In [1]:

```
# from google.colab import drive
```

In [2]:

```
# drive.mount('/content/drive/')
```

Import the Libraries

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf

import keras
from keras.models import Sequential, Model, load_model
from keras.applications import vgg16
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, Input,MaxP
ooling2D, InputLayer,UpSampling2D, Reshape,UpSampling1D
from keras.layers.normalization import BatchNormalization
from keras.layers.merge import concatenate
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator, load_img,img_to_array,
array_to_img
import pickle
import os
import cv2
import joblib
import math

from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from keras.utils import multi_gpu_model
from keras.callbacks import Callback
# from keras.utils.OneCycle import OneCycle

import keras.backend as K
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report, precision_recall_curve
import seaborn as sns
```

```
Using TensorFlow backend.
```

In [4]:

```
PATH="./depth_data_GD"
folders=sorted(os.listdir(PATH))[0:15]
test_folders=sorted(os.listdir(PATH))[15:17]
print(folders)
```

```
['basement_0001a_out', 'basement_0001b_out', 'bathroom_0001_out', 'b
athroom_0002_out', 'bathroom_0005_out', 'bathroom_0006_out', 'bathro
om_0007_out', 'bathroom_0010_out', 'bathroom_0011_out', 'bathroom_00
13_out', 'bathroom_0014a_out', 'bathroom_0016_out', 'bathroom_0019_o
ut', 'bathroom_0023_out', 'bathroom_0024_out']
```

Create training and Testing Data

In [6]:

```
training_data=[]
depth_data=[]
im_sze1=304
im_sze2=228
depth_sze1=55
depth_sze2=74



for folde in folders:
    #print(folde)
    im_list=sorted(os.listdir(os.path.join(PATH,folde)))
    #print(im_list)

    for img in im_list:
        #print(img)
        if 'jpg' in img:
            im_path=os.path.join(os.path.join(PATH,folde),img)
            #print(im_path)
            new_arr=cv2.imread(im_path)
            new_arr=cv2.resize(new_arr,(im_sze1,im_sze2))

            training_data.append(new_arr)
        elif 'png' in img:
            new_arr=cv2.imread(os.path.join(os.path.join(PATH,folde),img),0)
            new_arr=cv2.resize(new_arr,(depth_sze1,depth_sze2))
            depth_data.append(new_arr)
#             plt.imshow(new_arr)
#             plt.show()
```

In [7]:

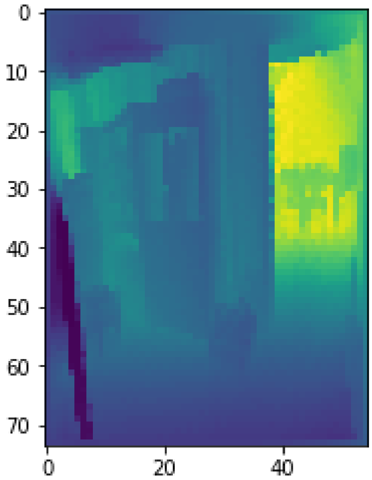
```
test_data=[]
test_depth_data=[]
im_sze1=304
im_sze2=228
depth_sze1=55
depth_sze2=74



for folde in test_folders:
    #print(folde)
    im_list=sorted(os.listdir(os.path.join(PATH,folde)))
    #print(im_list)

    for img in im_list:
        #print(img)
        if 'jpg' in img:
            im_path=os.path.join(os.path.join(PATH,folde),img)
            #print(im_path)
            new_arr=cv2.imread(im_path)
            new_arr=cv2.resize(new_arr,(im_sze1,im_sze2))

            test_data.append(new_arr)
        elif 'png' in img:
            new_arr=cv2.imread(os.path.join(os.path.join(PATH,folde),img),0)
            new_arr=cv2.resize(new_arr,(depth_sze1,depth_sze2))
            test_depth_data.append(new_arr)
#             plt.imshow(new_arr)
#             plt.show()
```

In [8]:

```
plt.imshow(depth_data[0])
```

Out[8]:

<matplotlib.image.AxesImage at 0x7f55d70acdd8>



In [10]:

```
depth_data_scaled=np.array(depth_data).reshape(-1,depth_sze1,depth_sze2,1).astyp
e('float32')/255
training_data_scaled=np.array(training_data).reshape(-1,im_sze2,im_sze1,3).astyp
e('float32')/255
```

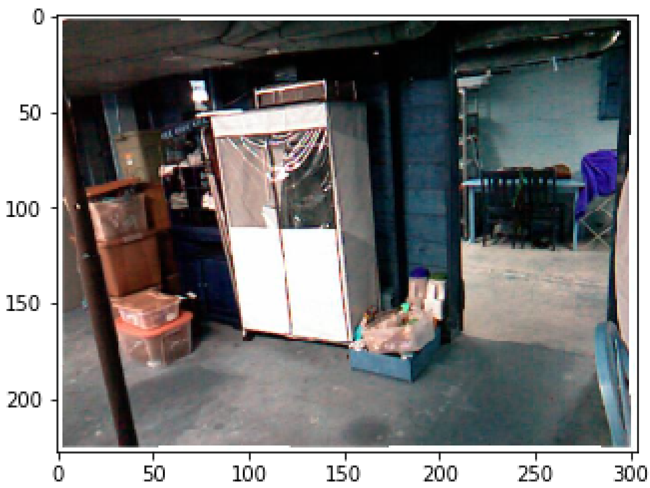
In [11]:

```
test_depth_data_scaled=np.array(test_depth_data).reshape(-1,depth_sze1,depth_sze
2,1).astype('float32')/255
test_data_scaled=np.array(test_data).reshape(-1,im_sze2,im_sze1,3).astype('float
32')/255
```

In [12]:

```
print(depth_data_scaled.shape)

# plt.imshow(np.reshape(depth_data[0],[depth_sze2,depth_sze1]))
# plt.show()

plt.imshow(training_data_scaled[0])
plt.show()
# plt.imshow(np.reshape(depth_data[0],[depth_sze2,depth_sze1]))
# plt.show()
```

```
(1476, 55, 74, 1)
```



Define Custom loss functions

In [13]:

```
def depth_loss(y_true, y_pred):

#     y_true=K.cast(y_true, dtype='float32')

#     y_pred=K.cast(y_pred, dtype='float64')
#     d=K.cast(K.log(y_pred) - K.log(y_true),dtype='float64')

#     log_diff=K.cast(K.sum(K.square(d))/(depth_sze1*depth_sze2),dtype='float6
4')
#     print('boo')
#     penalty=K.square(K.sum(d))/K.cast(K.square(depth_sze1*depth_sze2),dtype='f
loat64')

    y_true=K.cast(y_true, dtype='float32')

    y_pred=K.cast(y_pred, dtype='float32')

    lnYTrue = K.switch(K.equal(y_true, 0), K.zeros_like(y_true), K.log(y_true))
    #lnYPred = K.switch(K.equal(y_pred, 0), K.zeros_like(y_pred), K.log(y_pred))

    lnYPred=K.tf.where(K.tf.math.is_inf(y_pred), K.tf.zeros_like(y_pred), y_pred
)

    #d=K.cast(K.log(y_pred) - K.log(y_true),dtype='float32')
    d_arr=K.cast(lnYTrue - lnYPred,dtype='float32')

    #d_arr=K.get_value(d)
    #d_arr[d_arr==-np.inf]=0
    #d_arr[d_arr==np.inf]=0
    #bools=K.equal(d, -np.inf)

    #print(K.eval(bools))

    #print(K.int_shape(bools))

    #d_arr = K.switch(K.equal(K.log(y_pred) - K.log(y_true), 0), K.zeros_like(K.
log(y_pred) - K.log(y_true)), K.log(y_pred) - K.log(y_true))

    #has_inf = K.tf.constant([-np.inf, 1.,shape=(192,192,1)])

    #wh = K.tf.where(K.tf.equal(bools,True))

    #d_arr=K.tf.where(K.tf.math.is_inf(y_pred), K.tf.zeros_like(y_pred), d)
    #d_arr=K.tf.where(K.tf.is_nan(d_arr), K.tf.zeros_like(d_arr), d_arr)

    #print(K.eval(wh))
    #var = K.zeros(shape=(192, 192, 1))
    #print(d_arr)

