In [1]:
```
%%HTML
<style> code {background-color : pink !important;} </style>
```

Camera Calibration with OpenCV

Run the code in the cell below to extract object points and image points for camera calibration.

In [3]:
```
import numpy as np
import cv2
import glob
import matplotlib.pyplot as plt
%matplotlib qt

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((6*9,3), np.float32)
objp[:,:2] = np.mgrid[0:9, 0:6].T.reshape(-1,2)

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d points in real world space
imgpoints = [] # 2d points in image plane.

# Make a list of calibration images
#images = glob.glob('calibration_wide/GO*.jpg')
images = glob.glob('Canon6D-Rokinon14mmf2.8/IMG*.jpg')

# Step through the list and search for chessboard corners
for idx, fname in enumerate(images):
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Find the chessboard corners
    ret, corners = cv2.findChessboardCorners(gray, (9,6), None)
    if idx==0:
        print(corners[0])
        print(corners[0][0])
        print(corners[1][0])
        print(np.array([corners[0][0],corners[1][0],corners[9][0],corners[10][
0]]))

    # If found, add object points, image points
    if ret == True:
        objpoints.append(objp)
        imgpoints.append(corners)

        # Draw and display the corners
        cv2.drawChessboardCorners(img, (9,6), corners, ret)
        write_name = './Canon6D-Rokinon14mmf2.8_calib/'+'corners_found'+str(id
x)+'.jpg'
        cv2.imwrite(write_name, img)
        cv2.imshow('img', img)
        cv2.waitKey(100)

cv2.destroyAllWindows()
```

```
[[957.0776 584.1981]]
[957.0776 584.1981]
[1100.7969   598.66846]
[[ 957.0776   584.1981 ]
 [1100.7969   598.66846]
 [ 959.5271   730.12067]
 [1102.9689   735.1699 ]]
```

If the above cell ran sucessfully, you should now have `objpoints` and `imgpoints` needed for camera calibration. Run the cell below to calibrate, calculate distortion coefficients, and test undistortion on an image!

In [4]:
```
import pickle
%matplotlib inline

# Test undistortion on an image
#img = cv2.imread('calibration_wide/test_image.jpg')
img = cv2.imread('Canon6D-Rokinon14mmf2.8/test3.jpg')
img_size = (img.shape[1], img.shape[0])
#print(objpoints)

# Do camera calibration given object points and image points
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, img_s
ize,None,None)


dst = cv2.undistort(img, mtx, dist, None, mtx)
cv2.imwrite('Canon6D-Rokinon14mmf2.8/test3_undist.jpg',dst)

# Save the camera calibration result for later use (we won't worry about rvecs
 / tvecs)
dist_pickle = {}
dist_pickle["mtx"] = mtx
dist_pickle["dist"] = dist
pickle.dump( dist_pickle, open( "Canon6D-Rokinon14mmf2.8/wide_dist_pickle.p",
"wb" ) )
#dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
# Visualize undistortion
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20,10))
ax1.imshow(img)
ax1.set_title('Original Image', fontsize=30)
ax2.imshow(dst)
ax2.set_title('Undistorted Image', fontsize=30)
```

Out[4]: <matplotlib.text.Text at 0x1d3f10837f0>