdir(r"D:\Gesture\leapGestRecog\leapGestRecog\0" + str(i) + '/'):
        if not j.startswith('.'): # Again avoid hidden folders
            count = 0 # To tally images of a given gesture
            for k in os.listdir(r"D:\Gesture\leapGestRecog\leapGestRecog\0" +
                                str(i) + '/' + j + '/'):
                                # Loop over the images
                img = Image.open(r"D:\Gesture\leapGestRecog\leapGestRecog\0" +
                                 str(i) + '/' + j + '/' + k).convert('L')
                                # Read in and convert to greyscale
                img = img.resize((320, 120))
                arr = np.array(img)
                x_data.append(arr)
                count = count + 1
            y_values = np.full((count, 1), lookup[j])
            y_data.append(y_values)
            datacount = datacount + count
x_data = np.array(x_data, dtype = 'float32')
y_data = np.array(y_data)
y_data = y_data.reshape(datacount, 1) # Reshape to be the correct size
```

```
CPU times: total: 8.72 s
Wall time: 2min 9s
```

In [13]:

```
%%time
from random import randint
for i in range(0, 10):
    plt.imshow(x_data[i*200 , :, :])
    plt.title(reverselookup[y_data[i*200 ,0]])
    plt.show()
```





```
CPU times: total: 500 ms
Wall time: 1.77 s
```

In [14]:

```
import keras
from keras.utils import to_categorical
y_data = to_categorical(y_data)
```

```
WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_sof
tmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [15]:

```
x_data = x_data.reshape((datacount, 120, 320, 1))
x_data /= 255
```

In [16]:

```
from sklearn.model_selection import train_test_split
x_train,x_further,y_train,y_further = train_test_split(x_data,y_data,test_size = 0.2)
x_validate,x_test,y_validate,y_test = train_test_split(x_further,y_further,test_size = 0.5)
```

In [17]:

```
from keras import layers
from keras import models
```

In [18]:

```
model=models.Sequential()
model.add(layers.Conv2D(32, (5, 5), strides=(2, 2), activation='relu', input_shape=(120, 320,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\backend.py:873: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name
tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [19]:

```
%%time
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=64, verbose=1, validation_data=(x_validate, y_validate
```

```
WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\optimizers\__init__.py:309: The name tf.trai
n.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/5
WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.Rag
gedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.e
xecuting_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

250/250 [==============================] - 36s 138ms/step - loss: 0.3129 - accuracy: 0.9003 - val_loss: 0.0050 - val_accuracy: 0.9995
Epoch 2/5
250/250 [==============================] - 32s 130ms/step - loss: 0.0170 - accuracy: 0.9955 - val_loss: 4.7832e-04 - val_accuracy: 1.0000
Epoch 3/5
250/250 [==============================] - 34s 136ms/step - loss: 0.0085 - accuracy: 0.9980 - val_loss: 0.0061 - val_accuracy: 0.9985
Epoch 4/5
250/250 [==============================] - 33s 130ms/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 6.0156e-04 - val_accuracy: 1.0000
Epoch 5/5
250/250 [==============================] - 32s 129ms/step - loss: 0.0036 - accuracy: 0.9987 - val_loss: 1.2652e-04 - val_accuracy: 1.0000
CPU times: total: 10min 42s
Wall time: 2min 58s
```

```
<keras.src.callbacks.History at 0x2ddcf15a7d0>
```

In [20]:

```python
%%time
[loss, acc] = model.evaluate(x_test,y_test,verbose=1)
print("Accuracy:" + str(acc))
```

```
63/63 [==============================] - 2s 22ms/step - loss: 2.3246e-04 - accuracy: 1.0000
Accuracy:1.0
CPU times: total: 3.45 s
Wall time: 3.89 s
```

In [21]:

```python
%%time
# Model weights and model
model.save_weights('gesture_model_weights.h5')
model.save("gesture_model.h5")
```

```
CPU times: total: 31.2 ms
Wall time: 124 ms
```

```
C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an
HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save
('my_model.keras')`.
  saving_api.save_model(
```

In [22]:

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow import image
import numpy as np
```

In [23]:

```python
model.save('gesture_recognition_model.h5')


# model.save_weights('gesture_recognition_model_weights.h5')
```

In [24]:

```python
from keras.models import load_model

loaded_model = load_model('gesture_recognition_model.h5')

#loaded_model.load_weights('gesture_recognition_model_weights.h5')
```

In [25]:

```python
from keras.preprocessing import image
import numpy as np
```

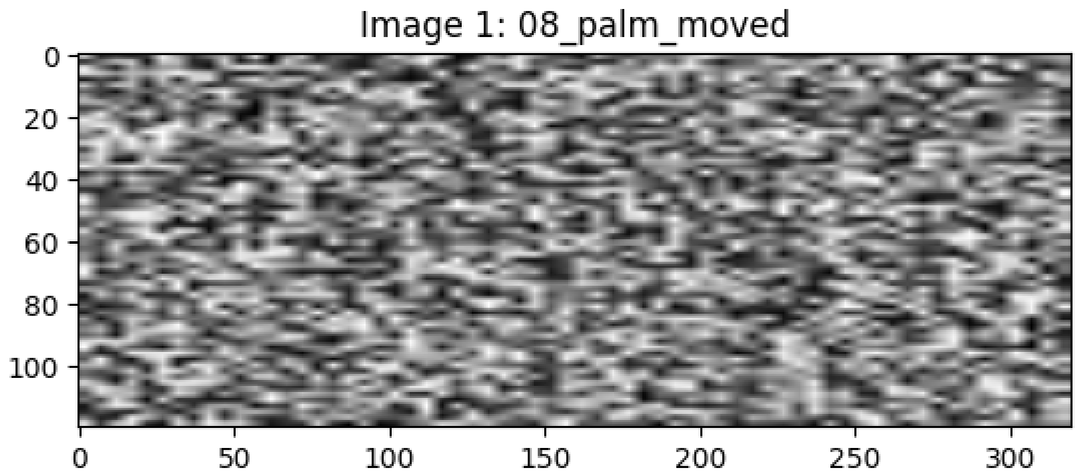
In [28]:

```
t_test = []

datacount = 0 # We'll use this to tally how many images are in our dataset
folder_path = '/kaggle/input/test2-img/'

for filename in os.listdir(r"D:\Gesture\leapGestRecog\leapGestRecog"):
    if filename.endswith(".jpg") or filename.endswith(".png"):

        img_path = os.path.join(folder_path, filename)
        count = 0 # To tally images of a given gesture


        img = Image.open(img_path).convert('L')  # Convert to grayscale
        img = img.resize((320, 120))
        arr = np.array(img)
        t_test.append(arr)
        count = count + 1

    datacount = datacount + count
t_test = np.array(t_test, dtype = 'float32')