    #new_d=K.variable(d_arr,dtype='float32')
    #new_d=d_arr


    log_diff=K.cast(K.sum(K.square(d_arr))/(depth_sze1*depth_sze2),dtype='float3
2')

    print('boo')

    penalty=K.square(K.sum(d_arr))/K.cast(K.square(depth_sze1*depth_sze2),dtype=
```

```
'float32')

    loss=log_diff-penalty
    #print(K.eval(loss))

    return loss
```

In [14]:

```
def depth_loss3(y_true, y_pred):

#     y_true=K.cast(y_true, dtype='float32')

#     y_pred=K.cast(y_pred, dtype='float64')
#     d=K.cast(K.log(y_pred) - K.log(y_true),dtype='float64')

#     log_diff=K.cast(K.sum(K.square(d))/(depth_sze1*depth_sze2),dtype='float6
4')
#     print('boo')
#     penalty=K.square(K.sum(d))/K.cast(K.square(depth_sze1*depth_sze2),dtype='f
loat64')

    y_true=K.cast(y_true, dtype='float32')

    y_pred=K.cast(y_pred, dtype='float32')

    #lnYTrue = K.switch(K.equal(y_true, 0), K.zeros_like(y_true), K.log(y_true))
    #lnYPred = K.switch(K.equal(y_pred, 0), K.zeros_like(y_pred), K.log(y_pred))

    lnYTrue =K.tf.where(K.tf.math.is_inf(y_true), K.tf.ones_like(y_true), y_true
)

    lnYPred=K.tf.where(K.tf.math.is_inf(y_pred), K.tf.ones_like(y_pred), y_pred)

    #print(K.eval(lnYPred))

    #d=K.cast(K.log(y_pred) - K.log(y_true),dtype='float32')
    d_arr=K.cast(lnYTrue - lnYPred,dtype='float32')


    #d_arr=K.get_value(d)
    #d_arr[d_arr==-np.inf]=0
    #d_arr[d_arr==np.inf]=0
    #bools=K.equal(d, -np.inf)

    #print(K.eval(bools))

    #print(K.int_shape(bools))

    #d_arr = K.switch(K.equal(K.log(y_pred) - K.log(y_true), 0), K.zeros_like(K.
log(y_pred) - K.log(y_true)), K.log(y_pred) - K.log(y_true))

    #has_inf = K.tf.constant([-np.inf, 1.,shape=(192,192,1)])

    #wh = K.tf.where(K.tf.equal(bools,True))

    #d_arr=K.tf.where(K.tf.math.is_inf(y_pred), K.tf.zeros_like(y_pred), d)
    #d_arr=K.tf.where(K.tf.is_nan(d_arr), K.tf.zeros_like(d_arr), d_arr)

    #print(K.eval(wh))
    #var = K.zeros(shape=(192, 192, 1))
    #print(d_arr)

    #new_d=K.variable(d_arr,dtype='float32')
    #new_d=d_arr


    log_diff=K.cast(K.sum(K.square(d_arr))/(depth_sze1*depth_sze2),dtype='float3
```

```
2')

    print('boo')

    penalty=K.square(K.sum(d_arr))/K.cast(K.square(depth_sze1*depth_sze2),dtype=
'float32')

    loss=log_diff+penalty
    #print(K.eval(loss))

    return loss
```

In [15]:

```
def depth_loss2(y_true, y_pred):


#     y_true=K.cast(y_true, dtype='float32')

#     y_pred=K.cast(y_pred, dtype='float64')
#     d=K.cast(K.log(y_pred) - K.log(y_true),dtype='float64')

#     log_diff=K.cast(K.sum(K.square(d))/(depth_sze1*depth_sze2),dtype='float6
4')
#     print('boo')
#     penalty=K.square(K.sum(d))/K.cast(K.square(depth_sze1*depth_sze2),dtype='f
loat64')

    y_true=K.cast(y_true, dtype='float32')

    y_pred=K.cast(y_pred, dtype='float32')
    d=K.cast((y_pred - y_true),dtype='float32')


    #d_arr=K.get_value(d)
    #d_arr[d_arr==-np.inf]=0
    #d_arr[d_arr==np.inf]=0
    bools=K.equal(d, -np.inf)

    #print(K.eval(bools))

    #print(K.int_shape(bools))
    #has_inf = K.tf.constant([-np.inf, 1.,shape=(192,192,1)])
    #wh = K.tf.where(K.tf.equal(bools,True))

    #lnYTrue = K.switch(KB.equal(d, 0), KB.zeros_like(d), KB.log(d))
    #d_arr = K.switch(KB.equal(d, 0), KB.zeros_like(d), KB.log(d))
    #d_arr=K.tf.where(K.tf.math.is_inf(d), K.tf.zeros_like(d), d)

    #d_arr=K.tf.where(K.tf.is_nan(d_arr), K.tf.zeros_like(d_arr), d_arr)

    #print(K.eval(wh))
    #var = K.zeros(shape=(192, 192, 1))
    #print(d_arr)

    #new_d=K.variable(d_arr,dtype='float32')
    #new_d=d_arr

    log_diff=K.cast(K.sum(K.square(d_arr))/(depth_sze1*depth_sze2),dtype='float3
2')

    print('boo')

    penalty=K.square(K.sum(d_arr))/K.cast(K.square(depth_sze1*depth_sze2),dtype=
'float32')

    loss=log_diff-penalty
    #print(K.eval(loss))

    return loss
```

Testing the created loss function

In [16]:

```
y_true = np.random.rand(57,76)
y_pred = np.random.rand(57,76)
y_pred.shape,y_true.shape


print(K.eval(depth_loss3(y_true, y_pred)))
```

```
boo
0.17396288
```

Making the Coarse Sequential Model

In [36]:

```
model=Sequential()

model.add(Conv2D(96,(11,11),strides=(4,4),input_shape=training_data_scaled.shape
[1:],padding='same'))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(256,(5,5),padding='same'))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(384,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation("relu"))
# model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(384,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation("relu"))
# model.add(MaxPooling2D(pool_size=(2,2)))

# model.add(Conv2D(256,(3,3),strides=(1,1),padding='same'))
# # model.add(BatchNormalization())
# model.add(Activation("relu"))
# model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
# model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(4096))
model.add(BatchNormalization())
model.add(Activation("linear"))
model.add(Dropout(0.4))

model.add(Reshape((64, 64,1)))

model.add(UpSampling2D(size=(2,2)))
model.add(Conv2D(1,(74,55),padding='valid'))
model.add(BatchNormalization())
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_9 (Conv2D)            (None, 57, 76, 96)        34944     
_________________________________________________________________
batch_normalization_11 (Batc (None, 57, 76, 96)        384       
_________________________________________________________________
activation_1 (Activation)    (None, 57, 76, 96)        0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 28, 38, 96)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 38, 256)       614656    
_________________________________________________________________
batch_normalization_12 (Batc (None, 28, 38, 256)       1024      
_________________________________________________________________
activation_2 (Activation)    (None, 28, 38, 256)       0         
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 19, 256)       0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 19, 384)       885120    
_________________________________________________________________
batch_normalization_13 (Batc (None, 14, 19, 384)       1536      
_________________________________________________________________
activation_3 (Activation)    (None, 14, 19, 384)       0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 19, 384)       1327488   
_________________________________________________________________
batch_normalization_14 (Batc (None, 14, 19, 384)       1536      
_________________________________________________________________
activation_4 (Activation)    (None, 14, 19, 384)       0         
_________________________________________________________________
dense_3 (Dense)              (None, 14, 19, 256)       98560     
_________________________________________________________________
batch_normalization_15 (Batc (None, 14, 19, 256)       1024      
_________________________________________________________________
activation_5 (Activation)    (None, 14, 19, 256)       0         
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 7, 9, 256)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 16128)             0         
_________________________________________________________________
dense_4 (Dense)              (None, 4096)              66064384  
_________________________________________________________________
batch_normalization_16 (Batc (None, 4096)              16384     
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0         
_________________________________________________________________
reshape_2 (Reshape)          (None, 64, 64, 1)         0         
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 128, 128, 1)       0         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 55, 74, 1)         4071      
_________________________________________________________________
batch_normalization_17 (Batc (None, 55, 74, 1)         4         
=================================================================
Total params: 69,051,115
Trainable params: 69,040,169
Non-trainable params: 10,946
_________________________________________________________________
```

In [37]:

```
parallel_model=multi_gpu_model(model,gpus=2)

parallel_model.compile(loss=depth_loss3, optimizer=optimizers.RMSprop(lr=1e-3),m
etrics=[depth_loss3])
histroy=parallel_model.fit(training_data_scaled,depth_data_scaled,batch_size=32,
epochs=20, validation_split=0.1)
```

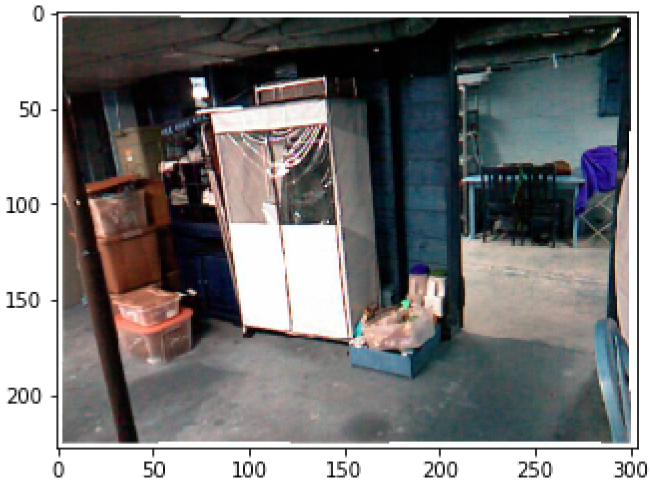
```
boo
boo
Train on 1328 samples, validate on 148 samples
Epoch 1/20
1328/1328 [==============================] - 16s 12ms/step - loss: 5
8.4019 - depth_loss3: 58.4019 - val_loss: 175.0677 - val_depth_loss
3: 175.0677
Epoch 2/20
1328/1328 [==============================] - 8s 6ms/step - loss: 41.
7253 - depth_loss3: 41.7253 - val_loss: 237.2015 - val_depth_loss3:
237.2015
Epoch 3/20
1328/1328 [==============================] - 7s 5ms/step - loss: 30.
2943 - depth_loss3: 30.2943 - val_loss: 224.6244 - val_depth_loss3:
224.6244
Epoch 4/20
1328/1328 [==============================] - 7s 5ms/step - loss: 22.
3744 - depth_loss3: 22.3744 - val_loss: 219.0942 - val_depth_loss3:
219.0942
Epoch 5/20
1328/1328 [==============================] - 7s 5ms/step - loss: 17.
4046 - depth_loss3: 17.4046 - val_loss: 52.5806 - val_depth_loss3: 5
2.5806
Epoch 6/20
1328/1328 [==============================] - 7s 5ms/step - loss: 14.
8895 - depth_loss3: 14.8895 - val_loss: 67.6807 - val_depth_loss3: 6
7.6807
Epoch 7/20
1328/1328 [==============================] - 7s 5ms/step - loss: 13.
1273 - depth_loss3: 13.1273 - val_loss: 40.2175 - val_depth_loss3: 4
0.2175
Epoch 8/20
1328/1328 [==============================] - 7s 5ms/step - loss: 11.
5047 - depth_loss3: 11.5047 - val_loss: 26.5950 - val_depth_loss3: 2
6.5950
Epoch 9/20
1328/1328 [==============================] - 7s 5ms/step - loss: 10.
0043 - depth_loss3: 10.0043 - val_loss: 32.8873 - val_depth_loss3: 3
2.8873
Epoch 10/20
1328/1328 [==============================] - 7s 5ms/step - loss: 8.5
669 - depth_loss3: 8.5669 - val_loss: 33.8323 - val_depth_loss3: 33.
8323
Epoch 11/20
1328/1328 [==============================] - 7s 5ms/step - loss: 7.2
573 - depth_loss3: 7.2573 - val_loss: 30.7151 - val_depth_loss3: 30.
7151
Epoch 12/20
1328/1328 [==============================] - 7s 5ms/step - loss: 6.0
811 - depth_loss3: 6.0811 - val_loss: 22.9839 - val_depth_loss3: 22.
9839
Epoch 13/20
1328/1328 [==============================] - 7s 5ms/step - loss: 4.9
593 - depth_loss3: 4.9593 - val_loss: 16.7523 - val_depth_loss3: 16.
7523
Epoch 14/20
1328/1328 [==============================] - 7s 5ms/step - loss: 4.0
516 - depth_loss3: 4.0516 - val_loss: 8.0452 - val_depth_loss3: 8.04
52
Epoch 15/20
1328/1328 [==============================] - 7s 5ms/step - loss: 3.1
```

```
559 - depth_loss3: 3.1559 - val_loss: 11.4017 - val_depth_loss3: 11.
4017
Epoch 16/20
1328/1328 [==============================] - 7s 5ms/step - loss: 2.4
386 - depth_loss3: 2.4386 - val_loss: 7.8825 - val_depth_loss3: 7.88
25
Epoch 17/20
1328/1328 [==============================] - 7s 5ms/step - loss: 1.7
644 - depth_loss3: 1.7644 - val_loss: 3.5889 - val_depth_loss3: 3.58
89
Epoch 18/20
1328/1328 [==============================] - 7s 5ms/step - loss: 1.2
548 - depth_loss3: 1.2548 - val_loss: 2.9372 - val_depth_loss3: 2.93
72
Epoch 19/20
1328/1328 [==============================] - 7s 5ms/step - loss: 0.8
685 - depth_loss3: 0.8685 - val_loss: 2.4765 - val_depth_loss3: 2.47
65
Epoch 20/20
1328/1328 [==============================] - 7s 5ms/step - loss: 0.5
862 - depth_loss3: 0.5862 - val_loss: 1.1630 - val_depth_loss3: 1.16
30
```

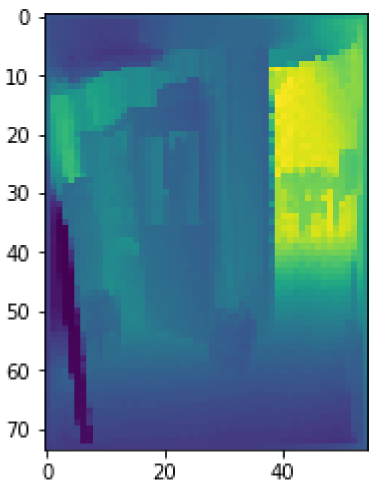
Actual Images

In [41]:

```
# print(training_data[0,:].shape)
# print(np.reshape(training_data[0,:],[im_sze1,im_sze2,3]).shape)
plt.imshow(training_data[0])
plt.show()

print(np.reshape(depth_data[0],[depth_sze1,depth_sze2]).shape)
# plt.imshow(np.reshape(depth_data[0],[depth_sze2,depth_sze1]))
plt.imshow(depth_data[0])

plt.show()
```



(55, 74)



In [46]:

```
y_pred_prob=parallel_model.predict(np.reshape(training_data[0]/255,[-1,im_sze2,i
m_sze1,3]))
y_pred_prob.shape
np.reshape(y_pred_prob,[depth_sze2,depth_sze1]).shape
```

Out[46]:

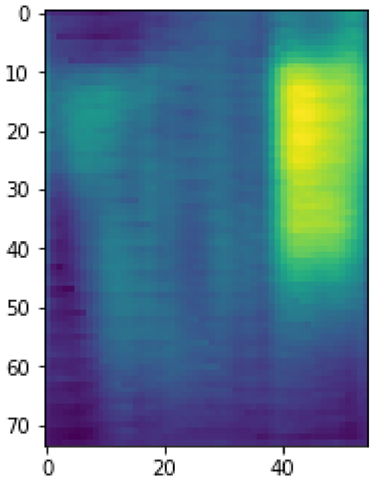
(74, 55)

Predicted Coarse Image

In [47]:

```
print(np.reshape(y_pred_prob*255,[depth_sze2,depth_sze1]).shape)
plt.imshow(np.reshape(y_pred_prob*255,[depth_sze2,depth_sze1]))
plt.show()
```

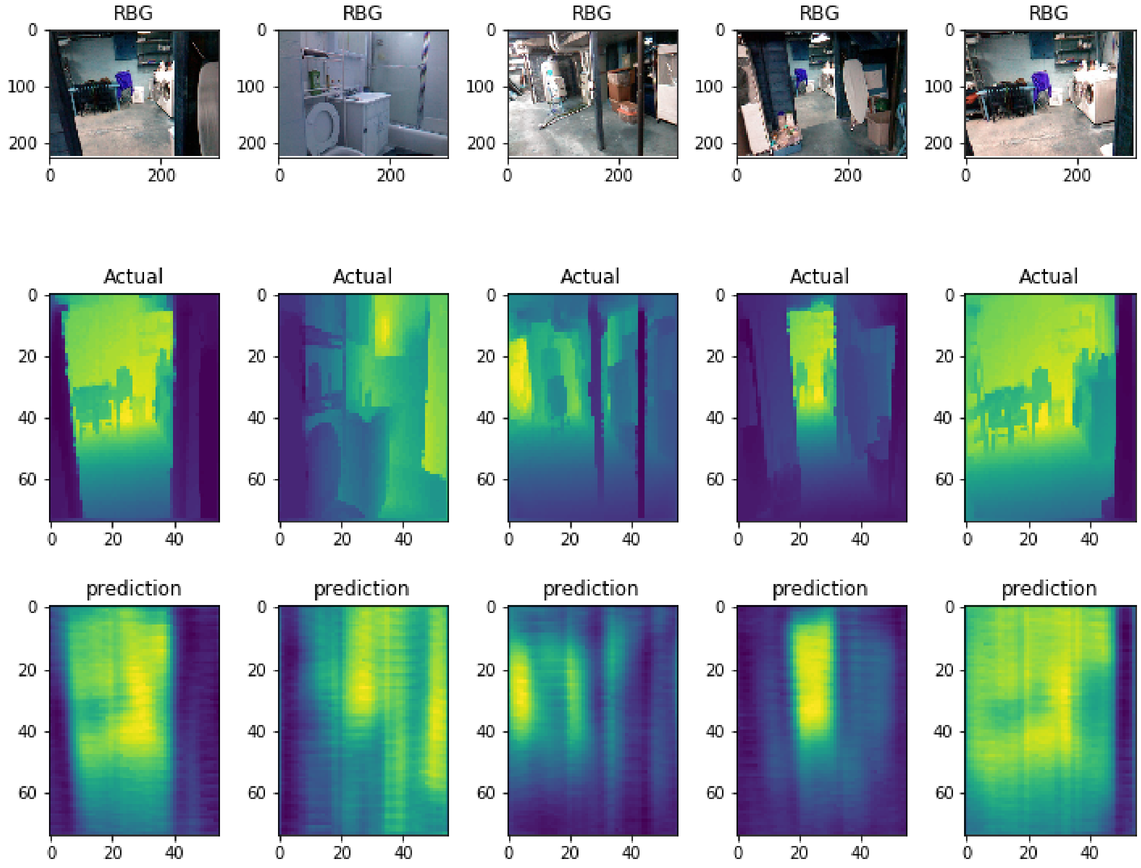
(74, 55)



Sample Prediction on Training Dataset

In [48]:

```
f = plt.figure(figsize=(10,8))
i=1
num_select=5

select=np.random.choice(len(training_data), num_select, replace=False)
for i in range(1,num_select+1):

    ax1=plt.subplot(3, num_select, i)
    ax1.imshow(training_data[select[i-1]])
    ax1.set_title('RBG')

    ax2=plt.subplot(3, num_select, i+num_select)
    ax2.imshow(depth_data[select[i-1]])
    ax2.set_title('Actual')

    y_pred_prob=parallel_model.predict(np.reshape(training_data[select[i-1]]/255
,[-1,im_sze2,im_sze1,3]))
    ax3=plt.subplot(3, num_select, i+2*num_select)
    ax3.imshow(np.reshape(y_pred_prob,[depth_sze2,depth_sze1]))
    ax3.set_title('prediction')
    #i=i+1
plt.tight_layout()
plt.show()
```



Coarse Functional Model

In [49]:

```
first_layer=Input(training_data_scaled.shape[1:])
conv1=Conv2D(96,(11,11),strides=(4,4),activation='relu',padding='same')(first_la
yer)
b1=BatchNormalization()(conv1)
p1=MaxPooling2D(pool_size=(2,2))(b1)

conv2=Conv2D(256,(5,5),activation='relu',padding='same')(p1)
b2=BatchNormalization()(conv2)
p2=MaxPooling2D(pool_size=(2,2))(b2)

conv3=Conv2D(384,(3,3),activation='relu',padding='same')(p2)
b3=BatchNormalization()(conv3)

conv4=Conv2D(384,(3,3),activation='relu',padding='same')(b3)
b4=BatchNormalization()(conv4)

Dlayer1=Dense(256,activation='relu')(b4)
b5=BatchNormalization()(Dlayer1)
p5=MaxPooling2D(pool_size=(2,2))(b5)

flat=Flatten()(p5)
flat=Dense(4096,activation='linear')(flat)
flat=BatchNormalization()(flat)
flat=Dropout(0.4)(flat)

mat=Reshape((64,64,1))(flat)

upsamp=UpSampling2D((2,2))(mat)

out1=Conv2D(1,(74,55))(upsamp)
out1=BatchNormalization()(out1)
# model.add(Flatten())
# model.add(Dense(4096))
# model.add(BatchNormalization())
# model.add(Activation("linear"))
# model.add(Dropout(0.4))

# model.add(Reshape((64, 64,1)))

# model.add(UpSampling2D(size=(2,2)))
# model.add(Conv2D(1,(74,55),padding='valid'))
model1=Model(inputs=first_layer,outputs=out1)
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         (None, 228, 304, 3)       0         
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 57, 76, 96)        34944     
_________________________________________________________________
batch_normalization_18 (Batc (None, 57, 76, 96)        384       
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 28, 38, 96)        0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 28, 38, 256)       614656    
_________________________________________________________________
batch_normalization_19 (Batc (None, 28, 38, 256)       1024      
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 14, 19, 256)       0         
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 14, 19, 384)       885120    
_________________________________________________________________
batch_normalization_20 (Batc (None, 14, 19, 384)       1536      
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 14, 19, 384)       1327488   
_________________________________________________________________
batch_normalization_21 (Batc (None, 14, 19, 384)       1536      
_________________________________________________________________
dense_5 (Dense)              (None, 14, 19, 256)       98560     
_________________________________________________________________
batch_normalization_22 (Batc (None, 14, 19, 256)       1024      
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 9, 256)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 16128)             0         
_________________________________________________________________
dense_6 (Dense)              (None, 4096)              66064384  
_________________________________________________________________
batch_normalization_23 (Batc (None, 4096)              16384     
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0         
_________________________________________________________________
reshape_3 (Reshape)          (None, 64, 64, 1)         0         
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 128, 128, 1)       0         
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 55, 74, 1)         4071      
_________________________________________________________________
batch_normalization_24 (Batc (None, 55, 74, 1)         4         
=================================================================
Total params: 69,051,115
Trainable params: 69,040,169
Non-trainable params: 10,946
_________________________________________________________________
```

In [50]:

```
# print(depth_data_scaled[0][:,:,0].shape)
# plt.imshow(depth_data_scaled[0][:,:,0])

model1.output
```

Out[50]:

```
<tf.Tensor 'batch_normalization_24/cond/Merge:0' shape=(?, 55, 74,
1) dtype=float32>
```

In [51]:

```
parallel_model2=multi_gpu_model(model1,gpus=2)

parallel_model2.compile(loss=depth_loss3, optimizer=optimizers.RMSprop(lr=1e-3),
metrics=[depth_loss3])
histroy=parallel_model2.fit(training_data_scaled,depth_data_scaled,batch_size=32
,epochs=30, validation_split=0.1)
```

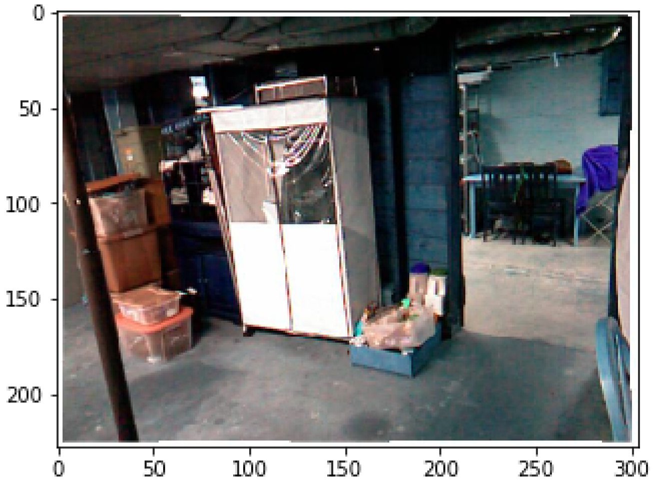
```
boo
boo
Train on 1328 samples, validate on 148 samples
Epoch 1/30
1328/1328 [==============================] - 12s 9ms/step - loss: 5
8.5409 - depth_loss3: 58.5409 - val_loss: 339.6501 - val_depth_loss
3: 339.6501
Epoch 2/30
1328/1328 [==============================] - 7s 5ms/step - loss: 41.
8850 - depth_loss3: 41.8850 - val_loss: 749.4695 - val_depth_loss3:
749.4695
Epoch 3/30
1328/1328 [==============================] - 7s 5ms/step - loss: 30.
1996 - depth_loss3: 30.1996 - val_loss: 135.5185 - val_depth_loss3:
135.5185
Epoch 4/30
1328/1328 [==============================] - 7s 5ms/step - loss: 22.
3063 - depth_loss3: 22.3063 - val_loss: 186.8483 - val_depth_loss3:
186.8483
Epoch 5/30
1328/1328 [==============================] - 7s 5ms/step - loss: 17.
4000 - depth_loss3: 17.4000 - val_loss: 148.7182 - val_depth_loss3:
148.7182
Epoch 6/30
1328/1328 [==============================] - 7s 5ms/step - loss: 14.
9231 - depth_loss3: 14.9231 - val_loss: 78.6712 - val_depth_loss3: 7
8.6712
Epoch 7/30
1328/1328 [==============================] - 7s 5ms/step - loss: 13.
0971 - depth_loss3: 13.0971 - val_loss: 47.3203 - val_depth_loss3: 4
7.3203
Epoch 8/30
1328/1328 [==============================] - 7s 5ms/step - loss: 11.
4683 - depth_loss3: 11.4683 - val_loss: 21.4715 - val_depth_loss3: 2
1.4715
Epoch 9/30
1328/1328 [==============================] - 7s 5ms/step - loss: 9.9
608 - depth_loss3: 9.9608 - val_loss: 39.4835 - val_depth_loss3: 39.
4835
Epoch 10/30
1328/1328 [==============================] - 7s 5ms/step - loss: 8.5
458 - depth_loss3: 8.5458 - val_loss: 25.8436 - val_depth_loss3: 25.
8436
Epoch 11/30
1328/1328 [==============================] - 7s 5ms/step - loss: 7.1
934 - depth_loss3: 7.1934 - val_loss: 23.0582 - val_depth_loss3: 23.
0582
Epoch 12/30
1328/1328 [==============================] - 7s 5ms/step - loss: 6.1
131 - depth_loss3: 6.1131 - val_loss: 13.9052 - val_depth_loss3: 13.
9052
Epoch 13/30
1328/1328 [==============================] - 7s 5ms/step - loss: 5.0
152 - depth_loss3: 5.0152 - val_loss: 10.4638 - val_depth_loss3: 10.
4638
Epoch 14/30
1328/1328 [==============================] - 7s 5ms/step - loss: 4.0
058 - depth_loss3: 4.0058 - val_loss: 10.9431 - val_depth_loss3: 10.
9431
Epoch 15/30
1328/1328 [==============================] - 7s 5ms/step - loss: 3.2
```

```
428 - depth_loss3: 3.2428 - val_loss: 4.7427 - val_depth_loss3: 4.74
27
Epoch 16/30
1328/1328 [==============================] - 7s 5ms/step - loss: 2.5
341 - depth_loss3: 2.5341 - val_loss: 4.9989 - val_depth_loss3: 4.99
89
Epoch 17/30
1328/1328 [==============================] - 7s 5ms/step - loss: 1.7
839 - depth_loss3: 1.7839 - val_loss: 4.3731 - val_depth_loss3: 4.37
31
Epoch 18/30
1328/1328 [==============================] - 7s 5ms/step - loss: 1.2
965 - depth_loss3: 1.2965 - val_loss: 1.6622 - val_depth_loss3: 1.66
22
Epoch 19/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.8
706 - depth_loss3: 0.8706 - val_loss: 2.0731 - val_depth_loss3: 2.07
31
Epoch 20/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.5
090 - depth_loss3: 0.5090 - val_loss: 1.2761 - val_depth_loss3: 1.27
61
Epoch 21/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.2
807 - depth_loss3: 0.2807 - val_loss: 0.7553 - val_depth_loss3: 0.75
53
Epoch 22/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
825 - depth_loss3: 0.1825 - val_loss: 0.5307 - val_depth_loss3: 0.53
07
Epoch 23/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
959 - depth_loss3: 0.1959 - val_loss: 0.4980 - val_depth_loss3: 0.49
80
Epoch 24/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
280 - depth_loss3: 0.1280 - val_loss: 0.3572 - val_depth_loss3: 0.35
72
Epoch 25/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
569 - depth_loss3: 0.1569 - val_loss: 0.4036 - val_depth_loss3: 0.40
36
Epoch 26/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
655 - depth_loss3: 0.1655 - val_loss: 0.4572 - val_depth_loss3: 0.45
72
Epoch 27/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
582 - depth_loss3: 0.1582 - val_loss: 0.3631 - val_depth_loss3: 0.36
31
Epoch 28/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
597 - depth_loss3: 0.1597 - val_loss: 0.4569 - val_depth_loss3: 0.45
69
Epoch 29/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
576 - depth_loss3: 0.1576 - val_loss: 0.4310 - val_depth_loss3: 0.43
10
Epoch 30/30
1328/1328 [==============================] - 7s 5ms/step - loss: 0.1
```

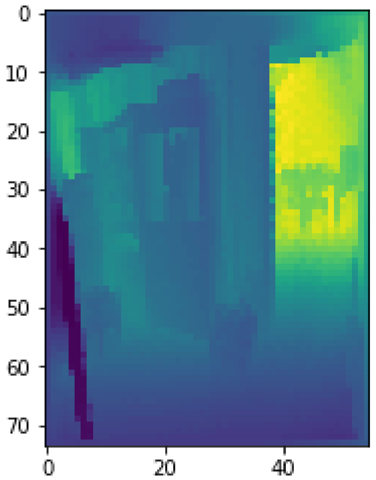
```
995 - depth_loss3: 0.1995 - val_loss: 0.3763 - val_depth_loss3: 0.37
63
```

In [56]:

```
# print(training_data[0,:].shape)
# print(np.reshape(training_data[0,:],[im_sze1,im_sze2,3]).shape)
plt.imshow(training_data[0])
plt.show()


print(np.reshape(depth_data[0],[depth_sze1,depth_sze2]).shape)
plt.imshow(np.reshape(depth_data[0],[depth_sze2,depth_sze1]))
plt.show()
```



```
(55, 74)
```

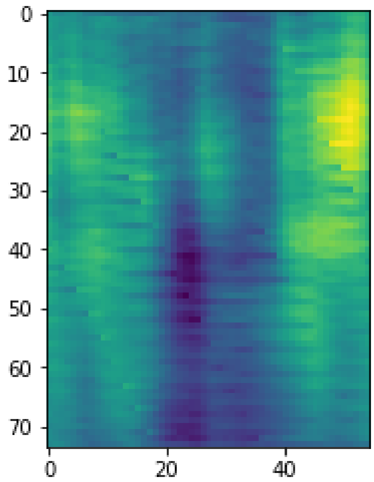


In [57]:

```
y_pred_prob=parallel_model2.predict(np.reshape(training_data[0],[-1,im_sze2,im_s
ze1,3]))
```

In [58]:

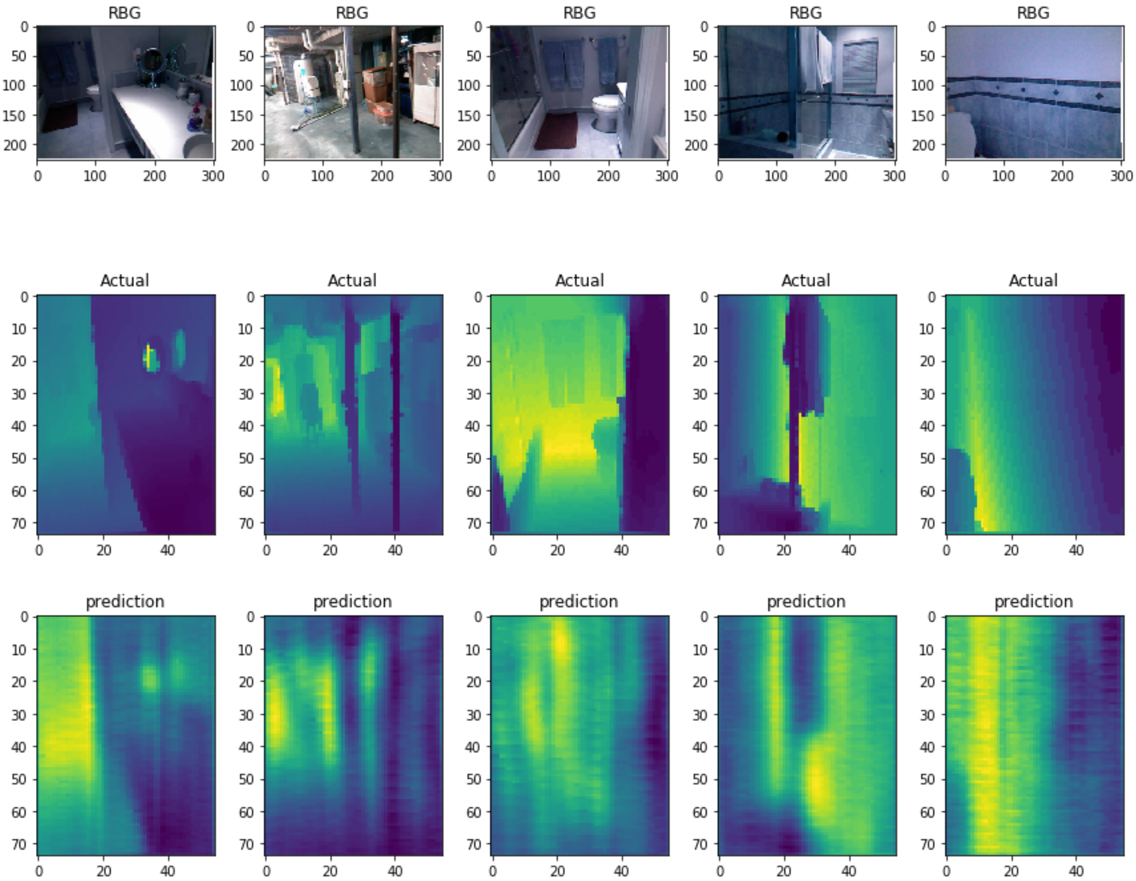
```
plt.imshow(np.reshape(y_pred_prob*255,[depth_sze2,depth_sze1]))
plt.show()
```



Sample Prediction on random examples from Training Dataset

In [59]:

```
f = plt.figure(figsize=(12,10))
i=1
num_select=5
select=np.random.choice(len(training_data), num_select, replace=False)
for i in range(1,num_select+1):

    ax1=plt.subplot(3, num_select, i)
    ax1.imshow(training_data[select[i-1]])
    ax1.set_title('RBG')

    ax2=plt.subplot(3, num_select, i+num_select)
    ax2.imshow(depth_data[select[i-1]])
    ax2.set_title('Actual')

    y_pred_prob=parallel_model2.predict(np.reshape(training_data[select[i-1]]/25
5,[-1,im_sze2,im_sze1,3]))
    ax3=plt.subplot(3, num_select, i+2*num_select)
    ax3.imshow(np.reshape(y_pred_prob,[depth_sze2,depth_sze1]))
    ax3.set_title('prediction')
    #i=i+1
plt.tight_layout()
plt.show()
```



In [ ]:

```
plt.imshow(training_data[select[1]])
```

Making the Full Model with Coarse + Fine Tune layers

In [18]:

```
second_layer=Input(training_data_scaled.shape[1:])
conv21=Conv2D(63,(9,9),strides=(2,2),padding='valid')(first_layer)
b21=BatchNormalization()(conv21)
p21=MaxPooling2D(pool_size=(2,2))(b21)


Concat=concatenate([out1, p21])
# print(type(Concat),type(p21))
conv22=Conv2D(64,(5,5),padding='same')(Concat)
b22=BatchNormalization()(conv22)

out=Conv2D(1,(5,5),padding='same')(b22)
out=BatchNormalization()(out)



final_model= Model(inputs=first_layer,outputs=out)
final_model.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 228, 304, 3)  0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 57, 76, 96)   34944       input_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 57, 76, 96)   384         conv2d_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 28, 38, 96)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 28, 38, 256)  614656      max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 28, 38, 256)  1024        conv2d_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 14, 19, 256)  0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 14, 19, 384)  885120      max_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 14, 19, 384)  1536        conv2d_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 14, 19, 384)  1327488     batch_normalization_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 14, 19, 384)  1536        conv2d_4[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 14, 19, 256)  98560       batch_normalization_4[0][0]
__________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 14, 19, 256)  1024        dense_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 7, 9, 256)    0           batch_normalization_5[0][0]
__________________________________________________________________________________________________
```

```
flatten_1 (Flatten)             (None, 16128)        0           max
_pooling2d_3[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 4096)         66064384    fla
tten_1[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 4096)         16384       den
se_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 4096)         0           bat
ch_normalization_6[0][0]
__________________________________________________________________________________________________
reshape_1 (Reshape)             (None, 64, 64, 1)    0           dro
pout_1[0][0]
__________________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D)  (None, 128, 128, 1)  0           res
hape_1[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 110, 148, 63) 15372       inp
ut_1[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 55, 74, 1)    4071        up_
sampling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 110, 148, 63) 252         con
v2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 55, 74, 1)    4           con
v2d_5[0][0]
__________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 55, 74, 63)   0           bat
ch_normalization_8[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 55, 74, 64)   0           bat
ch_normalization_7[0][0]
                                                                 max
_pooling2d_4[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 55, 74, 64)   102464      con
catenate_1[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 55, 74, 64)   256         con
v2d_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 55, 74, 1)    1601        bat
ch_normalization_9[0][0]
__________________________________________________________________________________________________
```

```
________________________________________________________________________________
batch_normalization_10 (BatchNo (None, 55, 74, 1)    4           con
v2d_8[0][0]
=================================================================
==============================
Total params: 69,171,064
Trainable params: 69,159,862
Non-trainable params: 11,202
_________________________________________________________________
______________________________
```

In [19]:

```
# model2=Sequential()

# model2.add(Conv2D(63,(9,9),strides=(2,2),input_shape=training_data_scaled.shap
e[1:],padding='valid'))
# model2.add(BatchNormalization())
# model2.add(Activation("relu"))
# model2.add(MaxPooling2D(pool_size=(2,2)))

# model2.summary()
# Concat=concatenate([model.output, model2.output])
# final_model= Model(inputs=[model.input,model2.input],outputs=Concat)


# newlayer=Input((55,74,64))
# conv1=Conv2D(64,kernel_size=2,activation='relu')(newlayer)
# conv1=BatchNormalization()(conv1)
# # conv2=Conv2D(64,kernel_size=5,activation='relu')(conv1)

# output=Conv2D(1,kernel_size=2,activation='linear')(conv1)
# # model.add(UpSampling2D(size=(2,2)))
# # model.add(Conv2D(1,(72,53),padding='valid'))

# fin=Model(inputs=model.input, outputs=output)
# """
# CONCATENATION
# model.add(Conv2D(64,(5,5),padding='same'))
# model.add(BatchNormalization())
# model.add(Activation("relu"))
# model.add(MaxPooling2D(pool_size=(2,2)))
# """


# output.summary()
```

In [20]:

```
parallel_model3=multi_gpu_model(final_model,gpus=3)

parallel_model3.compile(loss=depth_loss3, optimizer=optimizers.RMSprop(lr=1e-3),
metrics=[depth_loss3])
histroy=parallel_model3.fit(training_data_scaled,depth_data_scaled,batch_size=32
,epochs=30, validation_split=0.1)
```

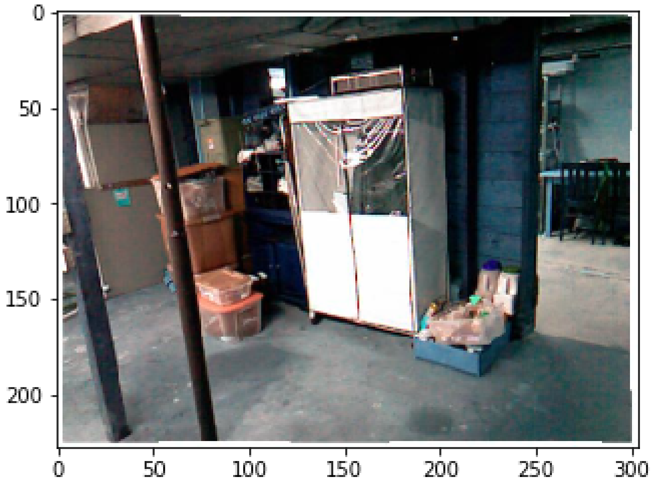
```
boo
boo
WARNING:tensorflow:From /apps/tensorflow/venv/lib/python3.6/site-pac
kages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorf
low.python.ops.math_ops) is deprecated and will be removed in a futu
re version.
Instructions for updating:
Use tf.cast instead.
Train on 1328 samples, validate on 148 samples
Epoch 1/30
1328/1328 [==============================] - 30s 22ms/step - loss: 5
8.6774 - depth_loss3: 58.6774 - val_loss: 2192.5896 - val_depth_loss
3: 2192.5896
Epoch 2/30
1328/1328 [==============================] - 12s 9ms/step - loss: 4
2.2030 - depth_loss3: 42.2030 - val_loss: 91.6935 - val_depth_loss3:
91.6935
Epoch 3/30
1328/1328 [==============================] - 12s 9ms/step - loss: 3
0.5563 - depth_loss3: 30.5563 - val_loss: 536.9134 - val_depth_loss
3: 536.9134
Epoch 4/30
1328/1328 [==============================] - 12s 9ms/step - loss: 2
2.4701 - depth_loss3: 22.4701 - val_loss: 61.6223 - val_depth_loss3:
61.6223
Epoch 5/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1
7.6318 - depth_loss3: 17.6318 - val_loss: 76.9335 - val_depth_loss3:
76.9335
Epoch 6/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1
5.1233 - depth_loss3: 15.1233 - val_loss: 79.3674 - val_depth_loss3:
79.3674
Epoch 7/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1
3.2728 - depth_loss3: 13.2728 - val_loss: 26.5768 - val_depth_loss3:
26.5768
Epoch 8/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1
1.6530 - depth_loss3: 11.6530 - val_loss: 40.1768 - val_depth_loss3:
40.1768
Epoch 9/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1
0.0743 - depth_loss3: 10.0743 - val_loss: 24.8131 - val_depth_loss3:
24.8131
Epoch 10/30
1328/1328 [==============================] - 12s 9ms/step - loss: 8.
6776 - depth_loss3: 8.6776 - val_loss: 26.3415 - val_depth_loss3: 2
6.3415
Epoch 11/30
1328/1328 [==============================] - 12s 9ms/step - loss: 7.
3528 - depth_loss3: 7.3528 - val_loss: 10.8918 - val_depth_loss3: 1
0.8918
Epoch 12/30
1328/1328 [==============================] - 12s 9ms/step - loss: 6.
1478 - depth_loss3: 6.1478 - val_loss: 19.0009 - val_depth_loss3: 1
9.0009
Epoch 13/30
1328/1328 [==============================] - 12s 9ms/step - loss: 5.
1529 - depth_loss3: 5.1529 - val_loss: 9.3052 - val_depth_loss3: 9.3
052
```

```
Epoch 14/30
1328/1328 [==============================] - 12s 9ms/step - loss: 4.
1487 - depth_loss3: 4.1487 - val_loss: 10.9072 - val_depth_loss3: 1
0.9072
Epoch 15/30
1328/1328 [==============================] - 12s 9ms/step - loss: 3.
2989 - depth_loss3: 3.2989 - val_loss: 5.2829 - val_depth_loss3: 5.2
829
Epoch 16/30
1328/1328 [==============================] - 12s 9ms/step - loss: 2.
4638 - depth_loss3: 2.4638 - val_loss: 6.2895 - val_depth_loss3: 6.2
895
Epoch 17/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1.
8874 - depth_loss3: 1.8874 - val_loss: 3.4222 - val_depth_loss3: 3.4
222
Epoch 18/30
1328/1328 [==============================] - 12s 9ms/step - loss: 1.
2783 - depth_loss3: 1.2783 - val_loss: 1.6980 - val_depth_loss3: 1.6
980
Epoch 19/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
9355 - depth_loss3: 0.9355 - val_loss: 1.8047 - val_depth_loss3: 1.8
047
Epoch 20/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
5911 - depth_loss3: 0.5911 - val_loss: 0.9640 - val_depth_loss3: 0.9
640
Epoch 21/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
3260 - depth_loss3: 0.3260 - val_loss: 0.8054 - val_depth_loss3: 0.8
054
Epoch 22/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
2467 - depth_loss3: 0.2467 - val_loss: 0.4927 - val_depth_loss3: 0.4
927
Epoch 23/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
2233 - depth_loss3: 0.2233 - val_loss: 0.3628 - val_depth_loss3: 0.3
628
Epoch 24/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
1372 - depth_loss3: 0.1372 - val_loss: 0.4741 - val_depth_loss3: 0.4
741
Epoch 25/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
2239 - depth_loss3: 0.2239 - val_loss: 0.4303 - val_depth_loss3: 0.4
303
Epoch 26/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
1519 - depth_loss3: 0.1519 - val_loss: 0.7111 - val_depth_loss3: 0.7
111
Epoch 27/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
1691 - depth_loss3: 0.1691 - val_loss: 0.4743 - val_depth_loss3: 0.4
743
Epoch 28/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
1728 - depth_loss3: 0.1728 - val_loss: 0.4733 - val_depth_loss3: 0.4
733
Epoch 29/30
```

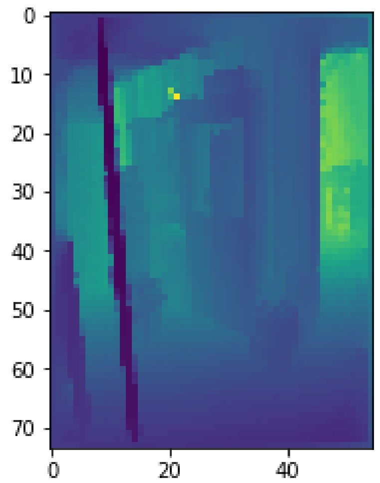
```
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
2076 - depth_loss3: 0.2076 - val_loss: 0.3486 - val_depth_loss3: 0.3
486
Epoch 30/30
1328/1328 [==============================] - 12s 9ms/step - loss: 0.
2280 - depth_loss3: 0.2280 - val_loss: 0.4498 - val_depth_loss3: 0.4
498
```

In [23]:

```
# print(training_data[0,:].shape)
# print(np.reshape(training_data[0,:],[im_sze1,im_sze2,3]).shape)
plt.imshow(training_data[1])
plt.show()


print(np.reshape(depth_data[1],[depth_sze1,depth_sze2]).shape)
plt.imshow(np.reshape(depth_data[1],[depth_sze2,depth_sze1]))
plt.show()
```

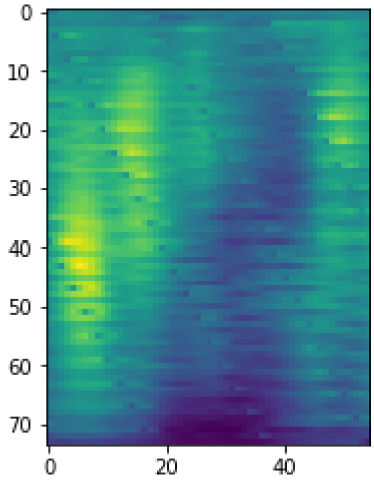



```
(55, 74)
```



In [35]:

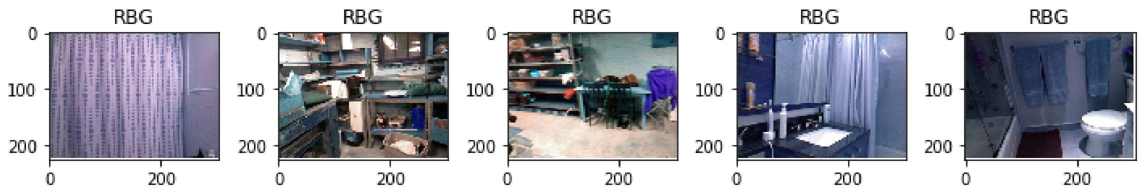
```
y_pred_prob=parallel_model3.predict(np.reshape(training_data[1],[-1,im_sze2,im_s
ze1,3]))
# y_pred_prob.shape
# .shape

plt.imshow(np.reshape(y_pred_prob*255,[depth_sze2,depth_sze1]))
plt.show()
```

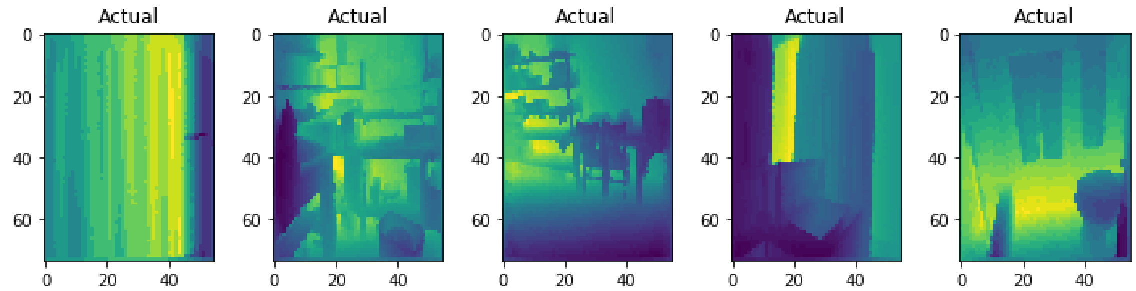


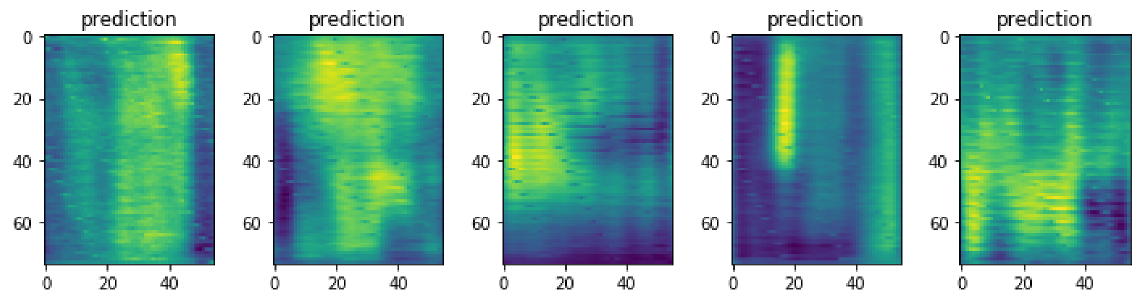
Sample predictions on radomly selected examples from training set

In [28]:

```
f = plt.figure(figsize=(10,15))
i=1
num_select=5
select=np.random.choice(len(training_data), num_select, replace=False)
for i in range(1,num_select+1):

    ax1=plt.subplot(3, num_select, i)
    ax1.imshow(training_data[select[i-1]])
    ax1.set_title('RBG')

    ax2=plt.subplot(3, num_select, i+num_select)
    ax2.imshow(depth_data[select[i-1]])
    ax2.set_title('Actual')

    y_pred_prob=parallel_model3.predict(np.reshape(training_data[select[i-1]]/25
5,[-1,im_sze2,im_sze1,3]))
    ax3=plt.subplot(3, num_select, i+2*num_select)
    ax3.imshow(np.reshape(y_pred_prob,[depth_sze2,depth_sze1]))
    ax3.set_title('prediction')
    #i=i+1
plt.tight_layout()
plt.show()
```



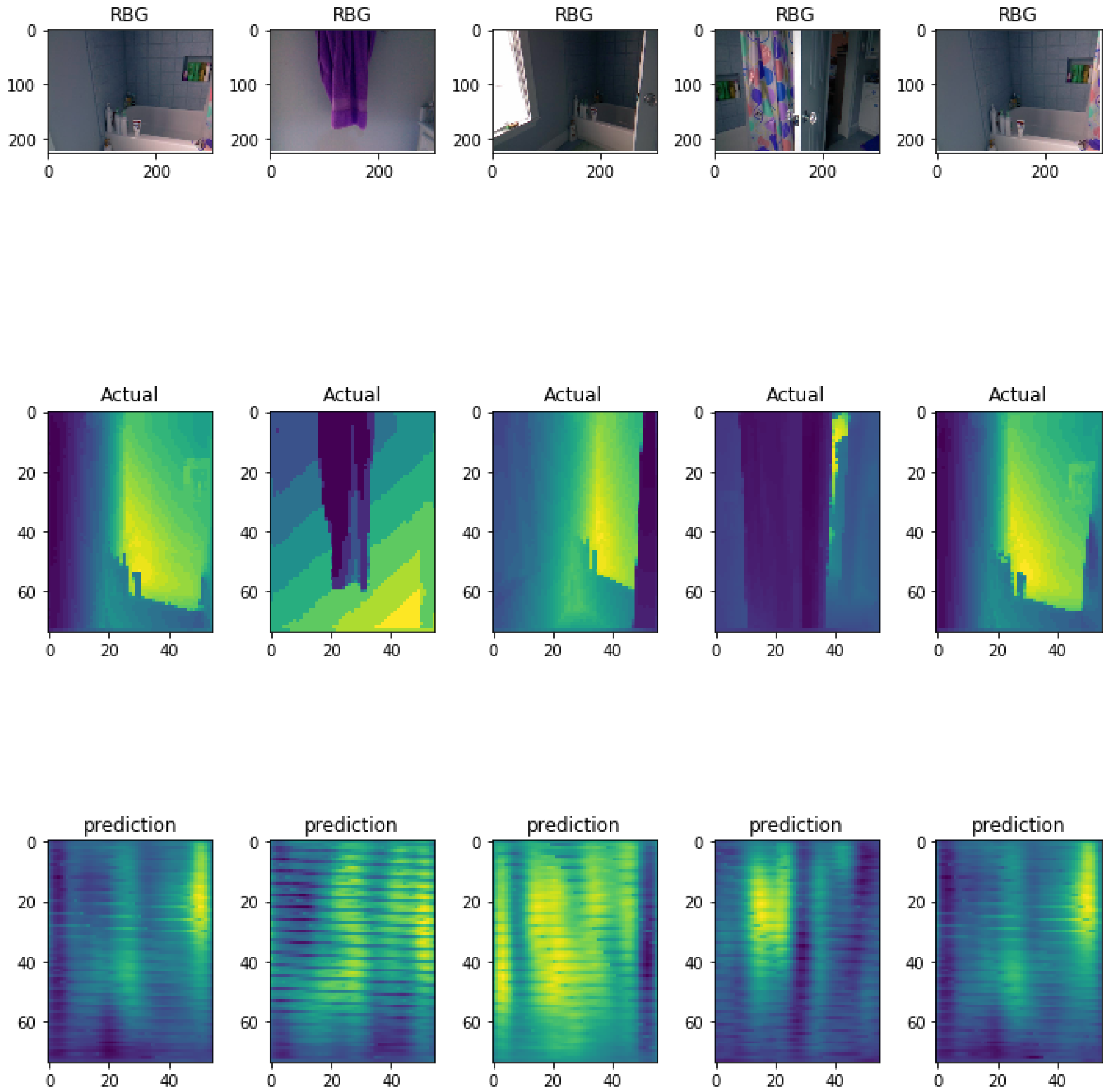




Predict Test Data

In [34]:

```
f = plt.figure(figsize=(10,12))
i=1
select=np.random.choice(len(test_data), 5, replace=False)
numselect=5
for i in range(1,6):
    #im1=cv2.imread(os.path.join(Datadir,'Good Parts',img1))
    #im2=cv2.imread(os.path.join(Datadir,'Defects',img2))

    ax1=plt.subplot(3, num_select, i)
    ax1.imshow(test_data[select[i-1]])
    ax1.set_title('RBG')

    ax2=plt.subplot(3, num_select, i+num_select)
    ax2.imshow(np.reshape(test_depth_data[select[i-1]],[depth_sze2,depth_sze1]))
    ax2.set_title('Actual')

    y_pred_prob=parallel_model3.predict(np.reshape(test_data[select[i-1]]/255,[-
1,im_sze2,im_sze1,3]))
    ax3=plt.subplot(3, num_select, i+2*num_select)
    ax3.imshow(np.reshape(y_pred_prob*255,[depth_sze2,depth_sze1]))
    ax3.set_title('prediction')

plt.tight_layout()
plt.show()
```



In [ ]:

```
plt.imshow(test_data[0])
plt.show()
plt.imshow(test_depth_data[0])
plt.show()
```

In [ ]:

```
selections
```

In [ ]:

```
help (np.random.choice)
```