In [656]:
```
import sys
!{sys.executable} -m pip install imutils
```

```
Collecting imutils
  Downloading imutils-0.4.5.tar.gz
Installing collected packages: imutils
  Running setup.py install for imutils ... done
Successfully installed imutils-0.4.5
```

In [2]:
```
import scipy.io as spio
import matplotlib.pyplot as plt
import numpy as np
from skimage.color import rgb2gray
from sklearn.decomposition import PCA
import cv2
```

Test 1: Ideal Case

Loading Data

In [4]:
```
cam1_1 = spio.loadmat('cam1_1.mat', squeeze_me=True)
cam2_1 = spio.loadmat('cam2_1.mat', squeeze_me=True)
cam3_1 = spio.loadmat('cam3_1.mat', squeeze_me=True)

vidFrames1_1 = cam1_1['vidFrames1_1']
vidFrames2_1 = cam2_1['vidFrames2_1']
vidFrames3_1 = cam3_1['vidFrames3_1']

# Extract number of frames
[m, n, c, n_frames1_1] = vidFrames1_1.shape
[m, n, c, n_frames2_1] = vidFrames2_1.shape
[m, n, c, n_frames3_1] = vidFrames3_1.shape
```

Extract Positions Of A Paint Can

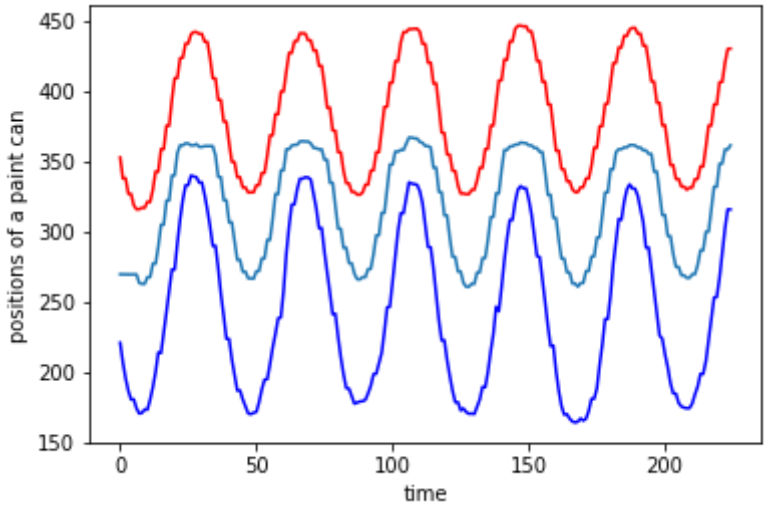
In [93]:
```
camera1_1_x = []
camera1_1_y = []
for j in range(1,n_frames1_1):
    frame = rgb2gray(np.float64(vidFrames1_1[:,:,:,j]))
    CroppedFrame = frame[200:390,290:355]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 290
    y = y + 200
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.average(x)], [np.average(y)])
    camera1_1_y.append(np.average(y))
    camera1_1_x.append(np.average(x))
    #plt.show()
```

In [94]:
```
camera2_1_x = []
camera2_1_y = []

for j in range(1,n_frames2_1):
    frame = rgb2gray(np.float64(vidFrames2_1[:,:,:,j]))
    CroppedFrame = frame[100:360,250:330]
    thresh = cv2.threshold(CroppedFrame, 250, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 250
    y = y + 100
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.average(x)], [np.average(y)])
    camera2_1_y.append(np.average(y))
    camera2_1_x.append(np.average(x))
    #plt.show()
```

In [95]:
```
camera3_1_x = []
camera3_1_y = []

for j in range(1,n_frames3_1):
    frame = rgb2gray(np.float64(vidFrames3_1[:,:,:,j]))
    CroppedFrame = frame[250:320,280:470]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 280
    y = y + 250
    #plt.imshow(CroppedFrame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera3_1_y.append(np.average(y))
    camera3_1_x.append(np.average(x))
    #plt.show()
```

Carefully Shift Data To Align Waves In Phase

In [271]:
```
plt.plot(camera1_1_y)
plt.plot(camera2_1_y[10:235], 'b')
plt.plot(camera3_1_x[:225], 'r')
plt.xlabel('time')
plt.ylabel('positions of a paint can')
plt.show()
```

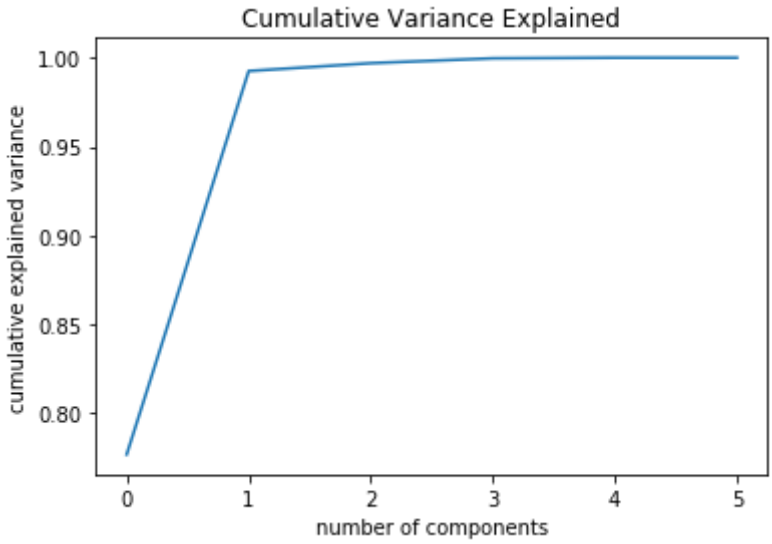


PCA Analysis

In [97]:
```
X_1 = np.stack((camera1_1_x, camera1_1_y,
                camera2_1_x[10:235], camera2_1_y[10:235],
                camera3_1_x[:225], camera3_1_y[:225]))

# Centering the data
X_1 = X_1 - np.mean(X_1, axis=0)
X_1.shape
```

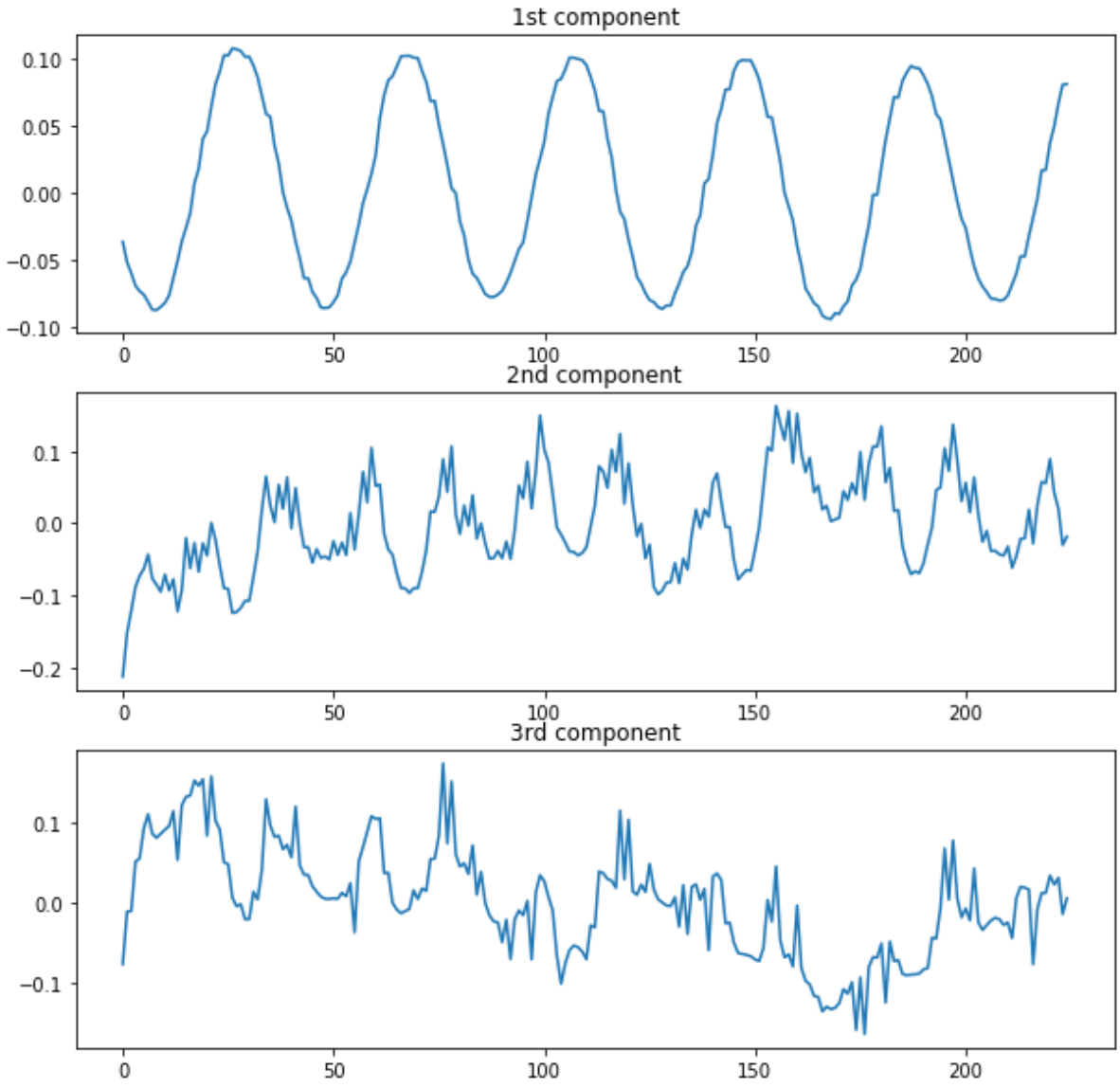
Out[97]: (6, 225)

In [98]:
```
pca1 = PCA().fit(X_1)
plt.title("Cumulative Variance Explained")
plt.plot(np.cumsum(pca1.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



POD Modes

In [99]:
```
plt.figure(figsize=(10,10))
plt.subplot(3,1,1)
plt.title("1st component")
plt.plot(pca1.components_[1])
plt.subplot(3,1,2)
plt.title("2nd component")
plt. plot(pca1.components_[2])
plt.subplot(3,1,3)
plt.title("3rd component")
plt. plot(pca1.components_[3])
plt.show()
```



Test 2: Noisy Case

Loading Data

In [62]:
```
cam1_2 = spio.loadmat('cam1_2.mat', squeeze_me=True)
cam2_2 = spio.loadmat('cam2_2.mat', squeeze_me=True)
cam3_2 = spio.loadmat('cam3_2.mat', squeeze_me=True)

vidFrames1_2 = cam1_2['vidFrames1_2']
vidFrames2_2 = cam2_2['vidFrames2_2']
vidFrames3_2 = cam3_2['vidFrames3_2']

# Extract number of frames
[m, n, c, n_frames1_2] = vidFrames1_2.shape
[m, n, c, n_frames2_2] = vidFrames2_2.shape
[m, n, c, n_frames3_2] = vidFrames3_2.shape
```

Extract Positions Of A Paint Can

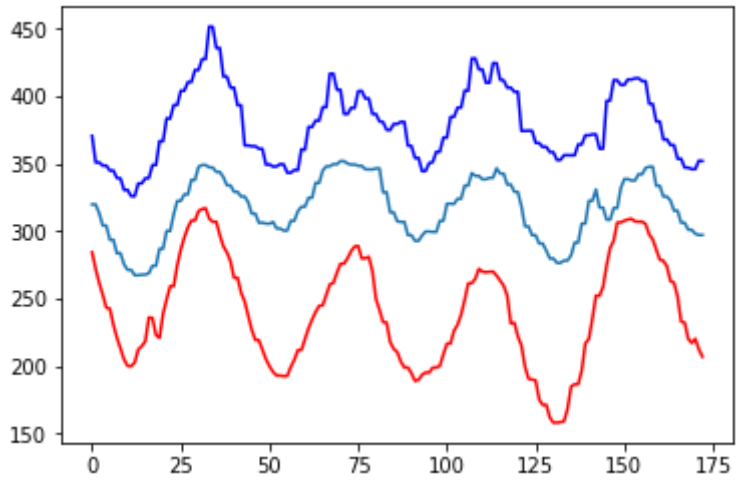
In [213]:
```
camera1_2_x = []
camera1_2_y = []

for j in range(1,n_frames1_2):
    frame = rgb2gray(np.float64(vidFrames1_2[:,:,:,j]))
    CroppedFrame = frame[240:400,320:390]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 320
    y = y + 240
    #plt.imshow(CroppedFrame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera1_2_y.append(np.average(y))
    camera1_2_x.append(np.average(x))
    #plt.show()
```

In [223]:
```
camera2_2_x = []
camera2_2_y = []

for j in range(1,n_frames2_2):
    frame = rgb2gray(np.float64(vidFrames2_2[:,:,:,j]))
    CroppedFrame = frame[70:420,200:410]
    thresh = cv2.threshold(CroppedFrame, 240, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 200
    y = y + 70
    #plt.imshow(thresh, cmap="gray")
    #plt.scatter([np.average(x)], [np.average(y)])
    camera2_2_y.append(np.average(y))
    camera2_2_x.append(np.average(x))
    #plt.show()
```

In [247]:
```
camera3_2_x = []
camera3_2_y = []

for j in range(1,n_frames3_2):
    frame = rgb2gray(np.float64(vidFrames3_2[:,:,:,j]))
    CroppedFrame = frame[210:330,280:490]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 280
    y = y + 210
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera3_2_y.append(np.average(y))
    camera3_2_x.append(np.average(x))
    #plt.show()
```

Carefully Shift Data To Align Waves In Phase

In [265]:
```
plt.plot(camera1_2_y[140:])
plt.plot(camera2_2_y[165:338], 'r')
plt.plot(camera3_2_x[145:318], 'b')
plt.show()
```

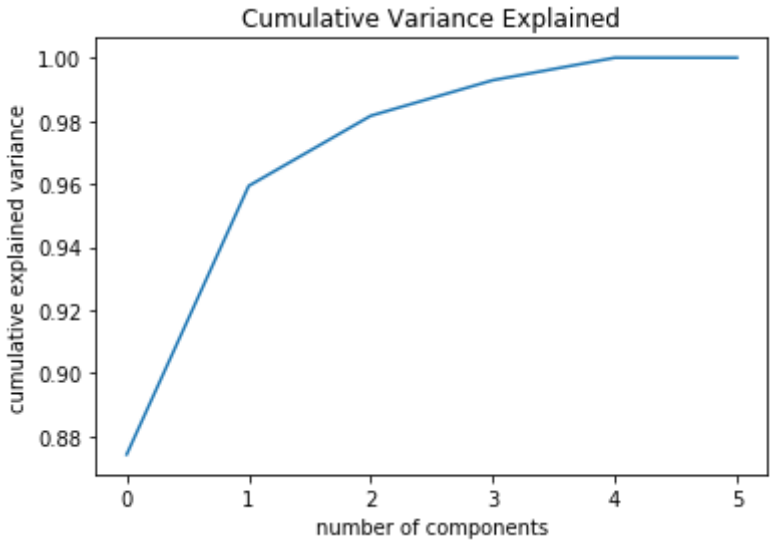


PCA Analysis

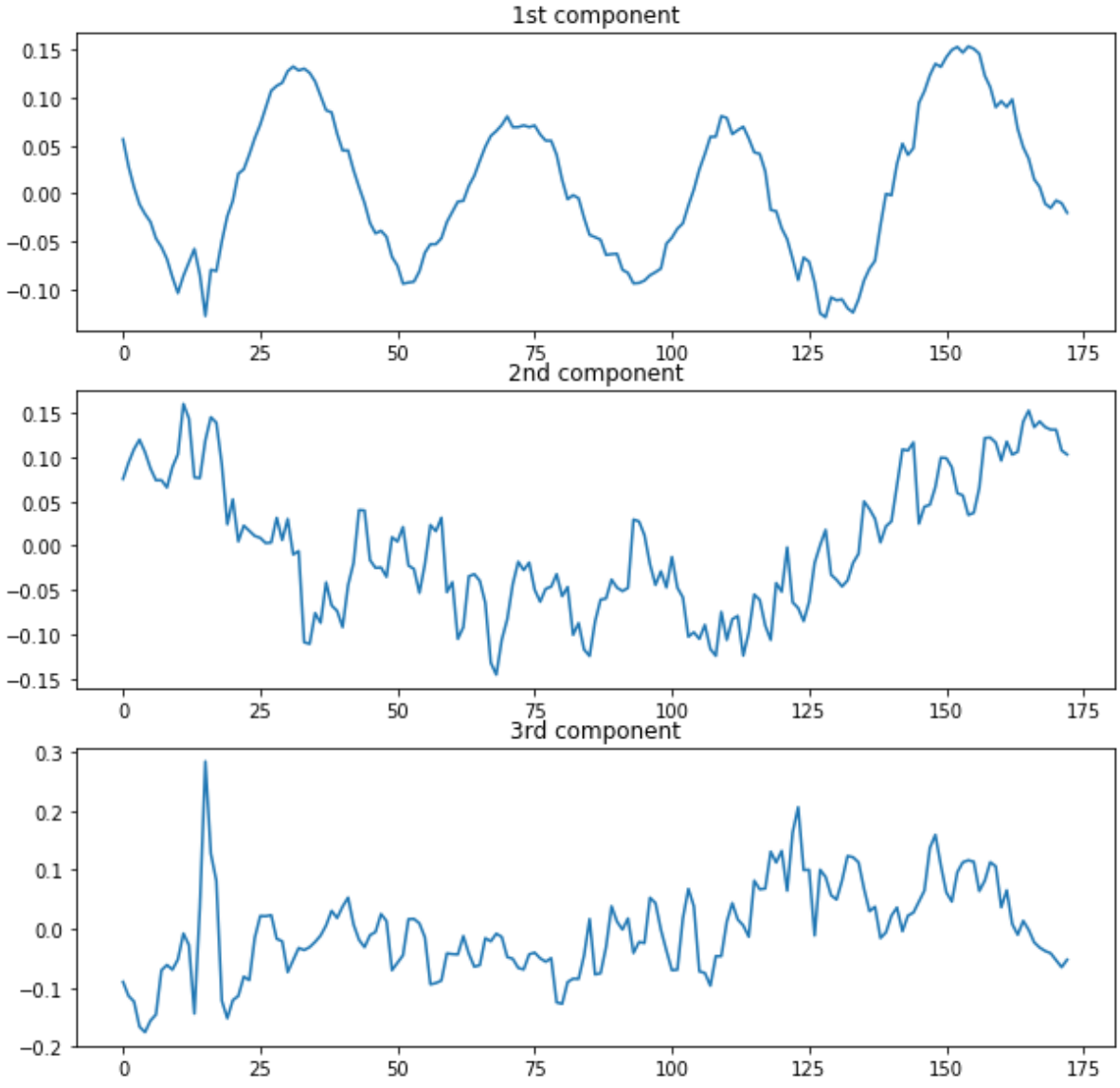
In [267]:
```
X_2 = np.stack((camera1_2_x[140:], camera1_2_y[140:],
                camera2_2_x[165:338], camera2_2_y[165:338],
                camera3_2_x[145:318], camera3_2_y[145:318]))

# Centering the data
X_2 = X_2 - np.mean(X_2, axis=0)
X_2.shape
```

Out[267]: (6, 173)

In [268]:
```
pca2 = PCA().fit(X_2)
plt.title("Cumulative Variance Explained")
plt.plot(np.cumsum(pca2.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



In [269]:
```
plt.figure(figsize=(10,10))
plt.subplot(3,1,1)
plt.title("1st component")
plt.plot(pca2.components_[1])
plt.subplot(3,1,2)
plt.title("2nd component")
plt.plot(pca2.components_[2])
plt.subplot(3,1,3)
plt.title("3rd component")
plt.plot(pca2.components_[3])
plt.show()
```



Test 3: Horizontal Displacement

Loading Data

In [ ]:
```
cam1_3 = spio.loadmat('cam1_3.mat', squeeze_me=True)
cam2_3 = spio.loadmat('cam2_3.mat', squeeze_me=True)
cam3_3 = spio.loadmat('cam3_3.mat', squeeze_me=True)

vidFrames1_3 = cam1_3['vidFrames1_3']
vidFrames2_3 = cam2_3['vidFrames2_3']
vidFrames3_3 = cam3_3['vidFrames3_3']

# Extract number of frames
[m, n, c, n_frames1_3] = vidFrames1_3.shape
[m, n, c, n_frames2_3] = vidFrames2_3.shape
[m, n, c, n_frames3_3] = vidFrames3_3.shape
```

Extract Positions Of A Paint Can

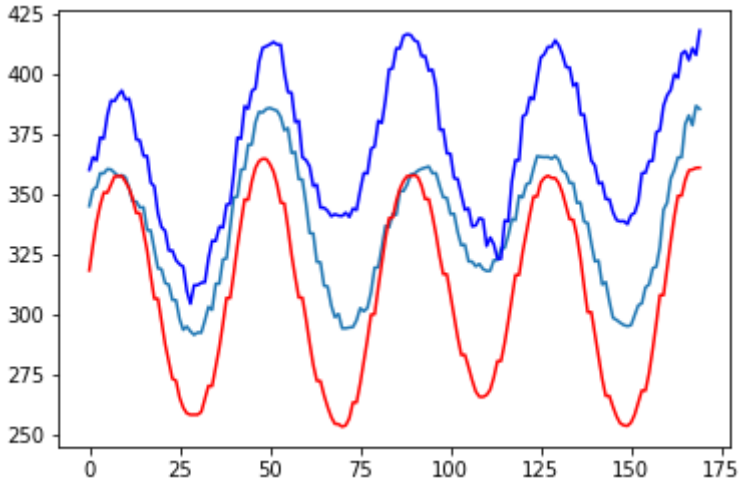
In [108]:
```
camera1_3_x = []
camera1_3_y = []
for j in range(1,n_frames3_1):
    frame = rgb2gray(np.float64(vidFrames1_3[:,:,:,j]))
    CroppedFrame = frame[240:400,280:370]
    max_v = np.amax(np.amax(CroppedFrame))
    [y,x] = np.where(max_v == CroppedFrame)
    x = x + 280
    y = y + 240
    #plt.imshow(CroppedFrame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera1_3_y.append(np.average(y))
    camera1_3_x.append(np.average(x))
    #plt.show()
```

In [117]:
```
camera2_3_x = []
camera2_3_y = []

for j in range(1,n_frames2_3):
    frame = rgb2gray(np.float64(vidFrames2_3[:,:,:,j]))
    CroppedFrame = frame[190:400,200:430]
    thresh = cv2.threshold(CroppedFrame, 250, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 200
    y = y + 190
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera2_3_y.append(np.average(y))
    camera2_3_x.append(np.average(x))
    #plt.show()
```

In [125]:
```
camera3_3_x = []
camera3_3_y = []

for j in range(1,n_frames3_3):
    frame = rgb2gray(np.float64(vidFrames3_3[:,:,:,j]))
    CroppedFrame = frame[200:330,270:450]
    thresh = cv2.threshold(CroppedFrame, 250, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 270
    y = y + 200
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera3_3_y.append(np.average(y))
    camera3_3_x.append(np.average(x))
    #plt.show()
```

Carefully Shift Data To Align Waves In Phase

In [201]:
```
plt.plot(camera1_3_y[7:177])
plt.plot(camera2_3_y[35:205], 'r')
plt.plot(camera3_3_x[:170], 'b')
plt.show()
```

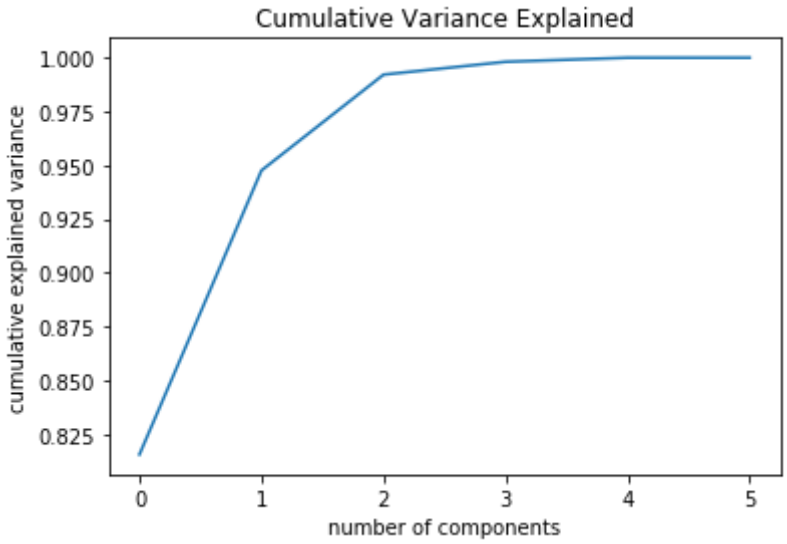


In [202]:
```
X_3 = np.stack((camera1_3_x[7:177], camera1_3_y[7:177],
                camera2_3_x[35:205], camera2_3_y[35:205],
                camera3_3_x[:170], camera3_3_y[:170]))

# Centering the data
X_3 = X_3 - np.mean(X_3, axis=0)
X_3.shape
```

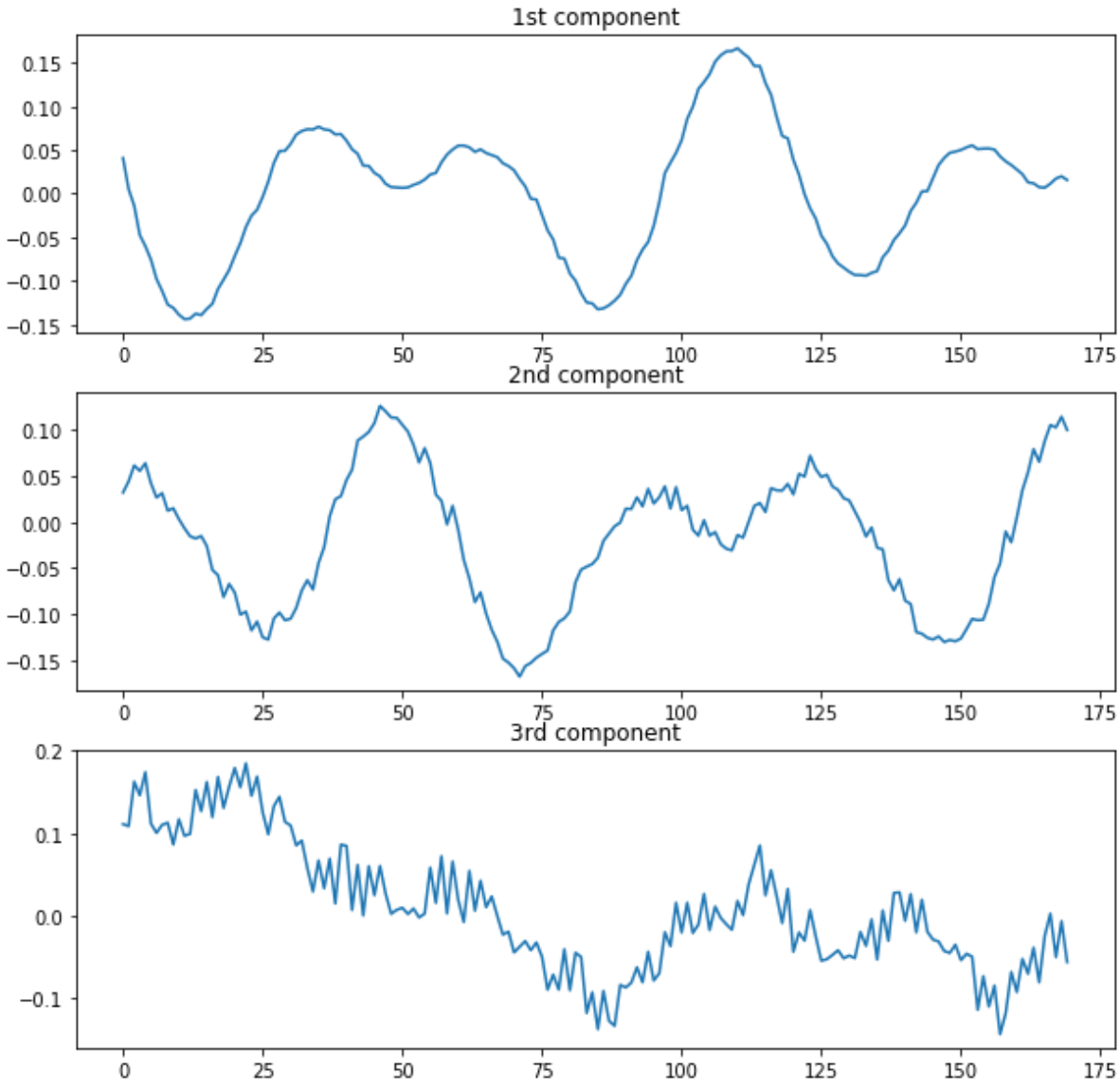
Out[202]: (6, 170)

PCA Analysis

In [203]:
```
pca3 = PCA().fit(X_3)
plt.title("Cumulative Variance Explained")
plt.plot(np.cumsum(pca3.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



In [204]:
```
plt.figure(figsize=(10,10))
plt.subplot(3,1,1)
plt.title("1st component")
plt.plot(pca3.components_[1,:])
plt.subplot(3,1,2)
plt.title("2nd component")
plt. plot(pca3.components_[2,:])
plt.subplot(3,1,3)
plt.title("3rd component")
plt. plot(pca3.components_[3,:])
plt.show()
```



Test 4: horizontal displacement and rotation

Loading Data

In [ ]:
```
cam1_4 = spio.loadmat('cam1_4.mat', squeeze_me=True)
cam2_4 = spio.loadmat('cam2_4.mat', squeeze_me=True)
cam3_4 = spio.loadmat('cam3_4.mat', squeeze_me=True)

vidFrames1_4 = cam1_4['vidFrames1_4']
vidFrames2_4 = cam2_4['vidFrames2_4']
vidFrames3_4 = cam3_4['vidFrames3_4']

# Extract number of frames
[m, n, c, n_frames1_4] = vidFrames1_4.shape
[m, n, c, n_frames2_4] = vidFrames2_4.shape
[m, n, c, n_frames3_4] = vidFrames3_4.shape
```

Extract Positions Of A Paint Can

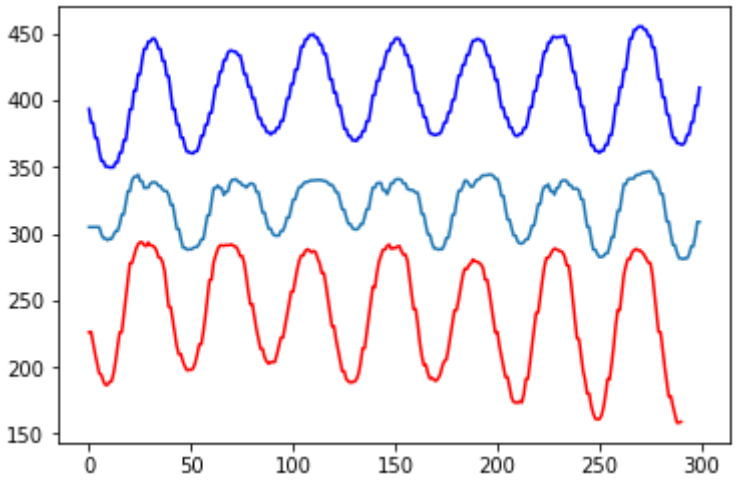
In [161]:
```
camera1_4_x = []
camera1_4_y = []

for j in range(1,n_frames1_4):
    frame = rgb2gray(np.float64(vidFrames1_4[:,:,:,j]))
    CroppedFrame = frame[250:360,340:430]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 340
    y = y + 250
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera1_4_y.append(np.average(y))
    camera1_4_x.append(np.average(x))
    #plt.show()
```

In [166]:
```
camera2_4_x = []
camera2_4_y = []

for j in range(1,n_frames2_4):
    frame = rgb2gray(np.float64(vidFrames2_4[:,:,:,j]))
    CroppedFrame = frame[130:300,220:410]
    thresh = cv2.threshold(CroppedFrame, 250, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 220
    y = y + 130
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera2_4_y.append(np.average(y))
    camera2_4_x.append(np.average(x))
    #plt.show()
```

In [168]:
```
camera3_4_x = []
camera3_4_y = []

for j in range(1,n_frames3_4):
    frame = rgb2gray(np.float64(vidFrames3_4[:,:,:,j]))
    CroppedFrame = frame[150:270,280:480]
    thresh = cv2.threshold(CroppedFrame, 200, 255, cv2.THRESH_BINARY)[1]
    max_v = np.amax(np.amax(thresh))
    [y,x] = np.where(max_v == thresh)
    x = x + 280
    y = y + 150
    #plt.imshow(frame, cmap="gray")
    #plt.scatter([np.median(x)], [np.median(y)])
    camera3_4_y.append(np.average(y))
    camera3_4_x.append(np.average(x))
    #plt.show()
```

Carefully Shift Data To Align Waves In Phase

In [177]:
```
plt.plot(camera1_4_y[:300])
plt.plot(camera2_4_y[9:309], 'r')
plt.plot(camera3_4_x[:300], 'b')
plt.show()
```



In [183]:
```
len(camera3_4_x[:300])
```

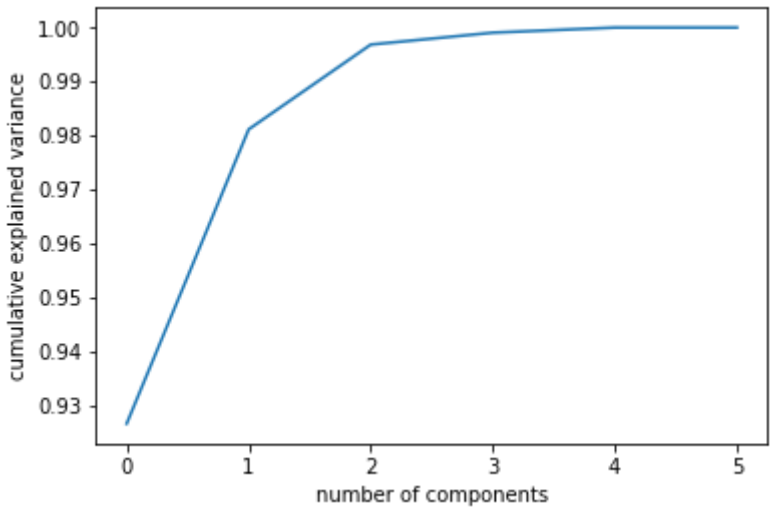
Out[183]: 300

PCA Analysis

In [184]:
```
X_4 = np.stack((camera1_4_x[:300], camera1_4_y[:300],
                camera2_4_x[9:309], camera2_4_y[9:309],
                camera3_4_x[:300], camera3_4_y[:300]))

# Centering the data
X_4 = X_4 - np.mean(X_4, axis=0)
X_4.shape
```

Out[184]: (6, 300)

In [185]:
```
pca4 = PCA().fit(X_4)
plt.plot(np.cumsum(pca4.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



In [186]:
```
plt.figure(figsize=(10,10))
plt.subplot(3,1,1)
plt.title("1st component")
plt.plot(pca4.components_[1,:])
plt.subplot(3,1,2)
plt.title("2nd component")
plt. plot(pca4.components_[2,:])
plt.subplot(3,1,3)
plt.title("3rd component")
plt. plot(pca4.components_[3,:])
plt.show()
```