```

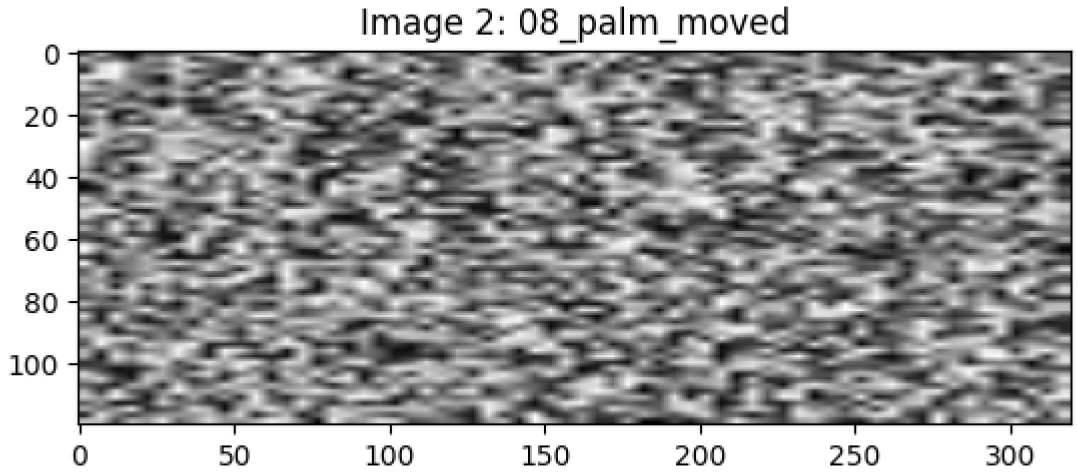
In [48]:

```python
import cv2  # Import OpenCV for image resizing

predicted_gestures = []

for i in range(t_test.shape[0]):
    img_show = t_test[i].reshape(64, 64)  # Adjust the shape according to your actual image size
    img_show_resized = cv2.resize(img_show, (320, 120))  # Resize the image to (120, 320)

    img2 = img_show_resized.reshape(1, 120, 320, 1)  # Adjust the shape accordingly
    img2 /= 255.0

    predictions = loaded_model.predict(img2)

    predicted_class = np.argmax(predictions)
    predicted_gesture = reverselookup[predicted_class]
    predicted_gestures.append(predicted_gesture)

    plt.imshow(img_show_resized, cmap='gray')

    plt.title(f"Image {i + 1}: {predicted_gesture}")
    plt.show()

print("Predicted Gestures:", predicted_gestures)
```

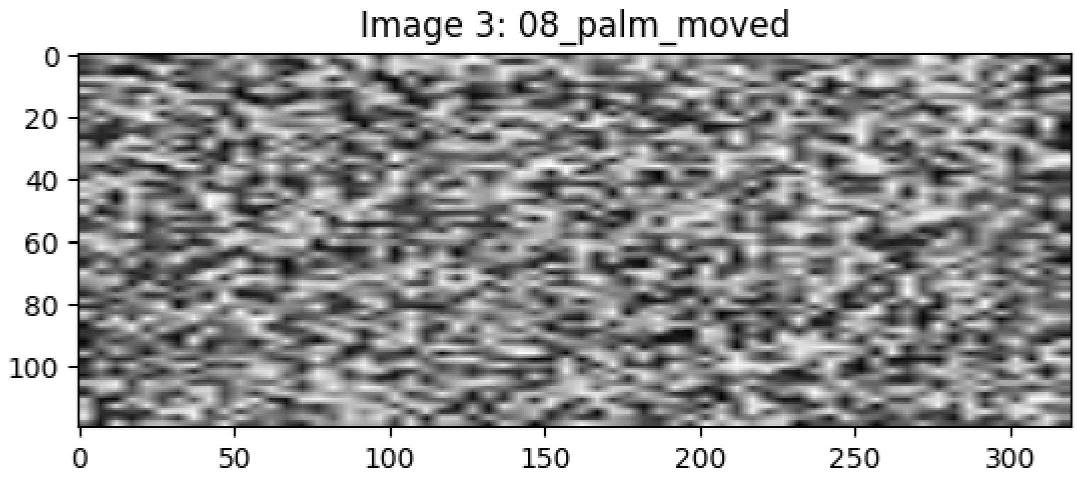
```
1/1 [==============================] - 0s 94ms/step
```



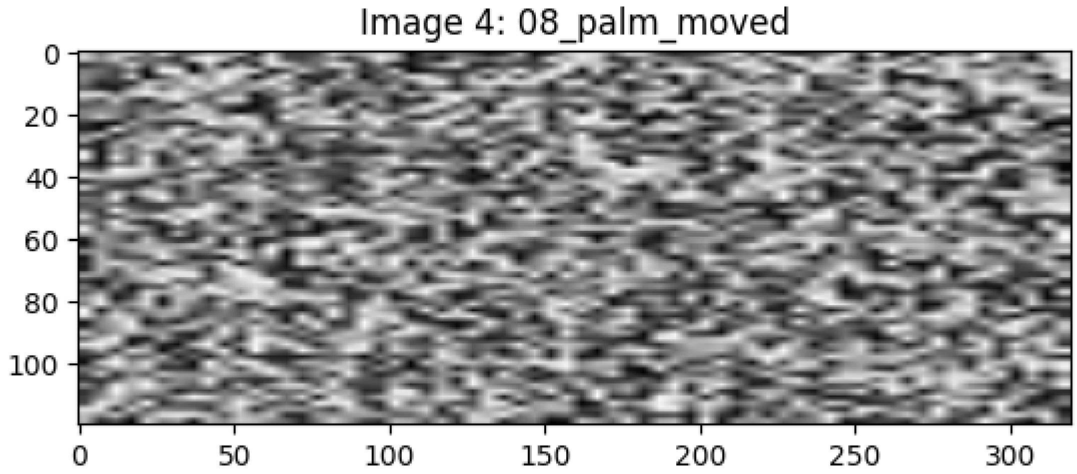
```
1/1 [==============================] - 0s 22ms/step
```



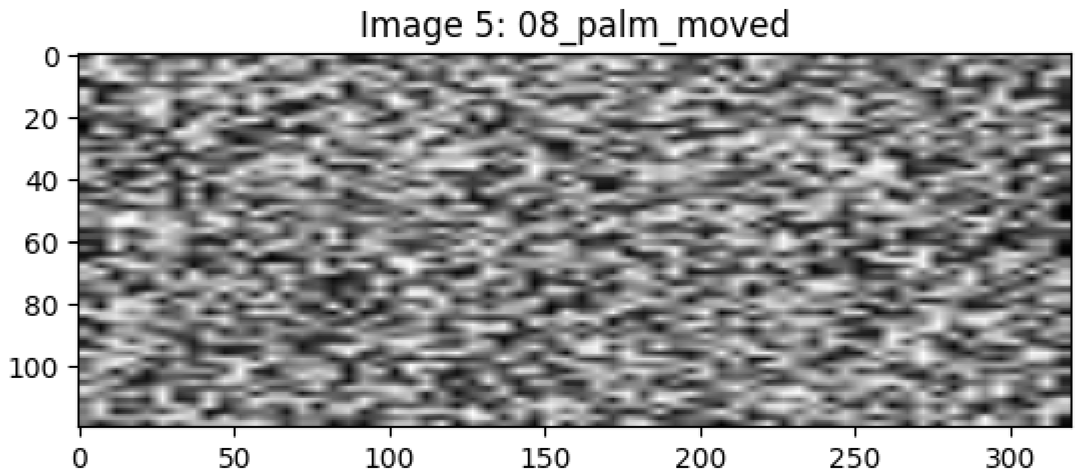
```
1/1 [==============================] - 0s 25ms/step
```



```
1/1 [==============================] - 0s 18ms/step
```



```
1/1 [==============================] - 0s 19ms/step
```



```
Predicted Gestures: ['08_palm_moved', '08_palm_moved', '08_palm_moved', '08_palm_moved', '08_palm_moved']
```

In [49]:

```
import matplotlib.pyplot as plt


folder_path = "D:\Gesture\leapGestRecog\leapGestRecog"


predicted_gestures = []

for filename in os.listdir(folder_path):
    if filename.endswith(".jpg") or filename.endswith(".png"):

        img_path = os.path.join(folder_path, filename)


        img = Image.open(img_path).convert('L')  # Convert to grayscale
        img = img.resize((320, 120))
        arr = np.array(img)
        t_test = arr.reshape((1, 120, 320, 1))
        t_test = t_test / 255.0
        plt.imshow(arr, cmap='gray')


        predictions = loaded_model.predict(t_test)


        predicted_class = np.argmax(predictions)
        predicted_gesture = reverselookup[predicted_class]
        predicted_gestures.append(predicted_gesture)
```

```python

print("Predicted Gestures:", predicted_gestures)
```

```
Predicted Gestures: []
```

In [50]:

```python
x_data.size, y_data.size, t_test.view
```

```
(768000000, 200000, <function ndarray.view>)
```

In [ ]: