Make the board and report images based on saved data

Data is saved in `5 - Depth Calculation OFD.ipynb` using the `grabframe` parameter

1. Load the `frame_results.p` file
2. process and make the figures

In [44]:
```
# set the to get the correct path
user = 'marcvanzyl'
```

In [45]:
```
import numpy as np
import cv2, os
from cv2 import aruco
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
import pickle
%matplotlib inline
```

In [46]:
```
import pandas as pd
```

Load the list of files

In [47]:
```
datadir = "/Users/{}/Google Drive/ScienceFair2021/DataCapture/smooth/".format


video_files = np.array([f for f in os.listdir(datadir) if f.endswith(".mp4")

# just sorts the files
video_files.sort()
```

In [48]:
```
video_files
```

Out[48]:
```
array(['Smooth_15_-2_0_4623HF.mp4', 'Smooth_15_-2_1_QV4WNZ.mp4',
       'Smooth_15_-2_2_JB90U4.mp4', 'Smooth_15_-2_3_MCDNNA.mp4',
       'Smooth_15_-2_4_VIWO2O.mp4', 'Smooth_15_-4_0_ESS22N.mp4',
       'Smooth_15_-4_1_8GD6QO.mp4', 'Smooth_15_-4_2_FMBNMX.mp4',
       'Smooth_15_-4_3_EEUONU.mp4', 'Smooth_15_-4_4_64CXH3.mp4',
       'Smooth_15_-6_0_2HDNNZ.mp4', 'Smooth_15_-6_1_D23FV7.mp4',
       'Smooth_15_-6_2_U8BJY4.mp4', 'Smooth_15_-6_3_GK86VH.mp4',
       'Smooth_15_-6_4_JU8DFH.mp4', 'Smooth_15_0_0_D26LA1.mp4',
       'Smooth_15_0_1_OOBKBZ.mp4', 'Smooth_15_0_2_MNBSLT.mp4',
       'Smooth_15_0_3_VY12DF.mp4', 'Smooth_15_0_4_46CIDO.mp4',
       'Smooth_30_-2_0_07HKRJ.mp4', 'Smooth_30_-2_1_UA678T.mp4',
       'Smooth_30_-2_2_Z2CEPT.mp4', 'Smooth_30_-2_3_DPNF63.mp4',
       'Smooth_30_-2_4_O2G6SZ.mp4', 'Smooth_30_-4_0_9WKFQF.mp4',
       'Smooth_30_-4_1_B6QVWT.mp4', 'Smooth_30_-4_2_9YG7KZ.mp4',
       'Smooth_30_-4_3_ZZ3YEH.mp4', 'Smooth_30_-4_4_ARXESZ.mp4',
       'Smooth_30_-6_0_HOA2A3.mp4', 'Smooth_30_-6_1_2F304V.mp4',
       'Smooth_30_-6_2_R5OVIS.mp4', 'Smooth_30_-6_3_7A1JE7.mp4',
       'Smooth_30_-6_4_YKIR3Y.mp4', 'Smooth_30_0_0_TOWEUT.mp4',
       'Smooth_30_0_1_RB5B21.mp4', 'Smooth_30_0_2_AAU13M.mp4',
       'Smooth_30_0_3_STATJY.mp4', 'Smooth_30_0_4_YKCRBC.mp4',
       'Smooth_45_-2_0_3P4T66.mp4', 'Smooth_45_-2_1_JFT8WZ.mp4',
       'Smooth_45_-2_2_XM1ETY.mp4', 'Smooth_45_-2_3_BR9Z93.mp4',
       'Smooth_45_-2_4_OZMRTY.mp4', 'Smooth_45_-4_0_GNGX25.mp4',
       'Smooth_45_-4_1_DPV5FD.mp4', 'Smooth_45_-4_2_BZ37KH.mp4',
       'Smooth_45_-4_3_GRNIXW.mp4', 'Smooth_45_-4_4_810HRV.mp4',
       'Smooth_45_-6_0_R378Z4.mp4', 'Smooth_45_-6_1_0NECPK.mp4',
       'Smooth_45_-6_2_U2KIG0.mp4', 'Smooth_45_-6_3_9UF74H.mp4',
       'Smooth_45_-6_4_WAKIAQ.mp4', 'Smooth_45_0_0_V4SN9S.mp4',
       'Smooth_45_0_1_IGTHOY.mp4', 'Smooth_45_0_2_B8A482.mp4',
       'Smooth_45_0_3_9GBKOP.mp4', 'Smooth_45_0_4_SLPA8L.mp4',
       'Smooth_60_-2_0_EENTSE.mp4', 'Smooth_60_-2_1_TOLJ0V.mp4',
       'Smooth_60_-2_2_QJP3WV.mp4', 'Smooth_60_-2_3_C9X5WP.mp4',
       'Smooth_60_-2_4_UX8ITL.mp4', 'Smooth_60_-4_0_EVQ115.mp4',
       'Smooth_60_-4_1_9CNJ4F.mp4', 'Smooth_60_-4_2_XJTK3X.mp4',
       'Smooth_60_-4_3_DDB2LK.mp4', 'Smooth_60_-4_4_PPJSIF.mp4',
       'Smooth_60_-6_0_EO5Q0F.mp4', 'Smooth_60_-6_1_ITFYF8.mp4',
       'Smooth_60_-6_2_P77OMH.mp4', 'Smooth_60_-6_3_HY8MRS.mp4',
       'Smooth_60_-6_4_ONJJEZ.mp4', 'Smooth_60_0_0_6KASLJ.mp4',
       'Smooth_60_0_1_H2CBPV.mp4', 'Smooth_60_0_2_HI1FZ4.mp4',
       'Smooth_60_0_3_0IMBF0.mp4', 'Smooth_60_0_4_XV9Y0W.mp4'],
      dtype='<U25')
```

Camera properties

These are used to convert pixel displacement to distances

Camera features:

- sensor size = 3.68 x 2.76 mm
- sensor resolution = 3280 × 2464
- focal length = 3.04 mm

$$d_{mm} = \frac{pix \times 3.68}{3280}$$

The depth can now be found

$$Z = \frac{T \times f}{d_{mm}}$$

In [49]:
```
# number of pixel/frame displacemet per degree/s of rotation
rotational_coeff = 2.05661689

# camera frames per second
camera_fps = 21
```

In [50]:
```
# Actual distances
actual_dist = pd.Series([1550,1560,2020,2030,2575,2580], index=['OFZ0','OFZ1'

# convert to a dataframe
actual_df = pd.DataFrame(actual_dist)
actual_df.index.name = 'Zone'
actual_df
```

Out[50]:

Zone	Actual Distance [mm]
OFZ0	1550
OFZ1	1560
OFZ2	2020
OFZ3	2030
OFZ4	2575
OFZ5	2580

In [51]:
```
# calculate the movement of the camera each frame
def calc_camera_translation(v, frame_rate):
    return v/frame_rate

# calculate the distance using the optical flow pixel displacement
def calc_depth(pix, step):

    sensor_x = 3.68
    f = 3.04
    sensor_x_res = 3280

    d_mm = pix*sensor_x/sensor_x_res

    return step*f/d_mm
```

Load the results file

In [55]:
```
results = pickle.load(open('{}full_analysis_results.p'.format(datadir), 'rb')
```

In [ ]:
```
# extract the data
img = results['img']
filename = results['filename']
frame = results['frame']
flow = results['flow']
centers = results['centers']
center_ids = results['center_ids']
frame_res = results['frame_res']
```

In [56]:
```
results.keys()
```

Out[56]:
```
dict_keys(['15.0_-2.0', '15.0_-4.0', '15.0_-6.0', '15.0_0.0', '30.0_-2.0', '30
.0_-4.0', '30.0_-6.0', '30.0_0.0', '45.0_-2.0', '45.0_-4.0', '45.0_-6.0', '45.
0_0.0', '60.0_-2.0', '60.0_-4.0', '60.0_-6.0', '60.0_0.0'])
```

In [57]:
```
# parameters to calculate distance
rot_vel = -2
lin_vel = 60
```

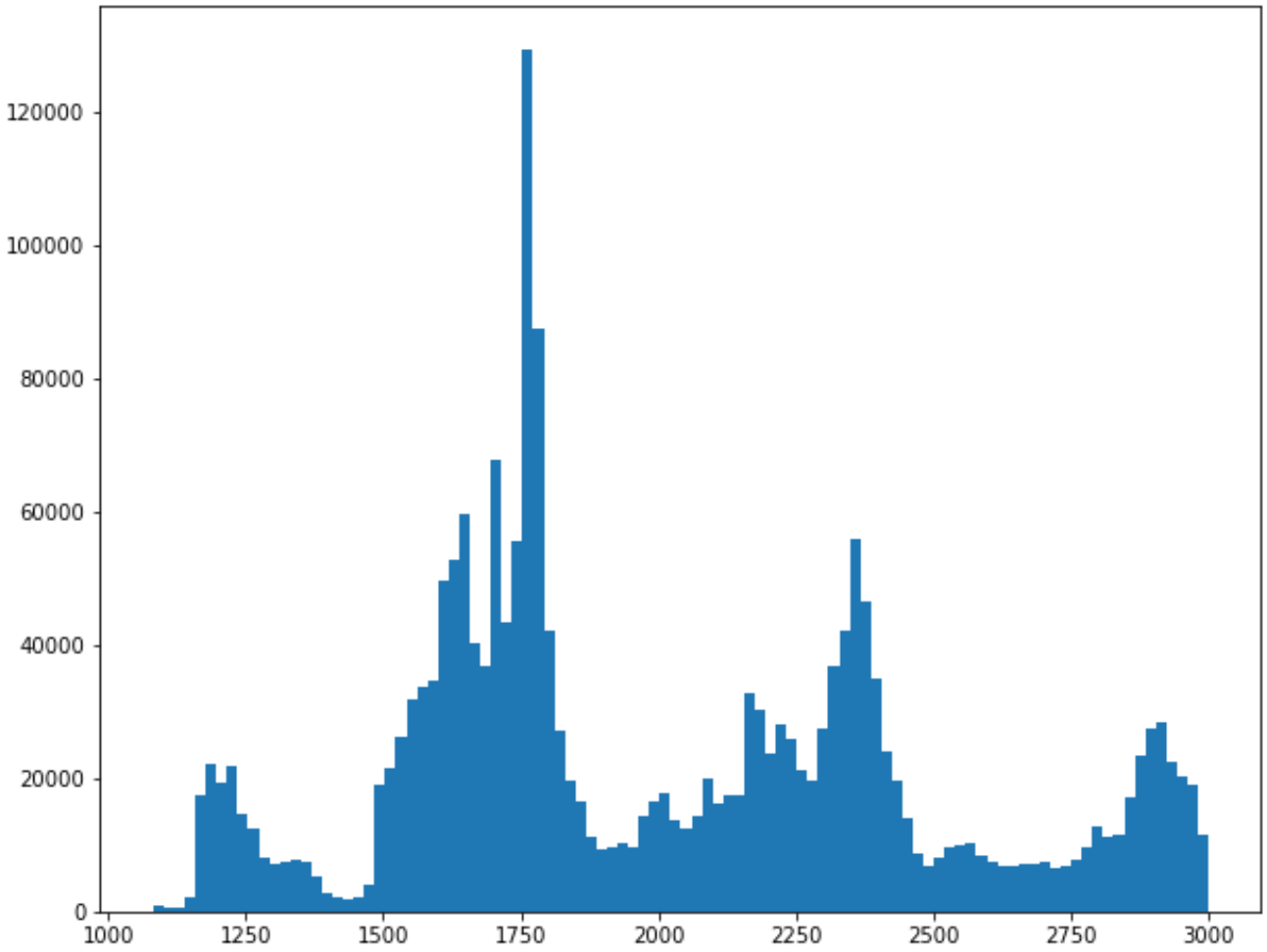
In [ ]:
```
# create bgr image from grayscale
alpha = np.ones_like(img)*80  # create the alpha channel
img_3gray = cv2.merge((img,img,img,alpha)) # create gray bgr image

hsv = cv2.cvtColor(img_3gray, cv2.COLOR_BGR2HSV)
```

In [13]:
```
mag = (cv2.multiply(cv2.add(flow[...,0],-10),5))

mag[mag<0.0] = 0
mag[mag>255] = 255
```

In [14]:
```
flow_x = flow[...,0]  # extract the x flow

dist = calc_depth((flow_x/2.2 + 1.05*rotational_coeff*rot_vel), calc_camera_t

dist[dist<500] = 0
dist[dist>3000] = 0
```

In [15]:
```
# remove all the points with less than 100mm distance (elminiate the zeroes)
dist_hist = dist[dist>100]
```

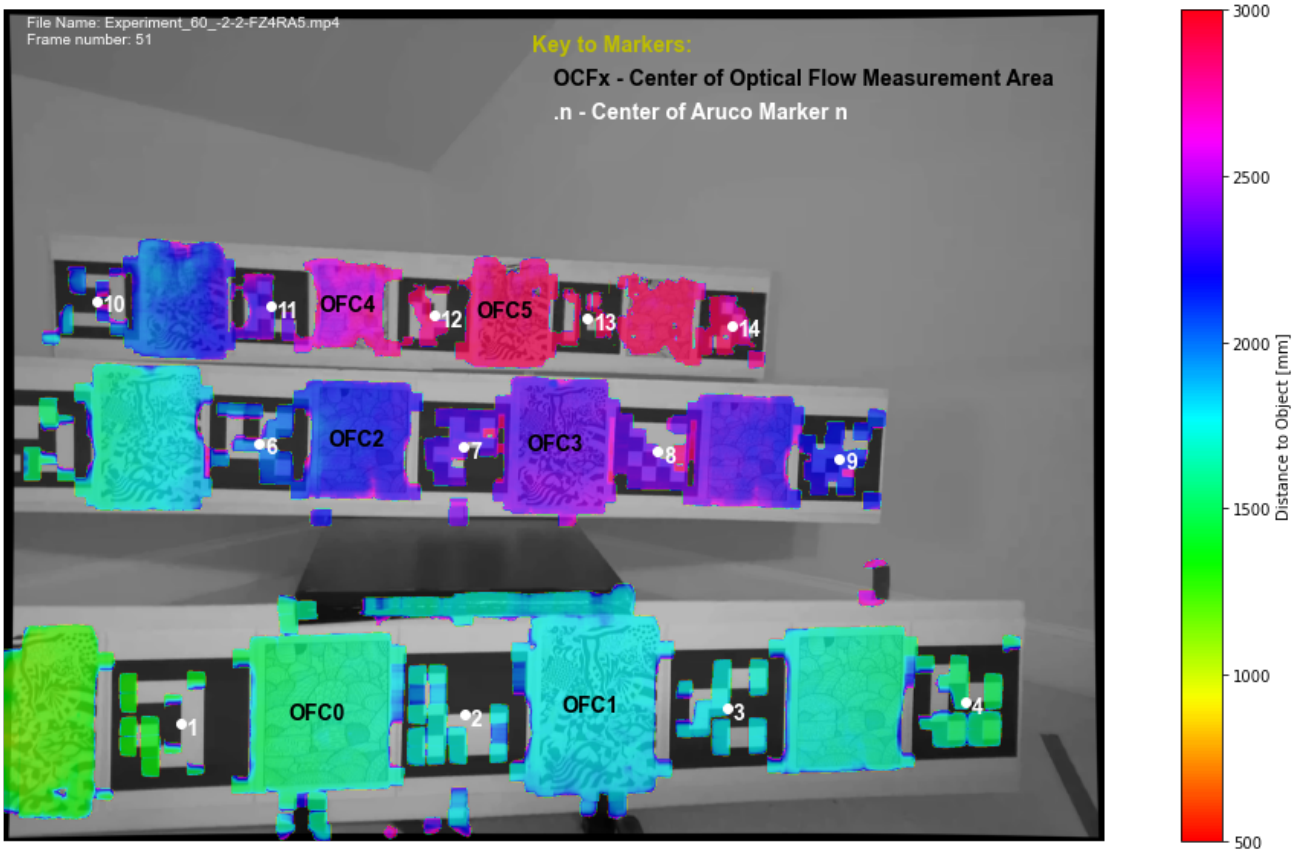
In [16]:
```
# plot the distance distribution
fig, ax = plt.subplots(figsize=(10,8))
ax.hist(dist_hist.reshape(1,-1)[0], bins=100)
plt.show()
```



In [17]:
```
dist[dist>2900] = 2900 # This an adjustment to clip the exremities in this pa
```

In [18]:

```
fig, ax = plt.subplots(figsize=(18,10))



# create bgr image from grayscale
alpha = np.ones_like(img)*80  # create the alpha channel
alpha[dist<300] = 255   # remove transparency where there is no flow informat


img_3gray = cv2.merge((img,img,img,alpha)) # create gray bgr image
hsv = cv2.cvtColor(img_3gray, cv2.COLOR_BGR2HSV)

pos = ax.imshow(dist, vmin=500., vmax=3000,cmap='hsv')
fig.colorbar(pos, ax=ax, label="Distance to Object [mm]")
ax.imshow(img_3gray)
ax.set_axis_off()

annot_font = {'fontname':'Arial', 'size':'14','weight':'bold'}
annot_font_filename = {'fontname':'Arial', 'size':'10'}

x = centers[:,0]
y = centers[:,1]

ax.scatter(x,y, color='w')
for i, ids in enumerate(center_ids):
    ax.annotate('{}'.format(int(ids)), (x[i]+20, y[i]+30), color='w',  **anno

for ind in frame_res[frame_res.index.str.contains('OFC')].index:
    ax.annotate('{}'.format(ind), (frame_res[ind][0]-100,frame_res[ind][1]),

ax.annotate('Key to Markers:', xy=(9,.3), xytext=(0.48, 0.95), textcoords='ax



ax.annotate('OCFx - Center of Optical Flow Measurement Area', xy=(9,.3), xyte
ax.annotate('.n - Center of Aruco Marker n', xy=(9,.3), xytext=(0.5, 0.87), t

ax.annotate('File Name: {}'.format(filename.split('/')[-1]), xy=(.1,.3), xyte
ax.annotate('Frame number: {}'.format(frame), xy=(9,.3), xytext=(0.02, 0.96),



plt.savefig('{}_{}_figure'.format(results['filename'][:-4], results['frame'])
        orientation='portrait', format=None,
        transparent=False, bbox_inches=None, pad_inches=0.1,
         metadata=None)
plt.show()
```



Final Results Plot

This is the final analysis to generate the summary plot

In [141…
```
result_dict = pickle.load(open('{}full_analysis_results.p'.format(datadir), '
```

In [142…
```
result_dict.keys()
```

Out[142…
```
dict_keys(['15.0_-2.0', '15.0_-4.0', '15.0_-6.0', '15.0_0.0', '30.0_-2.0', '30
.0_-4.0', '30.0_-6.0', '30.0_0.0', '45.0_-2.0', '45.0_-4.0', '45.0_-6.0', '45.
0_0.0', '60.0_-2.0', '60.0_-4.0', '60.0_-6.0', '60.0_0.0'])
```

In [143…
```
result_dict['30.0_-4.0']
```

Out[143…

Zone	n	Raw Flow [pix/frame]	Linear Flow [pix/frame]	Distance [mm]	Actual Dist [mm]	Frame Error [mm]	Frame StdDev [mm]	Rolling Window Error [mm]	Rolling Window Error % [%]	Rol Winc E Stdl [n
OFZ0	415	11.656	2.789	1415.3	1550	-134.7	230.0	-25.99	-1.68	84
OFZ1	386	11.036	2.669	1478.8	1560	-81.2	229.6	-36.06	-2.31	59
OFZ2	388	11.239	2.076	1901.6	2020	-118.4	410.6	20.45	1.01	122
OFZ3	386	10.818	2.044	1930.6	2030	-99.4	411.3	-9.74	-0.48	97
OFZ4	377	11.106	1.696	2328.0	2575	-247.0	624.6	-33.64	-1.31	19
OFZ5	377	10.670	1.661	2376.2	2580	-203.8	628.2	-54.40	-2.11	152

In [190…
```
# Grab the results
res_15_6 = result_dict['15.0_-6.0']
res_30_6 = result_dict['30.0_-6.0']
res_45_6 = result_dict['45.0_-6.0']
res_60_6 = result_dict['60.0_-6.0']
```

In [191…
```
# Grab the results
res_15_4 = result_dict['15.0_-4.0']
res_30_4 = result_dict['30.0_-4.0']
res_45_4 = result_dict['45.0_-4.0']
res_60_4 = result_dict['60.0_-4.0']
```

In [192…
```
# Grab the results
res_15_2 = result_dict['15.0_-2.0']
res_30_2 = result_dict['30.0_-2.0']
res_45_2 = result_dict['45.0_-2.0']
res_60_2 = result_dict['60.0_-2.0']
```

In [193…
```
# Grab the results
res_15_0 = result_dict['15.0_0.0']
res_30_0 = result_dict['30.0_0.0']
res_45_0 = result_dict['45.0_0.0']
res_60_0 = result_dict['60.0_0.0']
```

In [194…
```
error_ofd_0 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
error_ofd_0['15mm/s'] = res_15_0[('Rolling Window Error','[mm]')]
error_ofd_0['30mm/s'] = res_30_0[('Rolling Window Error','[mm]')]
error_ofd_0['45mm/s'] = res_45_0[('Rolling Window Error','[mm]')]
error_ofd_0['60mm/s'] = res_60_0[('Rolling Window Error','[mm]')]
```

In [195…
```
error_ofd_2 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
error_ofd_2['15mm/s'] = res_15_2[('Rolling Window Error','[mm]')]
error_ofd_2['30mm/s'] = res_30_2[('Rolling Window Error','[mm]')]
error_ofd_2['45mm/s'] = res_45_2[('Rolling Window Error','[mm]')]
error_ofd_2['60mm/s'] = res_60_2[('Rolling Window Error','[mm]')]
```

In [196…
```
error_ofd_4 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
error_ofd_4['15mm/s'] = res_15_4[('Rolling Window Error','[mm]')]
error_ofd_4['30mm/s'] = res_30_4[('Rolling Window Error','[mm]')]
error_ofd_4['45mm/s'] = res_45_4[('Rolling Window Error','[mm]')]
error_ofd_4['60mm/s'] = res_60_4[('Rolling Window Error','[mm]')]
```

In [197…
```
error_ofd_6 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
error_ofd_6['15mm/s'] = res_15_6[('Rolling Window Error','[mm]')]
error_ofd_6['30mm/s'] = res_30_6[('Rolling Window Error','[mm]')]
error_ofd_6['45mm/s'] = res_45_6[('Rolling Window Error','[mm]')]
error_ofd_6['60mm/s'] = res_60_6[('Rolling Window Error','[mm]')]
```

In [198…
```
error_ofd_6
```

Out[198…

	15mm/s	30mm/s	45mm/s	60mm/s
OFZ0	-140.87	11.24	-0.02	-5.14
OFZ1	-199.62	-8.38	0.05	8.53
OFZ2	-133.81	83.68	83.07	62.77
OFZ3	-251.28	24.12	54.64	54.85
OFZ4	-228.29	30.97	54.02	16.79
OFZ5	-342.88	-8.27	43.36	28.02

In [199…
```
std_ofd_0 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
std_ofd_0['15mm/s'] = res_15_0[('Rolling Window Error StdDev','[mm]')]
std_ofd_0['30mm/s'] = res_30_0[('Rolling Window Error StdDev','[mm]')]
std_ofd_0['45mm/s'] = res_45_0[('Rolling Window Error StdDev','[mm]')]
std_ofd_0['60mm/s'] = res_60_0[('Rolling Window Error StdDev','[mm]')]
```

In [200…
```
std_ofd_2 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
std_ofd_2['15mm/s'] = res_15_2[('Rolling Window Error StdDev','[mm]')]
std_ofd_2['30mm/s'] = res_30_2[('Rolling Window Error StdDev','[mm]')]
std_ofd_2['45mm/s'] = res_45_2[('Rolling Window Error StdDev','[mm]')]
std_ofd_2['60mm/s'] = res_60_2[('Rolling Window Error StdDev','[mm]')]
```

In [201…
```
std_ofd_4 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
std_ofd_4['15mm/s'] = res_15_4[('Rolling Window Error StdDev','[mm]')]
std_ofd_4['30mm/s'] = res_30_4[('Rolling Window Error StdDev','[mm]')]
std_ofd_4['45mm/s'] = res_45_4[('Rolling Window Error StdDev','[mm]')]
std_ofd_4['60mm/s'] = res_60_4[('Rolling Window Error StdDev','[mm]')]
```

In [202…
```
std_ofd_6 = pd.DataFrame(columns=['15mm/s','30mm/s', '45mm/s', '60mm/s', ])
std_ofd_6['15mm/s'] = res_15_6[('Rolling Window Error StdDev','[mm]')]
std_ofd_6['30mm/s'] = res_30_6[('Rolling Window Error StdDev','[mm]')]
std_ofd_6['45mm/s'] = res_45_6[('Rolling Window Error StdDev','[mm]')]
std_ofd_6['60mm/s'] = res_60_6[('Rolling Window Error StdDev','[mm]')]
```

In [203…
```
stereo_err = pd.Series({'OFZ0':42,'OFZ1':42,'OFZ2':82,'OFZ3':82,'OFZ4':45,'OF
stereo_std = pd.Series({'OFZ0':90,'OFZ1':90,'OFZ2':140,'OFZ3':140,'OFZ4':185,
```

In [204…

```
# create the x vector for positions along the x-axis
x_pos = np.array(range(4))
x = [ 'OF:15', 'OF:30', 'OF:45', 'OF:60'] # not sure why blank is required
```

In [256…

```
fig, ax = plt.subplots(3,2, figsize=(8.5,8))
plt.tight_layout()
for ind, target in enumerate(actual_dist.index):
    ind_y = int(ind/2)
    ind_x = int(ind%2)

    y_0 = error_ofd_0.loc[target].values
    e_0 = std_ofd_0.loc[target].values

    y_2 = error_ofd_2.loc[target].values
    e_2 = std_ofd_2.loc[target].values

    y_4 = error_ofd_4.loc[target].values
    e_4 = std_ofd_4.loc[target].values

    y_6 = error_ofd_6.loc[target].values
    e_6 = std_ofd_6.loc[target].values


    ax[ind_y, ind_x].set_title('Target: {} (Actual distance: {}mm)'.format(ta
    ax[ind_y, ind_x].axhline(y=0, xmin=0, xmax=1, color='k', linestyle='--')
    # The red line for the stereo results
    stereo = ax[ind_y, ind_x].axhline(y=stereo_err[target], xmin=0, xmax=1,
                      color='red', linestyle='-')



    stereo_var = ax[ind_y, ind_x].axhspan(stereo_err[target]-stereo_std[targe
                      stereo_err[target]+stereo_std[target],
                      xmin=0, xmax=1, color='grey', alpha=0.10)
    if ind_x == 0:
        ax[ind_y, ind_x].set_ylabel('  \nError and StdDev [mm]')
    # 0 deg/s
    r_0 = ax[ind_y, ind_x].errorbar(x_pos-.166, y_0, e_0, linestyle='None', c
                       ecolor='k', elinewidth=5, label='0 deg/s')
    # 2 deg/s
#    ax[ind_y, ind_x].errorbar(x_pos+.166, y_2, e_2, linestyle='None', color=
#                        ecolor='mediumblue', elinewidth=5, label='-2 deg/s')
    # 4 deg/s
    r_4 = ax[ind_y, ind_x].errorbar(x_pos, y_4, e_4, linestyle='None', color=
                       ecolor='darkviolet', elinewidth=5, label='-4 deg/s')

    # 6 deg/s
    r_6 = ax[ind_y, ind_x].errorbar(x_pos+.166, y_6, e_6, linestyle='None', c
                       ecolor='fuchsia', elinewidth=5, label='-6 deg/s')


    ax[ind_y, ind_x].set_ylim([-400,400])


    if ind_y == 2:
```

```
        ax[ind_y, ind_x].set_xlabel('Experiment (Linear Camera Velocity [mm/s
        ax[ind_y, ind_x].set_xticks(x_pos)
        ax[ind_y, ind_x].set_xticklabels(x)
    else:
        ax[ind_y, ind_x].set_xticks(x_pos)
        ax[ind_y, ind_x].set_xticklabels([])
        a=1


#handles, labels = ax[0,0].get_legend_handles_labels()
#ax[0,0].legend((stereo, stereo_var, r_0, r_4, r_6), ('Stereo', 'Stereo StdDe
plt.legend((stereo, stereo_var, r_0, r_4, r_6),
           ('Stereo', 'Stereo StdDev','0 deg/sec', '-4 deg/sec', '-6 deg/sec'
           loc='upper center', bbox_to_anchor=(0, -0.2),
          fancybox=True, ncol=5)

plt.savefig('{}error and std dev.png'.format(datadir),dpi=300, bbox_inches =
plt.show()
```