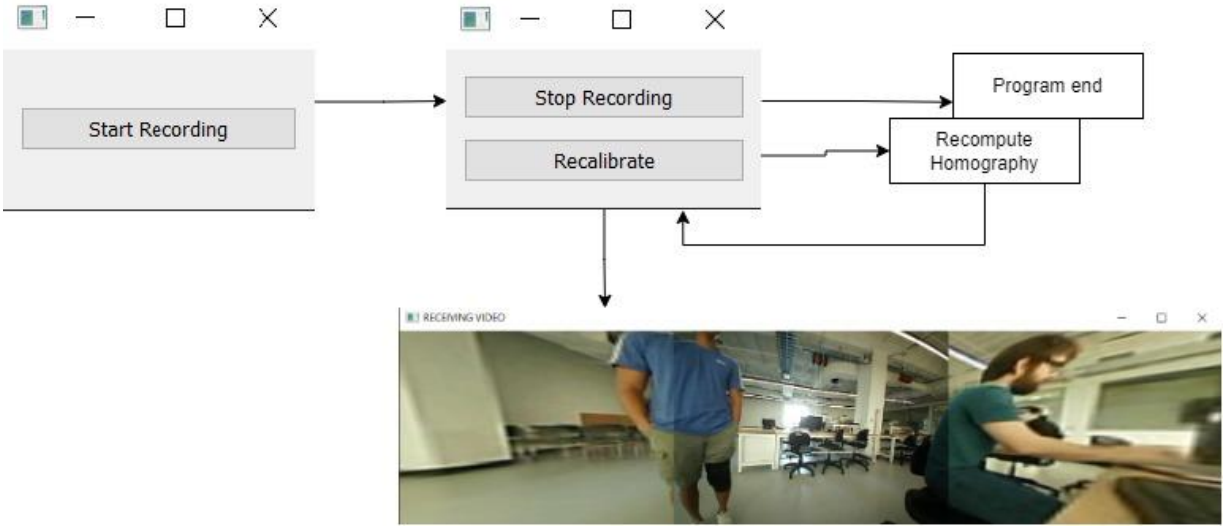
Expected Behavior:



GUI control flow diagram

Test Procedures:

/performance_testing/test_runner.py
The testing program will create a repeatable test for different stitching algorithms/optimizations. Runs for a set duration. Will be with cameras in a reproducible setting. Logs run settings and performance data to unique log file named by the date and time the test was run.
*added option to save a sample image

/performance_testing/log_parser.py
Reads the logs, plotting performance metrics related to the stitching.
*now shows raw images and stitched image for visual check of “goodness of stitch”

Functional Testing:

- python unittest framework explored but deemed to be non-priority
    - will likely come with refactor

Performance Testing:

- Focus on ensuring performance scales up properly with more cameras

    Parameters to test:

        Cameras: 2 - 6 cameras at a fixed 120° angle

Long duration runs (Memory usage)
Stitching implementations to test:
- Precomputed homography (4-6 cameras)

Optimizations:
- With/without CUDA acceleration
- With/without multithreading

User Testing:
- Receive non-group members feedback on GUI

Example Feedback	Implementation	Done
GUI should not crash when Homography fails	Added check for valid matrices before applying warp	YES
GUI should not crash when OpenCV stitcher fails		NO
GUI should alert user if serial (micro USB) not connected		NO

Beta Testing Reports:

Testing Report

Key Metrics

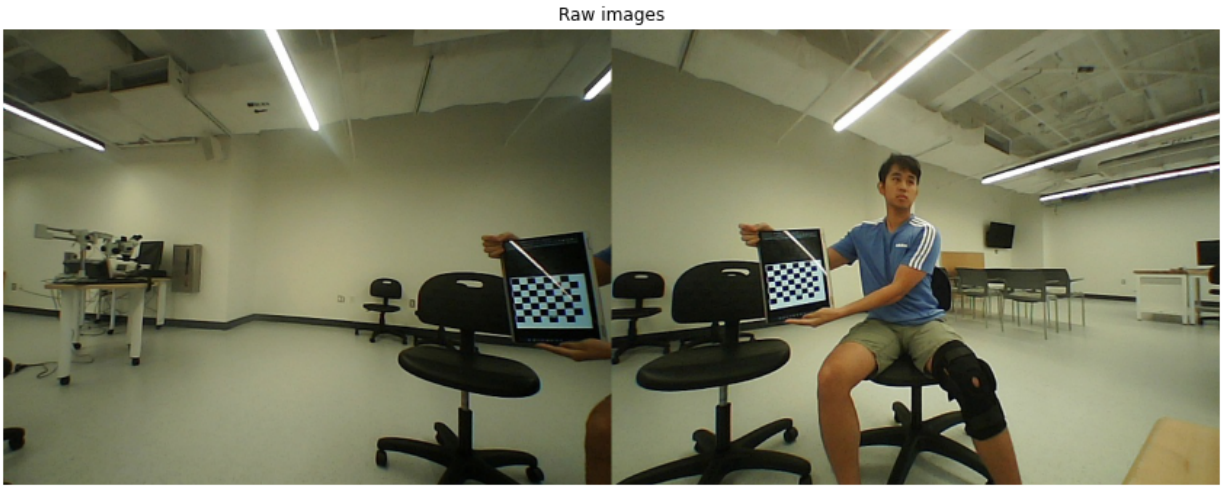
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
date_time = '2022-09-26_15.15.03'
log_file = f'logs/{date_time}.csv'

header_info = pd.read_csv(log_file, nrows=0)
print('Reading test log with Settings:')
for col in header_info.columns:
  print(' ', col)
```

```
Reading test log with Settings:
  Duration: 30
  Cameras: 2
  Stitching Algorithm: OpenCV
```

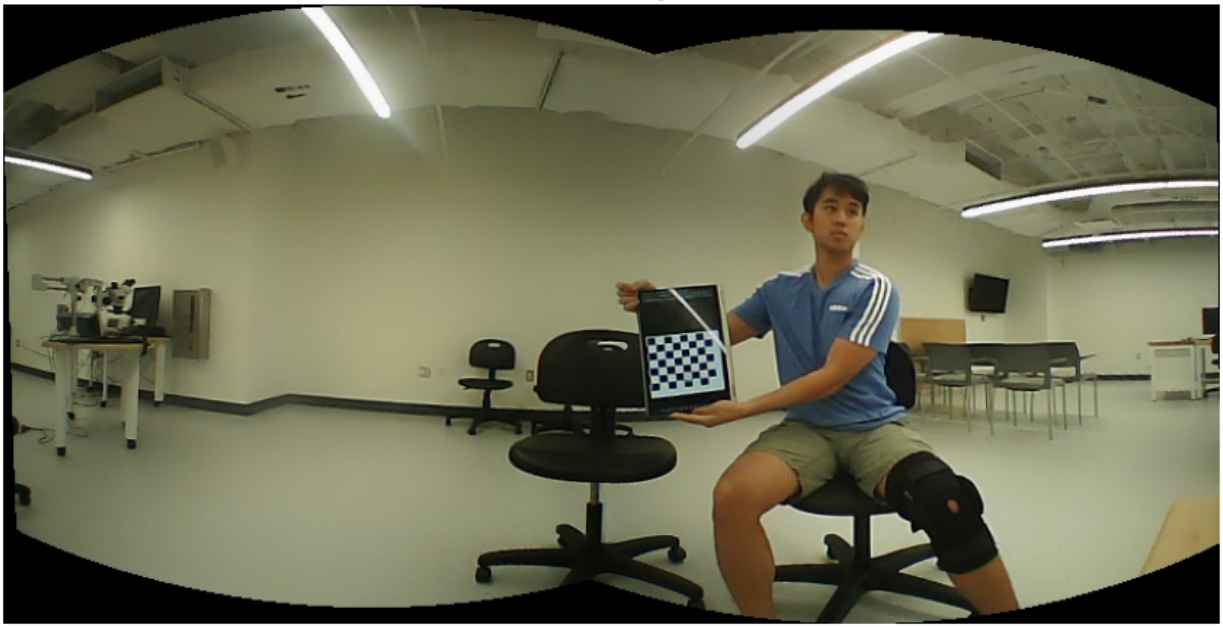
Sample Images

In [3]:
```
num_cameras = int(header_info.columns[1][-1])

from PIL import Image
import matplotlib.pyplot as plt

frames = [np.asarray(Image.open(f'logs/{date_time}_frame{i}.jpg')) for i in range(num_

plt.rcParams['figure.figsize'] = [15, 10]
plt.axis('off')
plt.title('Raw images')
plt.imshow(np.concatenate(frames, axis=1));
```



In [4]:
```
plt.axis('off')
plt.title('Stitched image')
plt.imshow(np.asarray(Image.open(f'logs/{date_time}_stitched.jpg')));
```

Stitched image



In [5]:
```
df = pd.read_csv(log_file, skiprows=[0])
df.head()
```

Out[5]:

	Start Time	End Time	Status
0	2510.421184	2510.593025	1
1	2510.597030	2510.790976	1
2	2510.793915	2511.237677	0
3	2511.275823	2511.694608	0
4	2511.697549	2511.989723	1

Successful Stitch Times

In [6]:
```
successful_stitches = df[df['Status'] == 0]
successful_stitches.head()
```

Out[6]:

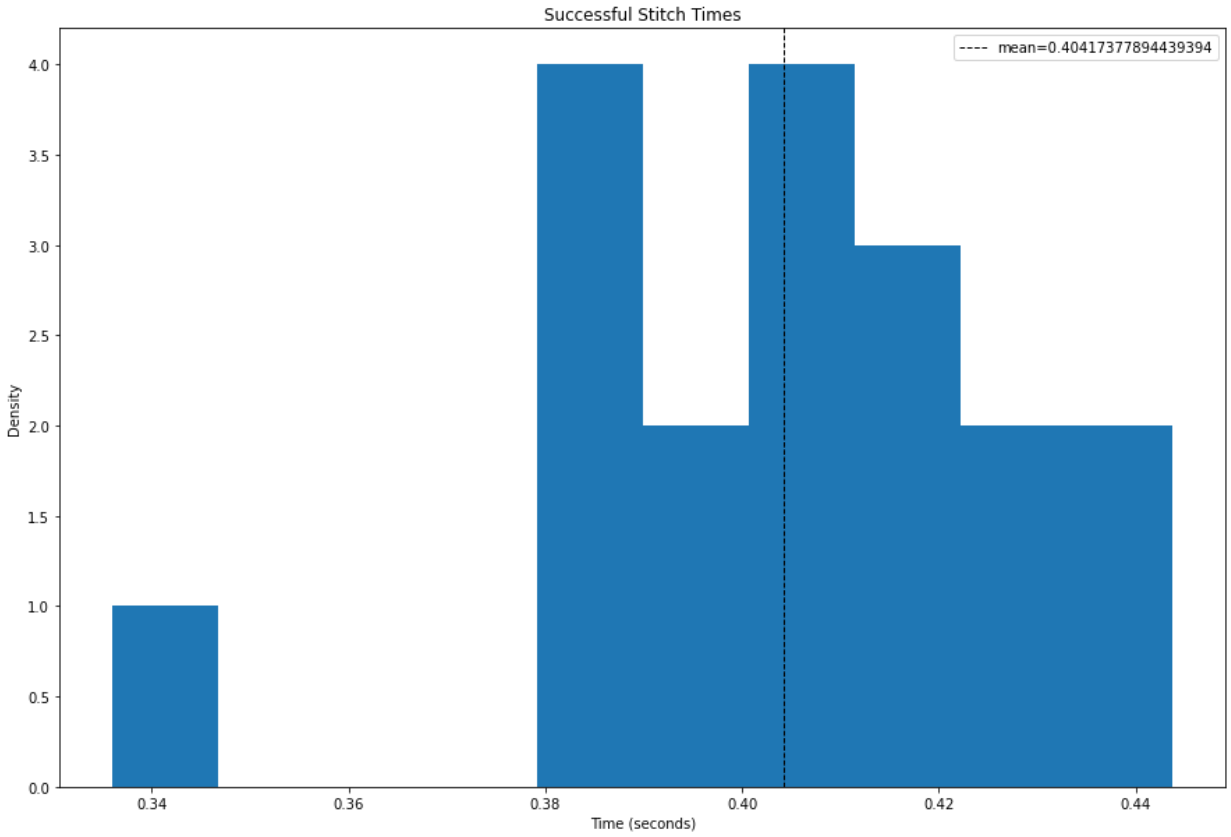
	Start Time	End Time	Status
2	2510.793915	2511.237677	0
3	2511.275823	2511.694608	0
19	2515.629448	2516.041723	0
20	2516.044597	2516.460307	0
22	2516.605291	2517.008636	0

In [7]:
```
stitch_times = successful_stitches['End Time'] - successful_stitches['Start Time']
print("Successful Stitch Time Statistics")
stitch_times.describe()
```

```
Successful Stitch Time Statistics
```

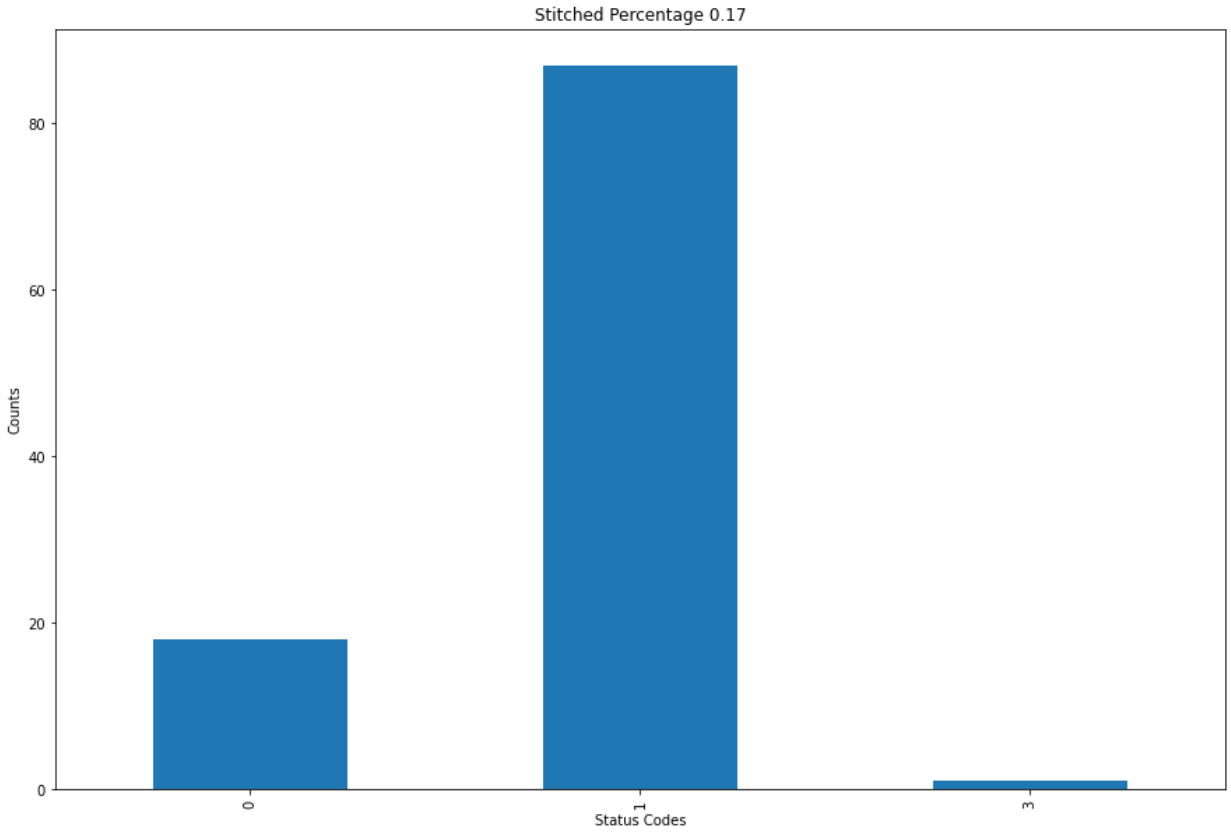
Out[7]:
```
count    18.000000
mean      0.404174
std       0.024893
min       0.336027
25%       0.390503
50%       0.403767
75%       0.418016
max       0.443762
dtype: float64
```

In [8]:
```
mean = np.mean(stitch_times)

plt.hist(stitch_times)
plt.axvline(mean, color='k', linestyle='dashed', linewidth=1,label=('mean='+str(mean))
plt.title('Successful Stitch Times')
plt.ylabel('Density')
plt.xlabel('Time (seconds)')
plt.legend();
```



Stitch Percentage

In [9]:
```
stitch_rate = round(successful_stitches.shape[0] / df.shape[0], 2)
stitch_rate

vc = df['Status'].value_counts().sort_index()
vc.plot(kind='bar')
plt.xlabel('Status Codes')
plt.ylabel('Counts')
plt.title(f'Stitched Percentage {stitch_rate}');
```


Stitched Percentage 0.17
Counts
0
20
40
60
80
0
1
3
Status Codes

In [ ]:

Testing Report

Key Metrics

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
date_time = '2022-09-26_17.07.08'
log_file = f'logs/{date_time}.csv'

header_info = pd.read_csv(log_file, nrows=0)
print('Reading test log with Settings:')
for col in header_info.columns:
  print(' ', col)
```

```
Reading test log with Settings:
  Duration: 30
  Cameras: 3
  Stitching Algorithm: Homography
```

Sample Images

In [3]:
```
num_cameras = int(header_info.columns[1][-1])

from PIL import Image
import matplotlib.pyplot as plt

frames = [np.asarray(Image.open(f'logs/{date_time}_frame{i}.jpg')) for i in range(num_

plt.rcParams['figure.figsize'] = [15, 10]
plt.axis('off')
plt.title('Raw images')
plt.imshow(np.concatenate(frames, axis=1));
```

In [4]:
```
plt.axis('off')
plt.title('Stitched image')
plt.imshow(np.asarray(Image.open(f'logs/{date_time}_stitched.jpg')));
```

Stitched image

In [5]:
```
df = pd.read_csv(log_file, skiprows=[0])
df.head()
```

Out[5]:

	Start Time	End Time	Status
0	1611.178659	1611.199747	0
1	1611.224962	1611.244671	0
2	1611.289771	1611.312466	0
3	1611.356953	1611.376992	0
4	1611.424957	1611.452870	0

Successful Stitch Times

In [6]:
```
successful_stitches = df[df['Status'] == 0]
successful_stitches.head()
```

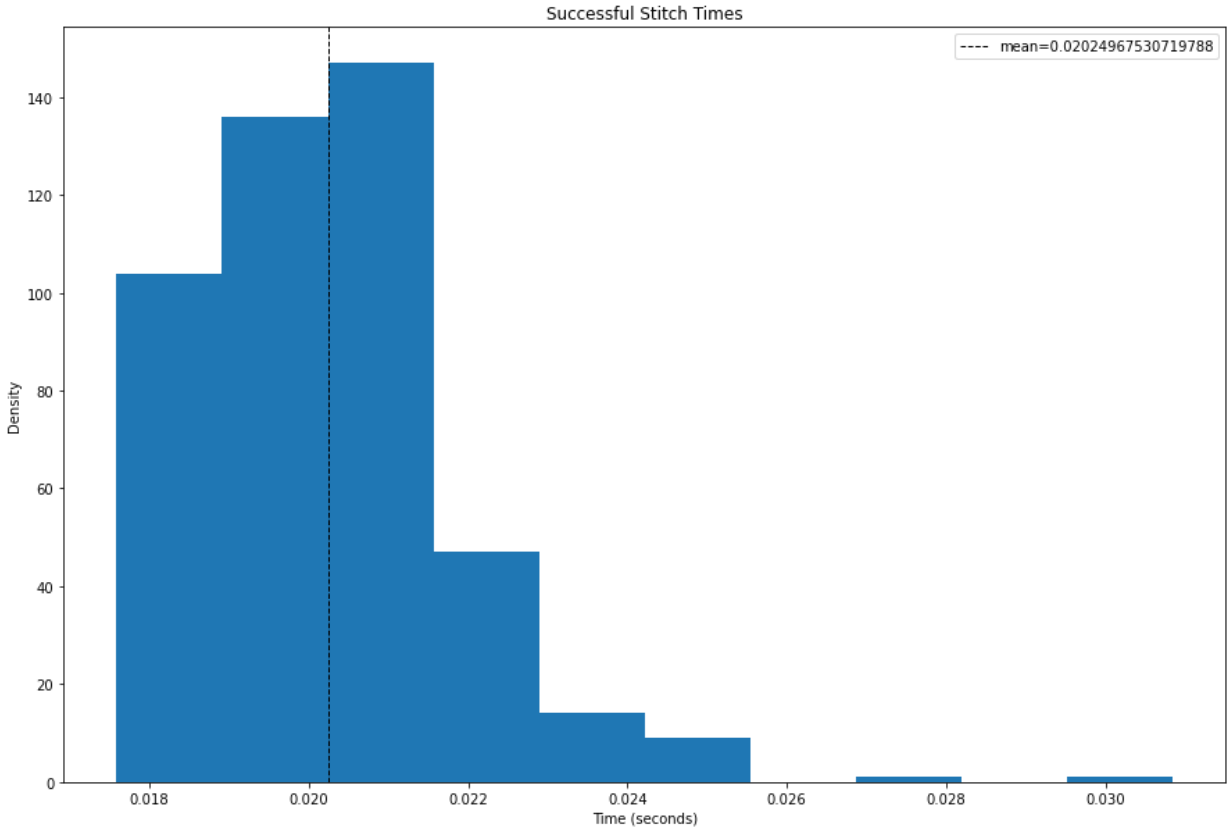
Out[6]:

	Start Time	End Time	Status
0	1611.178659	1611.199747	0
1	1611.224962	1611.244671	0
2	1611.289771	1611.312466	0
3	1611.356953	1611.376992	0
4	1611.424957	1611.452870	0

In [7]:
```
stitch_times = successful_stitches['End Time'] - successful_stitches['Start Time']
print("Successful Stitch Time Statistics")
stitch_times.describe()
```

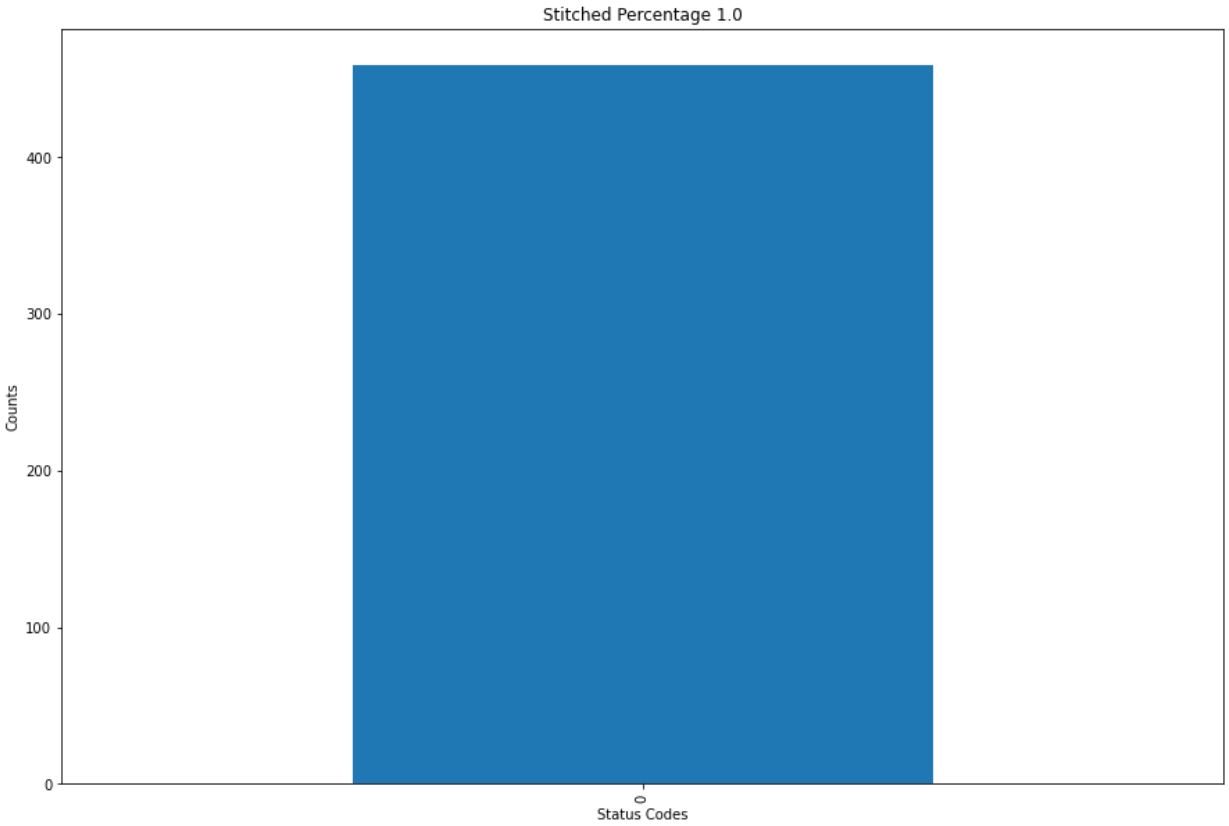
Out[7]:
```
Successful Stitch Time Statistics
count    459.000000
mean       0.020250
std        0.001594
min        0.017586
25%        0.019083
50%        0.020140
75%        0.020998
max        0.030847
dtype: float64
```

In [8]:
```
mean = np.mean(stitch_times)

plt.hist(stitch_times)
plt.axvline(mean, color='k', linestyle='dashed', linewidth=1,label=('mean='+str(mean))
plt.title('Successful Stitch Times')
plt.ylabel('Density')
plt.xlabel('Time (seconds)')
plt.legend();
```



Stitch Percentage

In [9]:
```
stitch_rate = round(successful_stitches.shape[0] / df.shape[0], 2)
stitch_rate

vc = df['Status'].value_counts().sort_index()
vc.plot(kind='bar')
plt.xlabel('Status Codes')
plt.ylabel('Counts')
plt.title(f'Stitched Percentage {stitch_rate}');
```


Stitched Percentage 1.0
Counts
400
300
200
100
0
0
Status Codes

In [ ]: