Kitti-Document

Unknown Author

April 3, 2015

Introduction

The goal is learning features using ego-motion/odometry.

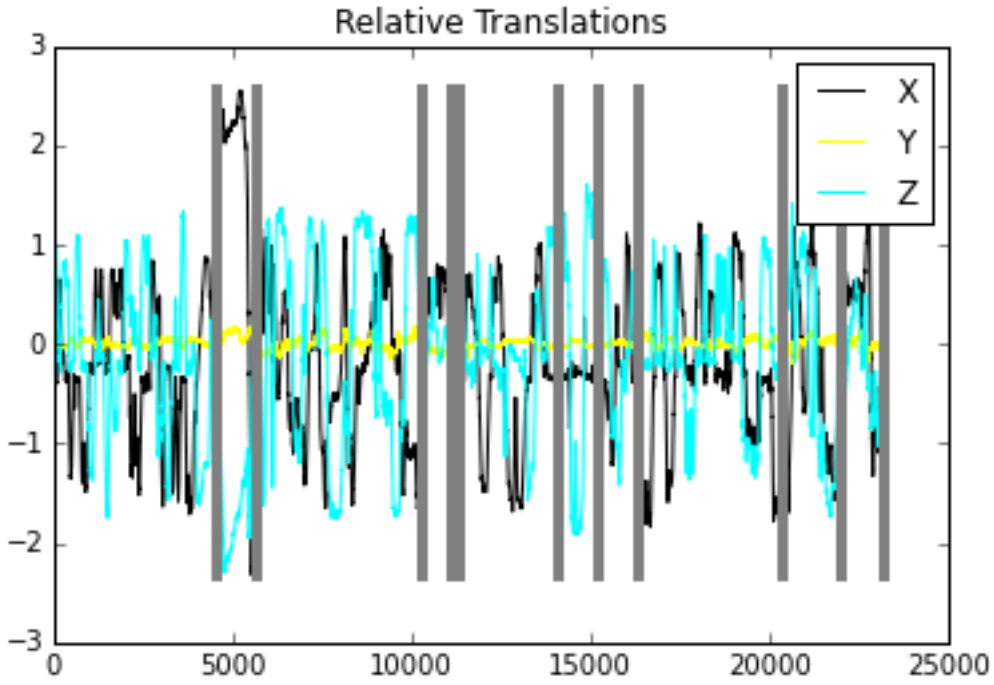
Kitti provides data of a car moving aroud with a camera and records the movements made by the car. For each frame the transformation with respect to frame number 1 of the sequence is provided. There are 11 total sequences for which data is present. The transformation is provided as a 3x4 transformation matrix. I represent transformation as translation + three euler angles.

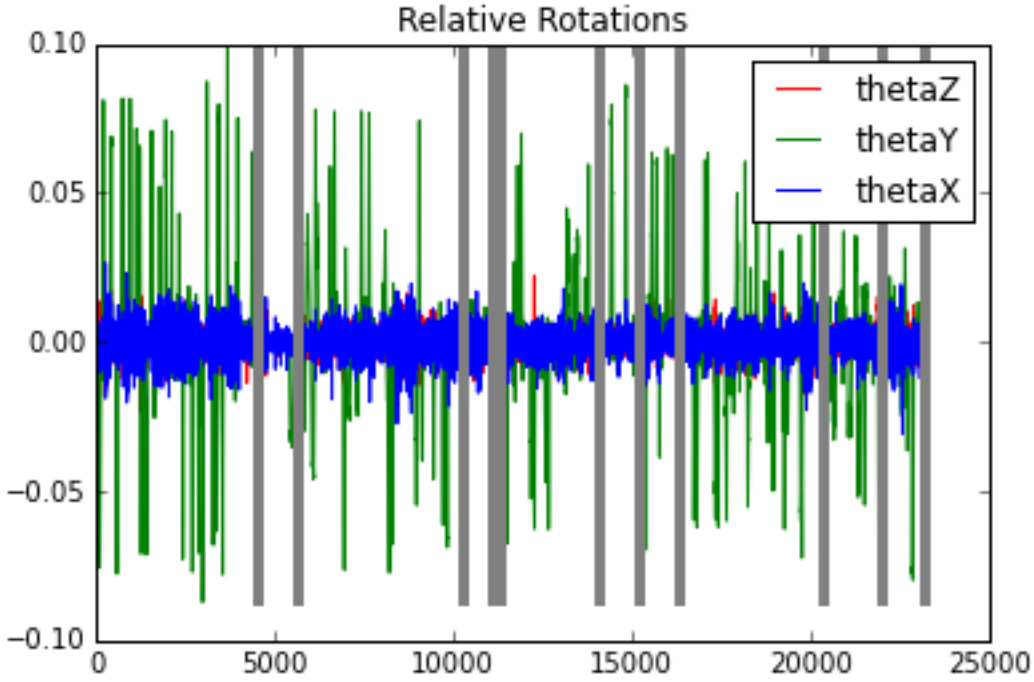
My goal is to train a network to predict these 6 transformation parameters from pairs of images. I can either regress to these 6 parameters or I can bin these transformation and train a classification network. Regressing the rotations has an issue that rotation of 360degrees is same as 0 degrees. But if the rotations are small then this is not an issue. Currently, my plan is to regress.

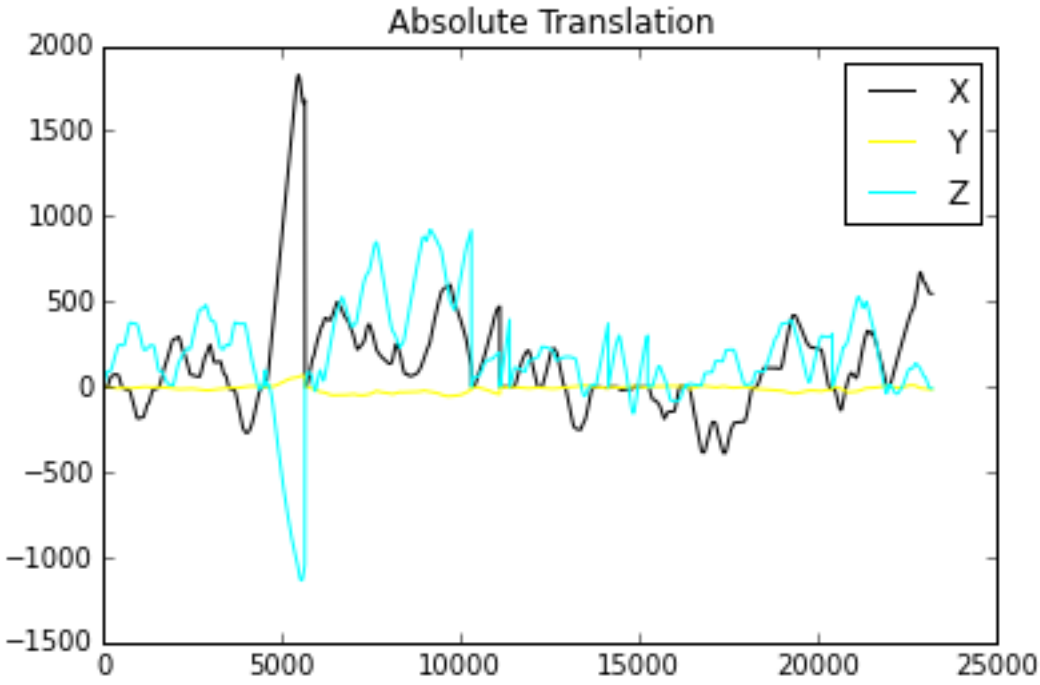
If my network is not working - then one thing I can try is to use optical flow and then try to regress on these variables.Labels of the Data

For understanding how the data looks like, I created a visualization of the labels. The three plots contain data for relative translation and rotations between consequent frames and absolute translation between frames as compared to the first frame. Each vertical grey line indicates the end of a sequence. There are 11 sequences overall. As expected, the most translation changes are along (X,Z) directions (The camera points in the Z direction) and rotations are about the Y axis.

In [1]:
```
%matplotlib inline
%load_ext autoreload
%autoreload 2
import os
kittiDir = '/work4/pulkitag-code/pkgs/caffe-v2-2/modelFiles/kitti/codes'
os.chdir(kittiDir)
import kitti_utils as ku
ku.plot_pose(seqNum='all', poseType='euler')
```


Relative Translations
X
Y
Z
3
2
1
0
−1
−2
−3
0
5000
10000
15000
20000
25000

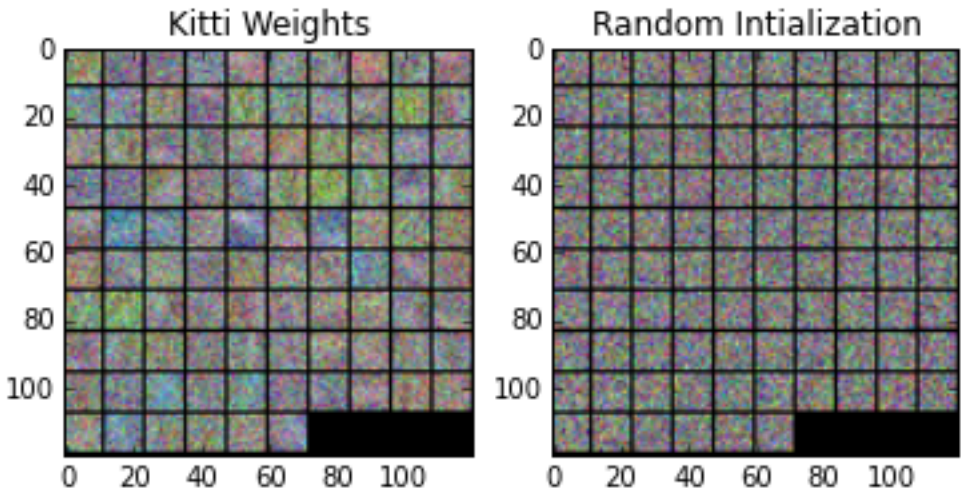

Relative Rotations
thetaZ
thetaY
thetaX
0.10
0.05
0.00
−0.05
−0.10
0
5000
10000
15000
20000
25000



Initialization from Scratch

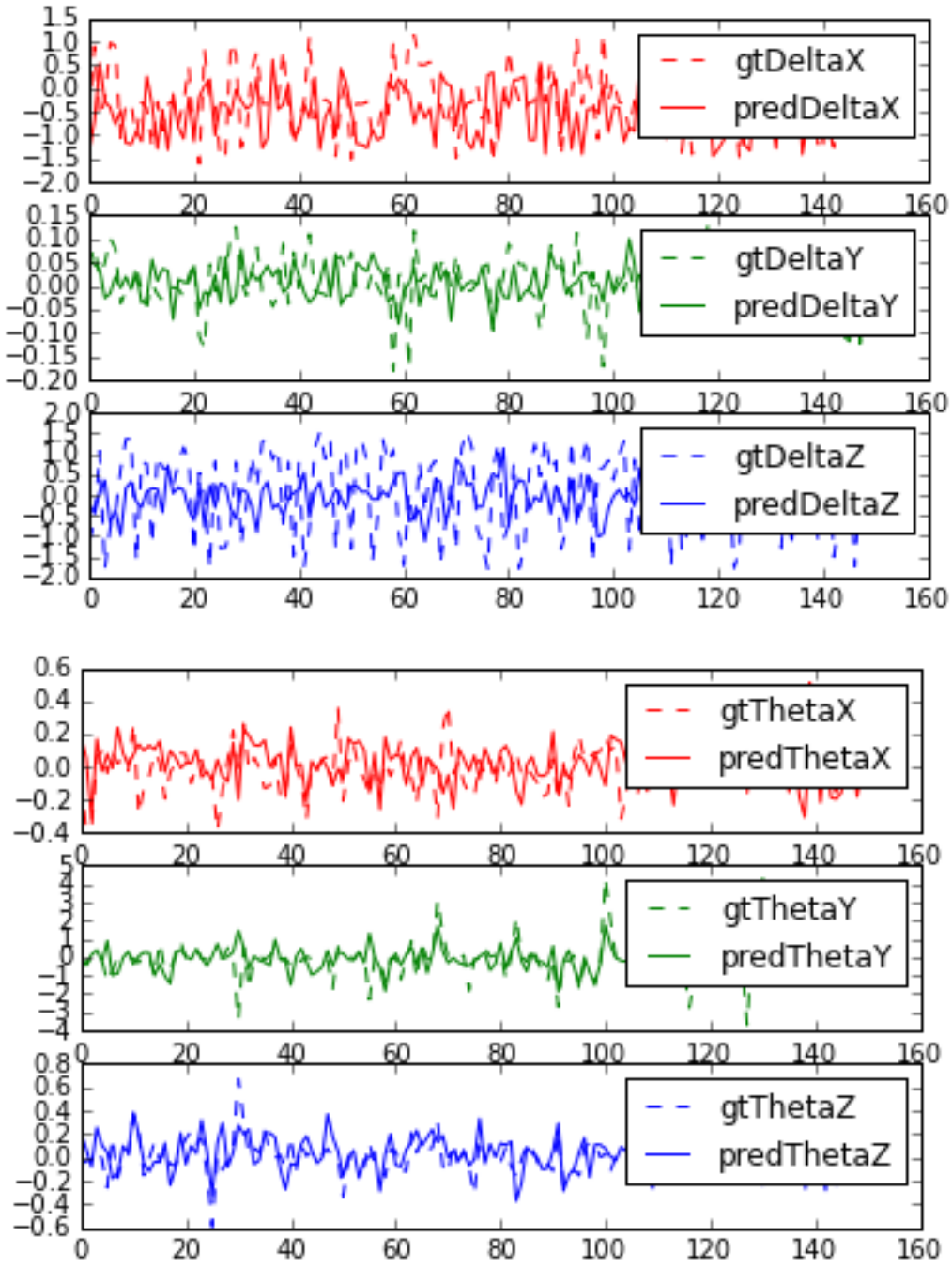
Weights of the network trained from scratch after 30,000 iterations. The network overfits. The training error is very small, but test error is quite large. (See the plots below).

In [31]:
```
import my_pycaffe as mp
import matplotlib.pyplot as plt
mp = reload(mp)
ax1 = plt.subplot(1,2,1)
ax2 = plt.subplot(1,2,2)
weightFile, protoFile = ku.get_weight_proto_file(isScratch=True, numIter=30000)
net = mp.MyNet(protoFile, weightFile)
net.vis_weights('conv1', ax=ax1, titleName='Kitti Weights')
netRandom = mp.MyNet(protoFile)
netRandom.vis_weights('conv1', ax=ax2, titleName='Random Intialization')
```



In [22]:
```
ku.get_accuracy(numIter=30000, imSz=256, poseType='euler', nrmlzType='zScoreScaleSeper
                isScratch=True, concatLayer='pool5', numBatches=3)
```

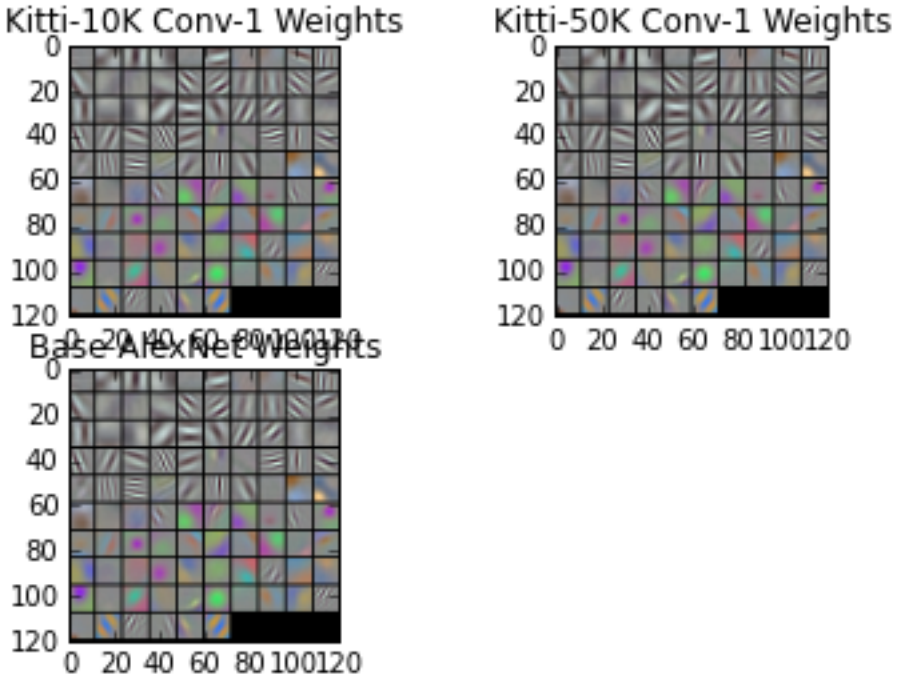
```
Intializing Network
Calculating Features
Plotting Results
```



The accuracy of the model trained from scratch. It can be seen that is dismal. Looking at rotation loss, especially for thetaY, it seems like L1 penalty of currently used euclidean loss maybe a plausible way forward.Initializing Training with AlexNet

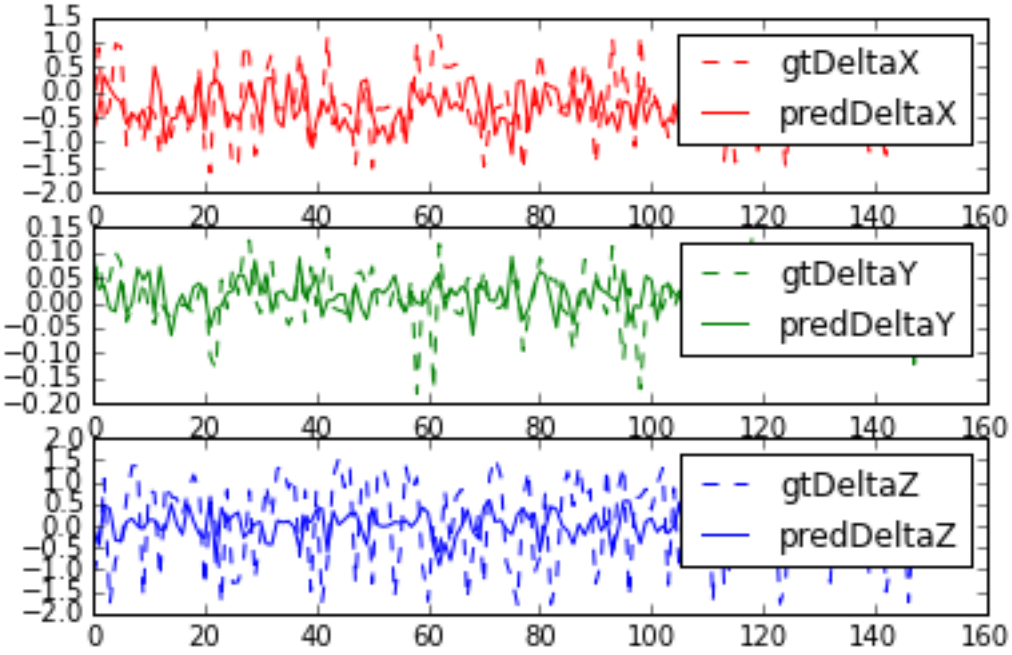
One thing we notice is that

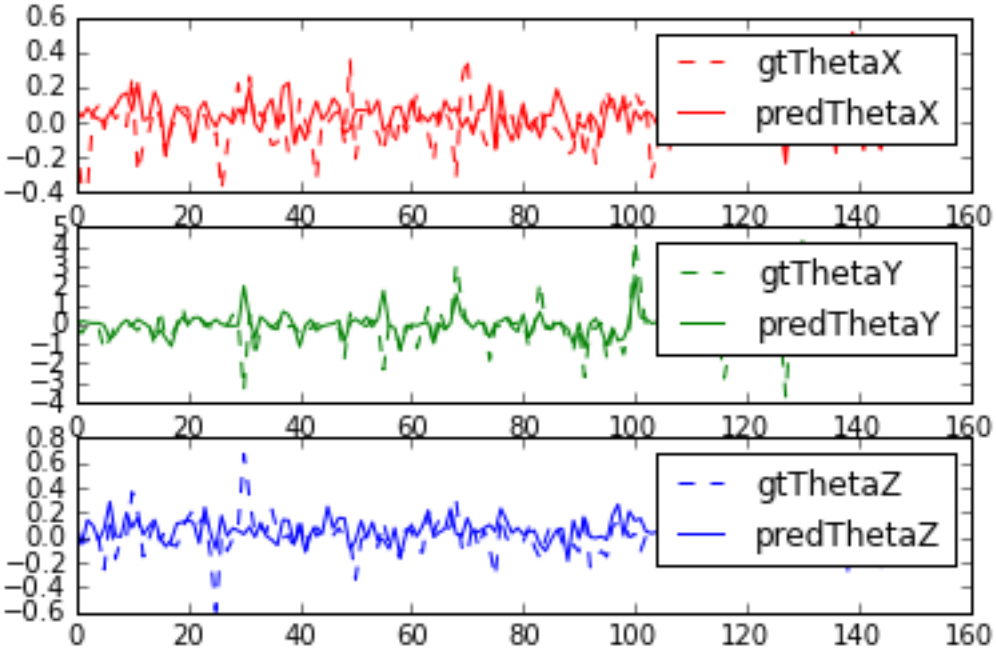
In [32]:
```
import my_pycaffe as mp
import matplotlib.pyplot as plt
mp = reload(mp)
ax1 = plt.subplot(2,2,1)
ax2 = plt.subplot(2,2,2)
ax3 = plt.subplot(2,2,3)
alexWeightFile = '/data1/pulkitag/caffe_models/caffe_imagenet_train_iter_310000'
weightFile10K, protoFile = ku.get_weight_proto_file(isScratch=False, numIter=20000)
weightFile50K, protoFile = ku.get_weight_proto_file(isScratch=False, numIter=50000)
netTune10K = mp.MyNet(protoFile, weightFile10K)
netTune10K.vis_weights('conv1',ax=ax1, titleName='Kitti-10K Conv-1 Weights')
netTune50K = mp.MyNet(protoFile, weightFile50K)
netTune50K.vis_weights('conv1',ax=ax2, titleName='Kitti-50K Conv-1 Weights')
netBase = mp.MyNet(protoFile, alexWeightFile)
netBase.vis_weights('conv1',ax=ax3, titleName='Base AlexNet Weights')
```



In [27]:
```
ku.get_accuracy(numIter=20000, imSz=256, poseType='euler', nrmlzType='zScoreScaleSeper
                isScratch=False, concatLayer='pool5', numBatches=3)
```

```
Intializing Network
Calculating Features
Plotting Results
```


gtDeltaX
predDeltaX
gtDeltaY
predDeltaY
gtDeltaZ
predDeltaZ


gtThetaX
predThetaX
gtThetaY
predThetaY
gtThetaZ
predThetaZ

In []: