 Run on Google Colab  View on Github

The notebook uses three methods to extract the center and other relevant info about the drop in the image. These methods are:

1. Cropping Image Dynamically
2. Subtracting all the Images from a Reference Image
3. Fitting an Ellipse

We have also made some preliminary analysis for all the three methods.

All the code can be found on github repository, specifically on the folder: codes

> Codes still need refactoring!

Imports

In [1]:
```
import matplotlib.pyplot as plt
from misc_tools import *
from run import Run
plt.rcdefaults()
```

In [2]:
```
# Image Directory
IMAGE_DIRECTORY = "../data/images1"
DATA_DIRECTORY = "../data/results"
r = Run(IMAGE_DIRECTORY, DATA_DIRECTORY)
```

Using Dynamic Cropping

In [3]:
```
# Strating Crop coordinates
crop_1 = (150, 40, 40, 40)
crop_2 = (120, 250, 50, 50)
crop_3 = (250, 230, 40, 40)

# Image Positions
positions_1 = (0, 70)
positions_2 = (75, 100)
positions_3 = (106, 155)

df_dc = r.dynamic_cropping(
    crop_1,
    crop_2,
    crop_3,
    positions_1,
    positions_2,
    positions_3,
    save=True, #Saving the generated DataFrame
    file_name="dc",
    plot=False, #whether to plot
    strict=True, #Using the equation of line
    crop_included=False, #Excluding the crop coordinates
)
```

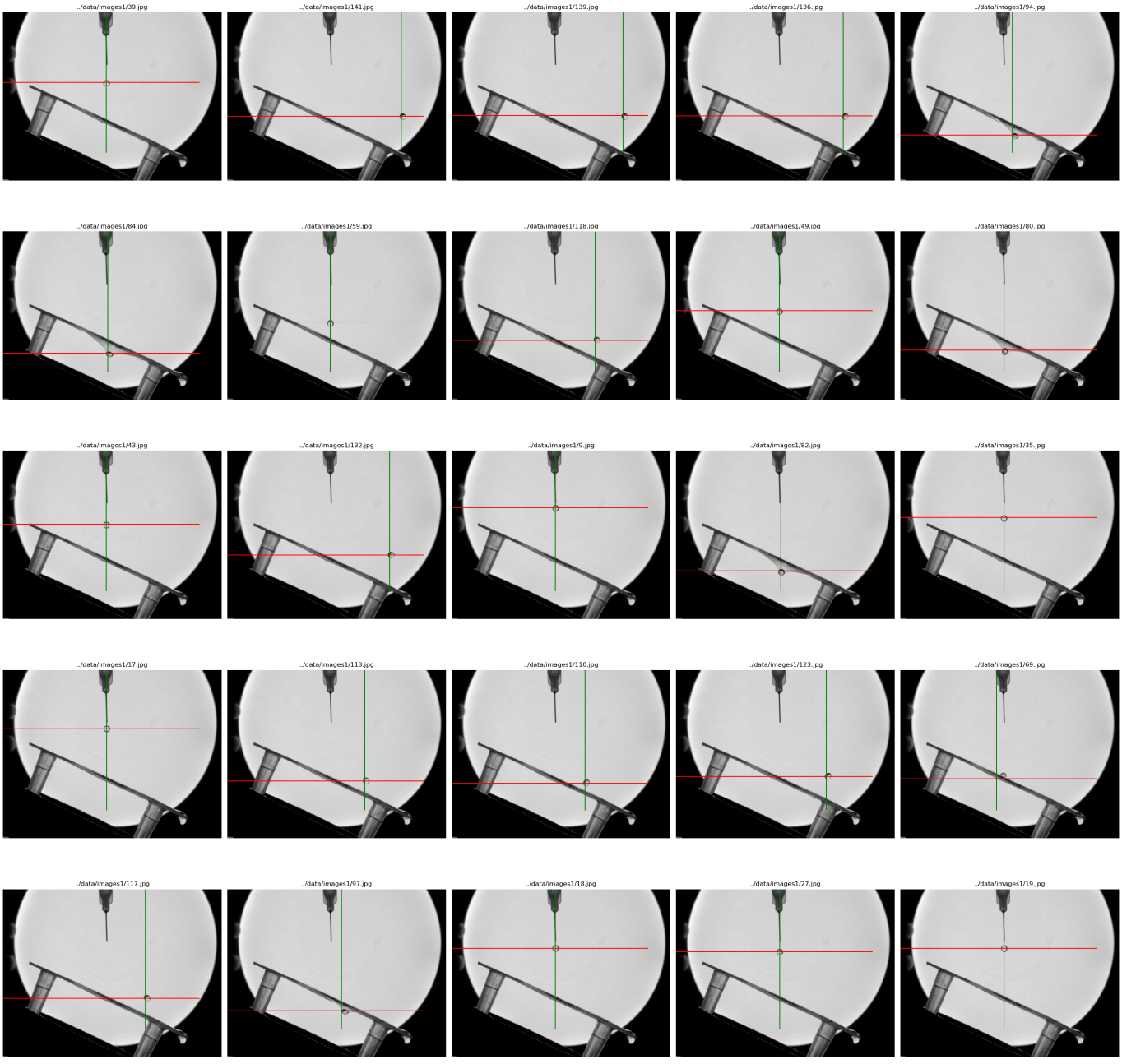
```
Starting dynamic cropping...
First step...
```

```
Second step...
Third step...
Saving to csv...
Done!
```

Getting some Samples

In [4]:

```
get_samples(len_samples = 25,df_path=df_dc,
 file_name="using_dc", title="Using DC")
```



Using DC

Preliminary Analysis

In [5]:

```
#Smoothing the data
df_dc = smoothen(df_dc)
df_dc
```

Out[5]:

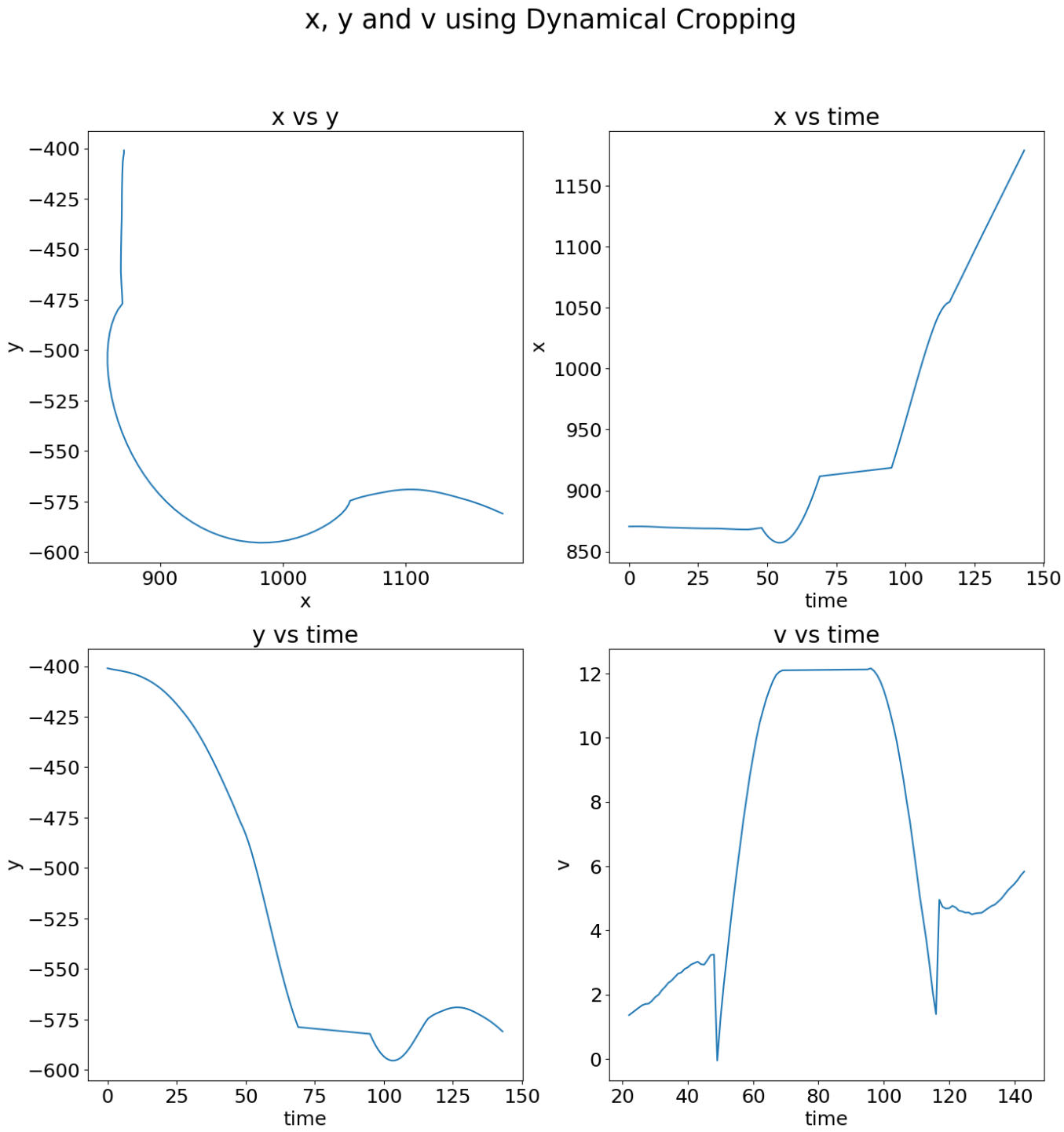
	id	x	y	r1	r2	r	vy	vx	v
0	../data/images1/0.jpg	870.587824	401.000000	10.0	13.0	11.0	NaN	NaN	NaN
1	../data/images1/1.jpg	870.618264	401.301569	10.0	13.0	11.0	NaN	NaN	NaN
2	../data/images1/2.jpg	870.639818	401.579694	10.0	13.0	11.0	NaN	NaN	NaN

	id	x	y	r1	r2	r	vy	vx	v
3	../data/images1/3.jpg	870.652864	401.810550	10.0	13.0	11.0	NaN	NaN	NaN
4	../data/images1/4.jpg	870.657780	402.039705	10.0	13.0	11.0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
139	../data/images1/150.jpg	1160.746077	576.424655	11.0	8.0	9.0	1.916808	4.826432	5.347604
140	../data/images1/151.jpg	1165.276423	577.477595	11.0	8.0	9.0	2.202874	4.853091	5.446056
141	../data/images1/152.jpg	1169.841369	578.584610	11.0	8.0	9.0	2.435243	4.888637	5.569656
142	../data/images1/153.jpg	1174.439592	579.800151	11.0	8.0	9.0	2.751182	4.963698	5.715468
143	../data/images1/154.jpg	1179.000000	581.000000	11.0	8.0	9.0	3.000000	5.000000	5.830952

119 rows × 9 columns

In [6]:

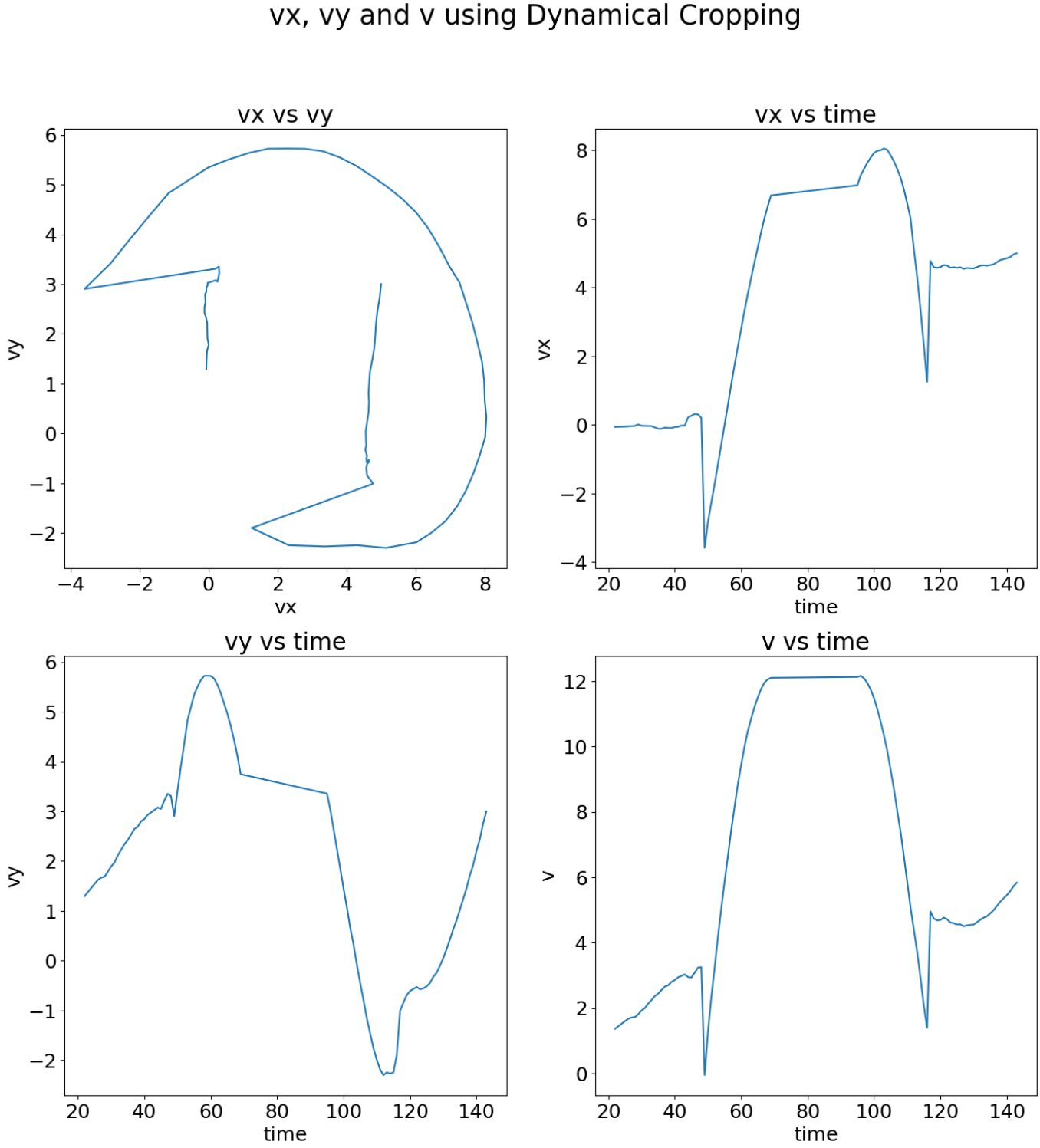
```
cols = ["x", "y", "v"]
plot_all(df_dc, title="x, y and v using Dynamical Cropping", cols = cols)
```



There are some missing values between the timestamps 60 and 80.

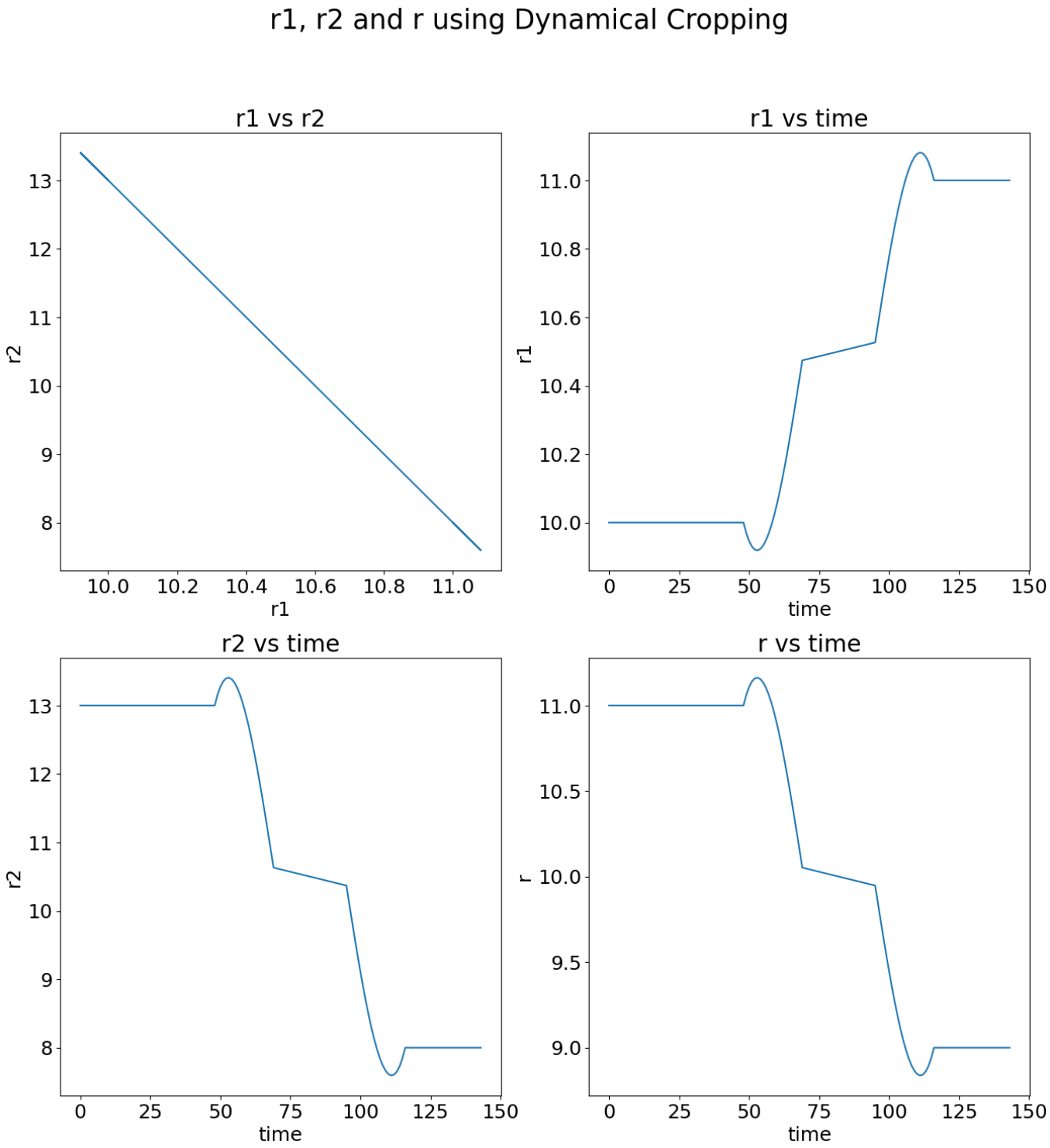
In [7]:

```
cols = ["vx", "vy", "v"]
plot_all(df_dc, title="vx, vy and v using Dynamical Cropping", cols = cols)
```



In [8]:

```
cols = ["r1", "r2", "r"]
plot_all(df_dc, title="r1, r2 and r using Dynamical Cropping", cols = cols)
```



By Subtracting Images

In [9]:
```
df_si = r.subtracting_images(subtract=True, strict=False,
            plot=False)
```

```
Getting list of images...
Extracting data from images...
Error on:  ../data/images1/71.jpg
Error on:  ../data/images1/102.jpg
Error on:  ../data/images1/160.jpg
Error on:  ../data/images1/161.jpg
Error on:  ../data/images1/162.jpg
Error on:  ../data/images1/163.jpg
Error on:  ../data/images1/164.jpg
Error on:  ../data/images1/165.jpg
Error on:  ../data/images1/166.jpg
Error on:  ../data/images1/167.jpg
Error on:  ../data/images1/168.jpg
Error on:  ../data/images1/169.jpg
```

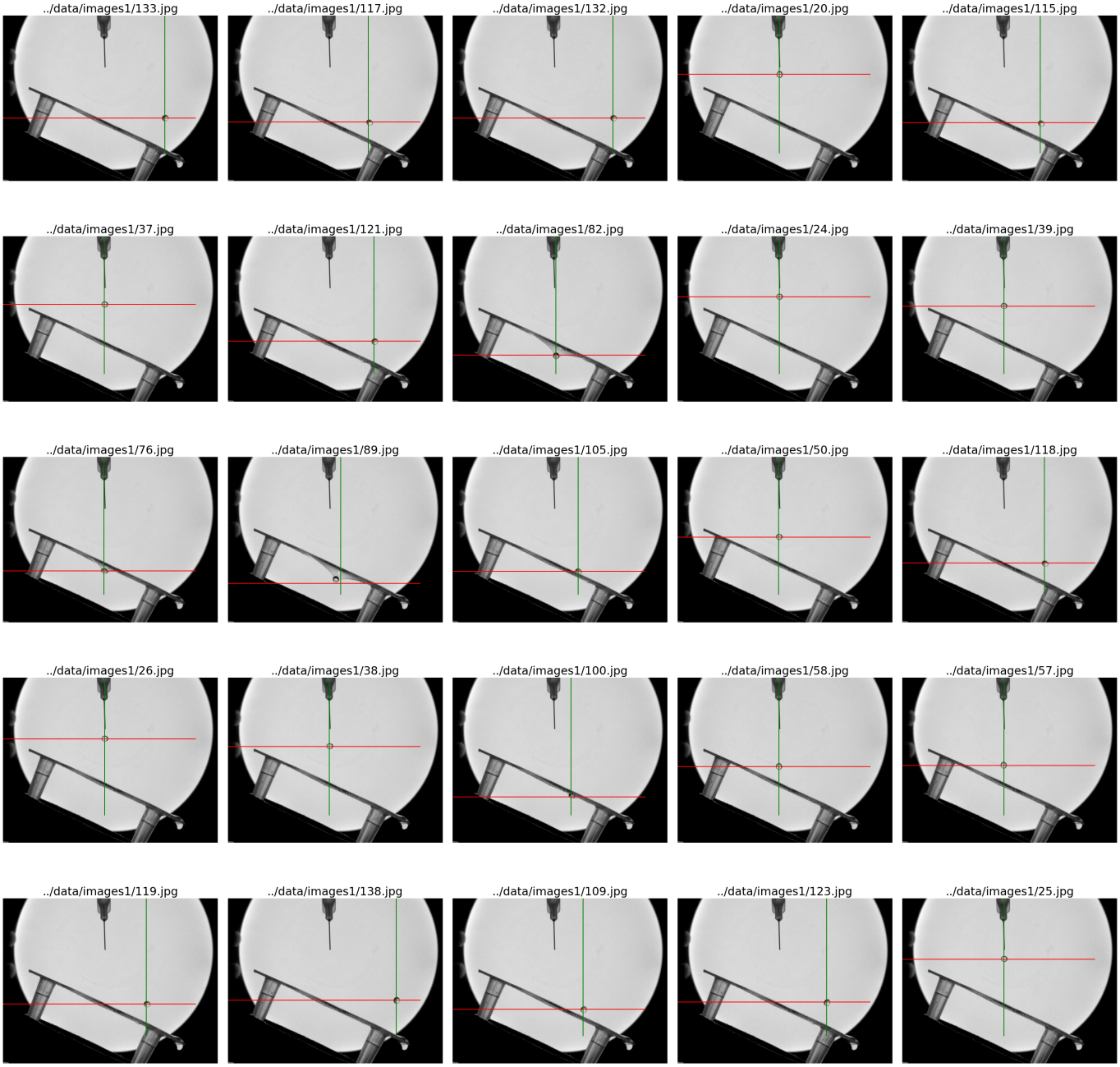
```
Error on:  ../data/images1/170.jpg
Saving to csv...
Done!
```

There are about a dozen of images where the method is not working.

Getting Some Samples

In [10]:

```
get_samples(len_samples = 25,df_path=df_si,
 file_name="using_si", title="Using SI")
```



Using SI

Preliminary Analysis

In [11]:

```
#Smoothing the data
df_si = smoothen(df_si)
df_si
```

Out[11]:

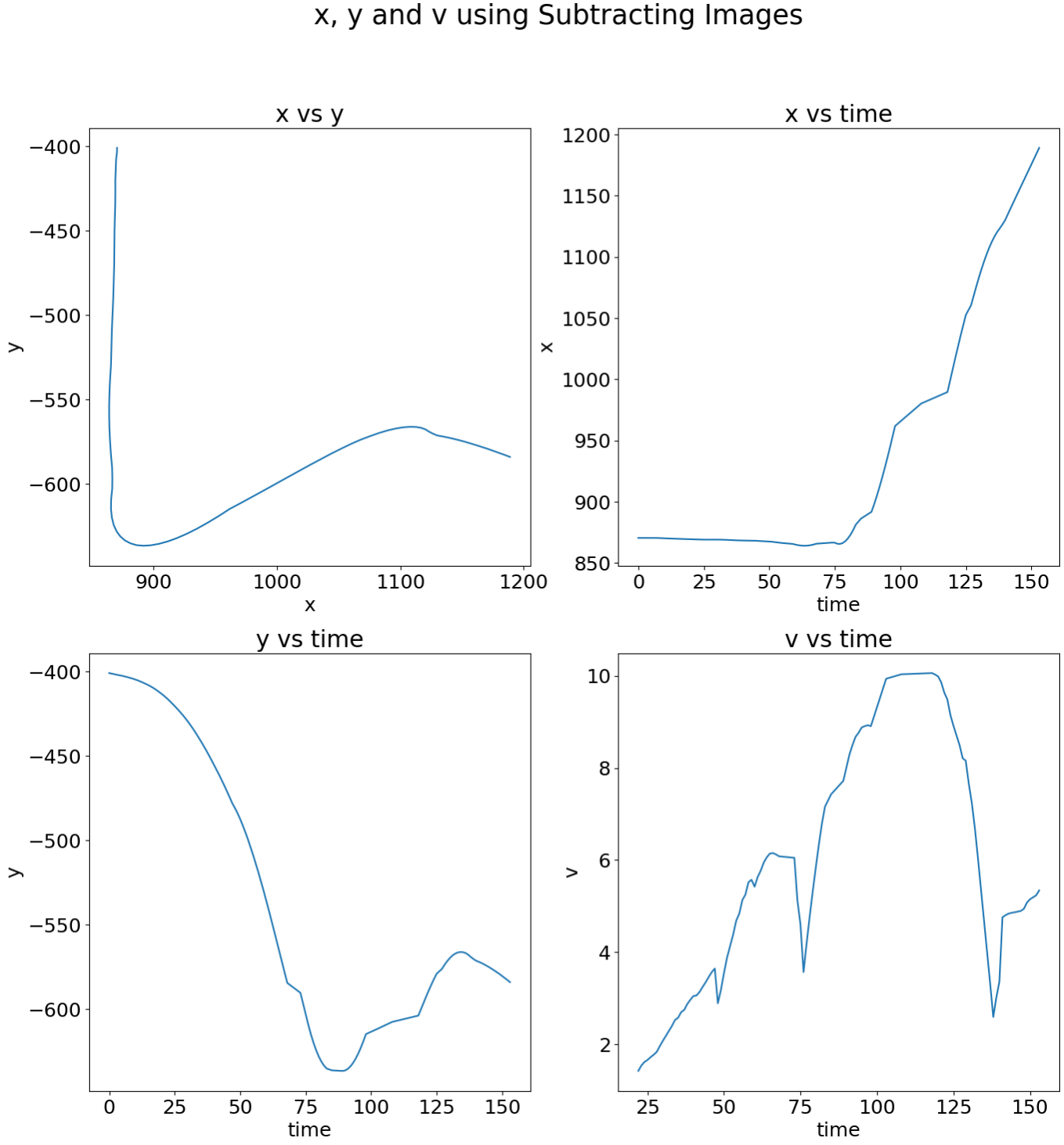
	id	x	y	r1	r2	r	vy	vx
0	../data/images1/0.jpg	870.497069	401.000000	10.000000	11.731707	10.646436	NaN	NaN
1	../data/images1/1.jpg	870.496502	401.347136	10.052940	11.732085	10.643978	NaN	NaN

	id	x	y	r1	r2	r	vy	vx
2	../data/images1/2.jpg	870.525808	401.677822	10.105880	11.764228	10.636604	NaN	NaN
3	../data/images1/3.jpg	870.546228	402.006239	10.127812	11.820382	10.624315	NaN	NaN
4	../data/images1/4.jpg	870.558140	402.307052	10.157497	11.823029	10.607109	NaN	NaN
...	...	...	...	...	...	...	...	...
149	../data/images1/151.jpg	1170.823218	579.191151	9.607865	10.217432	9.536774	1.020798	4.923426
150	../data/images1/152.jpg	1175.376631	580.361505	9.594252	10.207790	9.493099	1.085649	4.994706
151	../data/images1/153.jpg	1179.918510	581.539232	9.586689	10.219134	9.449991	1.112876	5.033844
152	../data/images1/154.jpg	1184.447154	582.745131	9.616185	10.189072	9.407071	1.140480	5.071469
153	../data/images1/155.jpg	1189.000000	584.000000	9.612970	10.226130	9.403101	1.206466	5.177349

128 rows × 9 columns

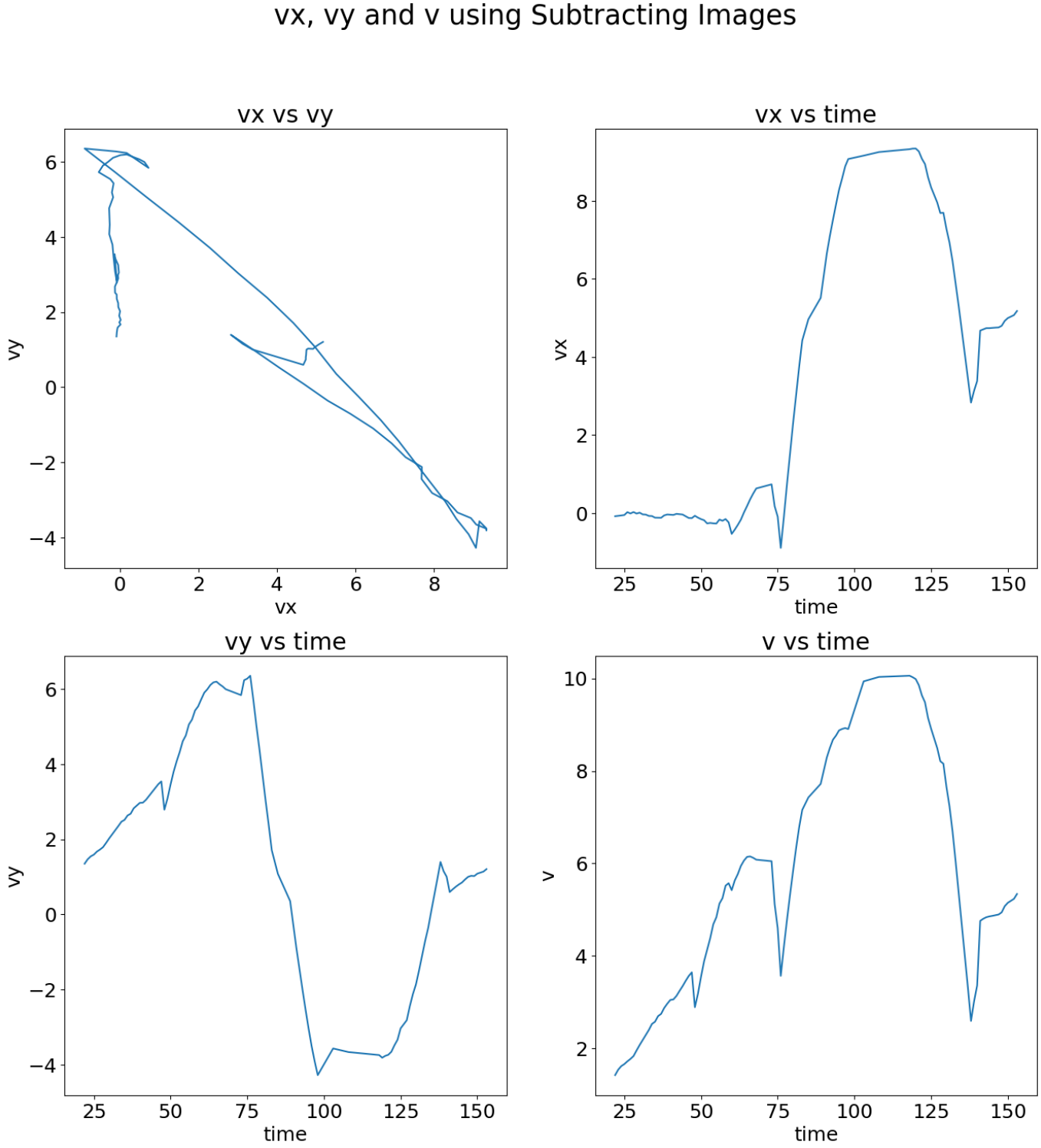
In [12]:

```
cols = ["x", "y", "v"]
plot_all(df_si, title="x, y and v using Subtracting Images", cols = cols)
```



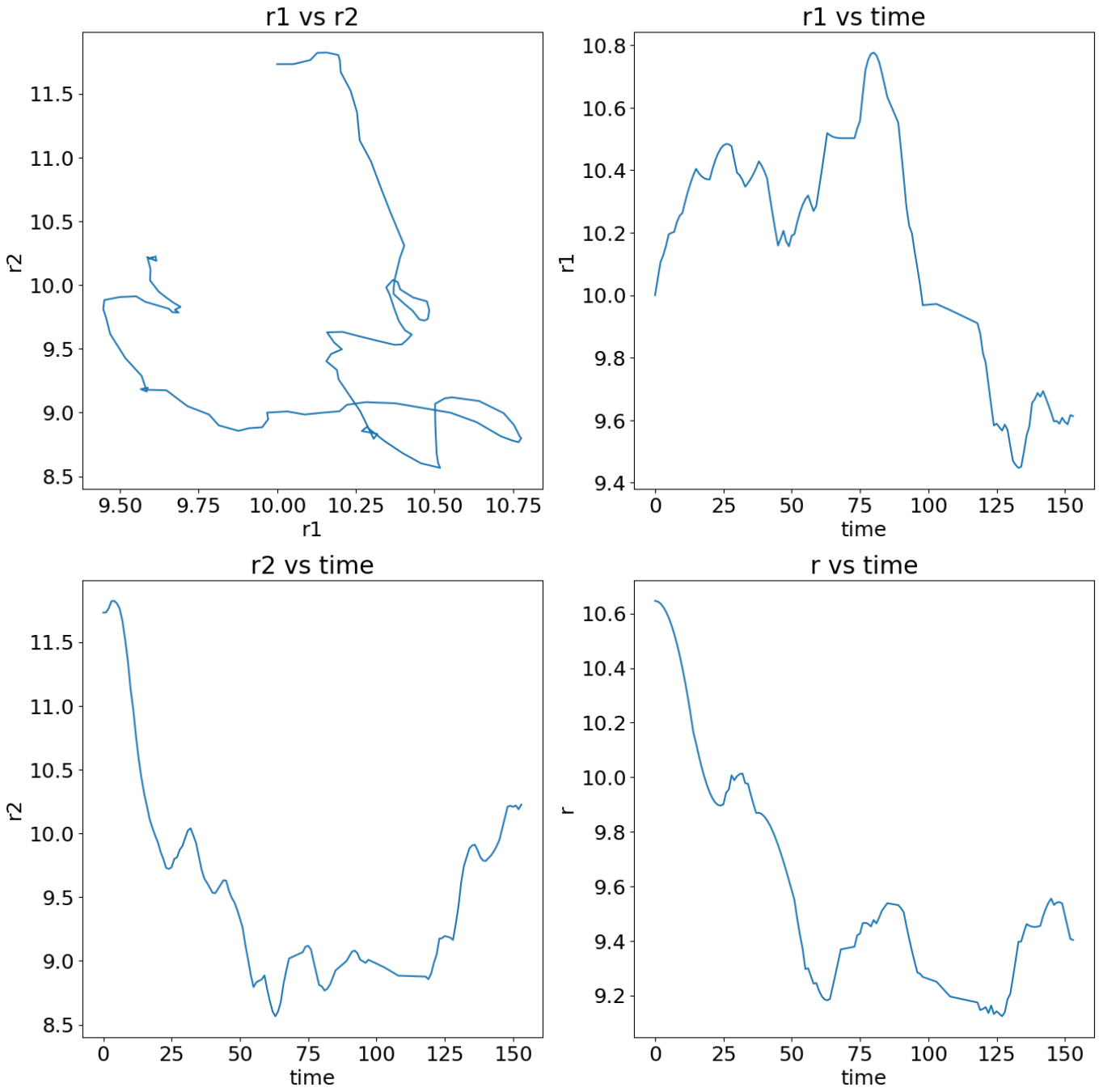
In [13]:

```
cols = ["vx", "vy", "v"]
plot_all(df_si, title="vx, vy and v using Subtracting Images", cols = cols)
```



In [14]:

```
cols = ["r1", "r2", "r"]
plot_all(df_si, title="r1, r2 and r using Subtracting Images", cols = cols)
```



By Fitting Ellipse

In [15]:
```
df_ap = r.all_points(plot=False, crop_included=True)
```

```
Getting list of images...
Extracting data from images...
Error on:  ../data/images1/71.jpg
Error on:  ../data/images1/72.jpg
Error on:  ../data/images1/102.jpg
Error on:  ../data/images1/159.jpg
Error on:  ../data/images1/160.jpg
Error on:  ../data/images1/161.jpg
Error on:  ../data/images1/162.jpg
Error on:  ../data/images1/163.jpg
Error on:  ../data/images1/164.jpg
Error on:  ../data/images1/165.jpg
Error on:  ../data/images1/166.jpg
Error on:  ../data/images1/167.jpg
Error on:  ../data/images1/168.jpg
```

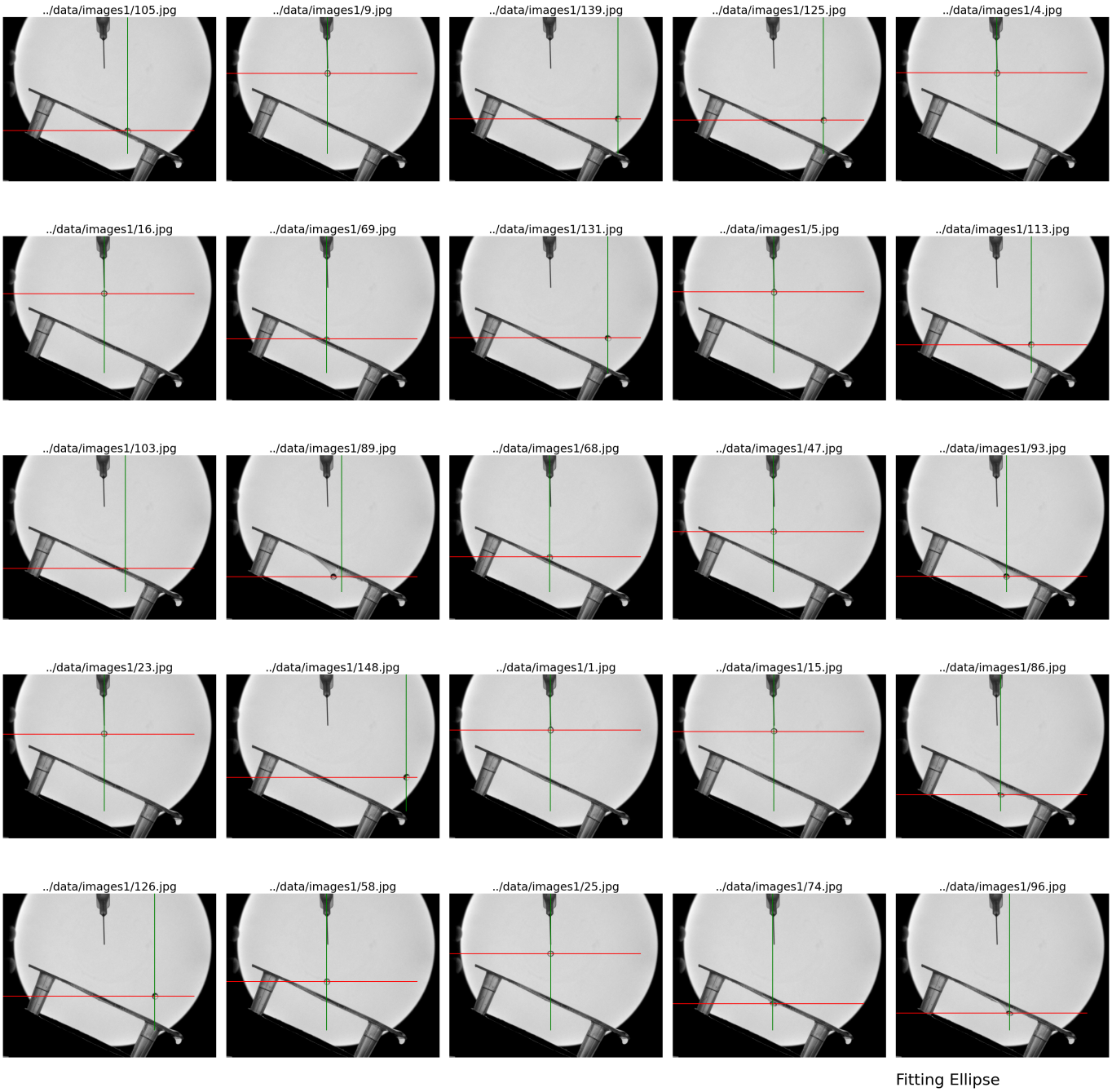
```
Error on:  ../data/images1/169.jpg
Error on:  ../data/images1/170.jpg
Saving to csv...
Done!
```

Again, there are about a dozen of images where the method is not working.

Getting Some Samples

In [16]:

```
get_samples(len_samples = 25,df_path=df_ap,
 file_name="using_ap", title="Fitting Ellipse")
```



Fitting Ellipse

Preliminary Analysis

In [17]:

```
#Smoothing the data
df_ap = smoothen(df_ap)
df_ap
```

Out[17]:

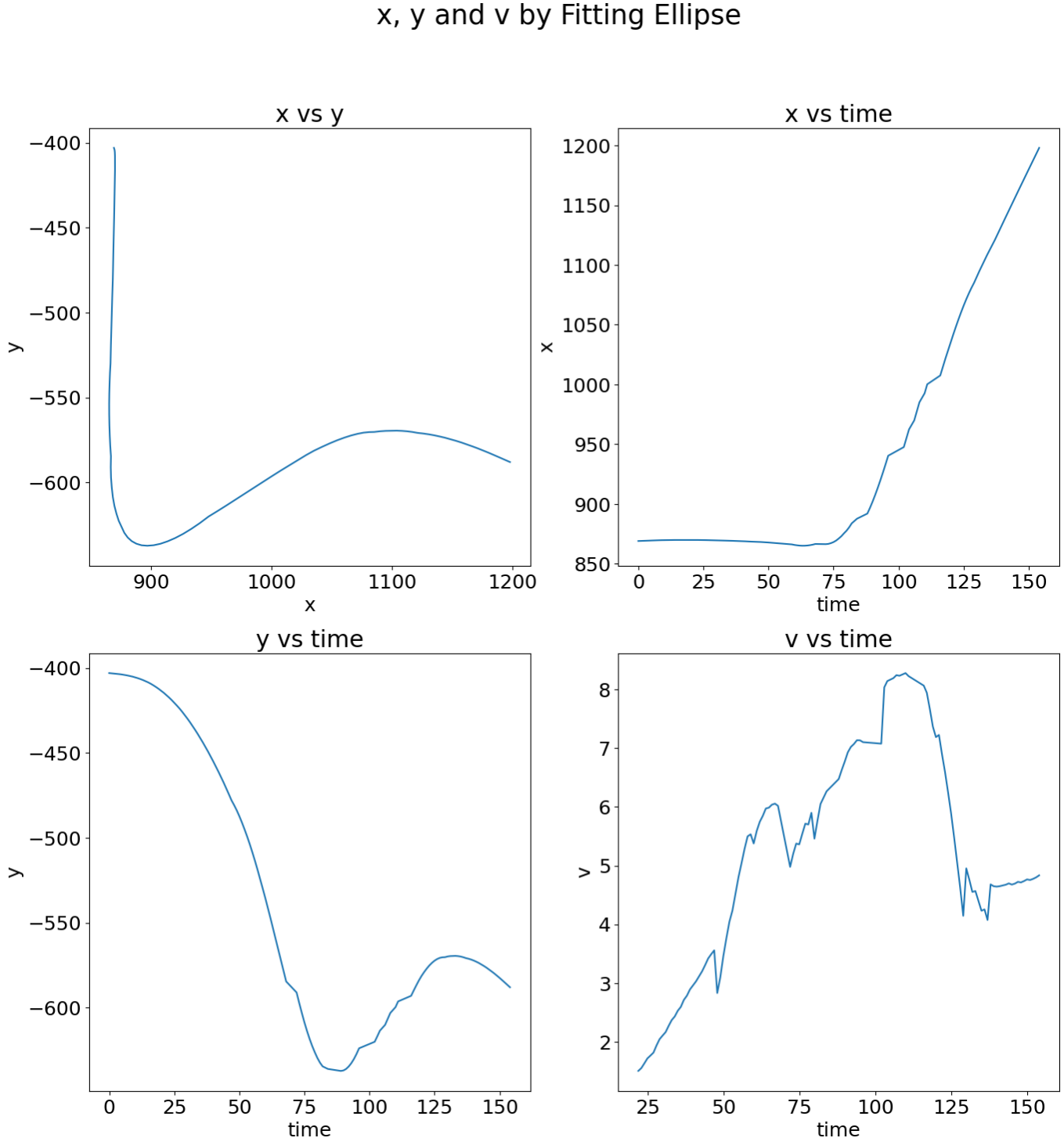
	id	x	y	r1	r2	theta	r	vy
0	../data/images1/0.jpg	869.000000	403.000000	8.000000	11.239176	0.251548	9.969370	NaN
1	../data/images1/1.jpg	869.070524	403.147287	8.213651	11.240499	0.330962	9.969749	NaN

	id	x	y	r1	r2	theta	r	vy	
2	../data/images1/2.jpg	869.179429	403.278881	8.425789	11.244470	0.340577	9.970883	NaN	
3	../data/images1/3.jpg	869.256192	403.440726	8.603895	11.313103	0.403795	9.972774	NaN	
4	../data/images1/4.jpg	869.331064	403.639629	8.847230	11.306863	0.403282	9.975421	NaN	
...	...	...	...	...	...	...	...	...	
150	../data/images1/153.jpg	1180.119304	582.475137	9.773870	10.373417	1.199565	9.943846	1.487238	4.5
151	../data/images1/154.jpg	1184.573076	583.823785	9.758366	10.364719	1.223732	9.903573	1.613916	4.5
152	../data/images1/155.jpg	1189.052562	585.211760	9.714124	10.419739	1.234078	9.896956	1.776139	4.4
153	../data/images1/156.jpg	1193.525808	586.595198	9.741917	10.437323	1.237128	9.892985	1.902628	4.4
154	../data/images1/157.jpg	1198.000000	588.000000	9.701834	10.495368	1.213515	9.891662	2.000000	4.5

137 rows × 10 columns

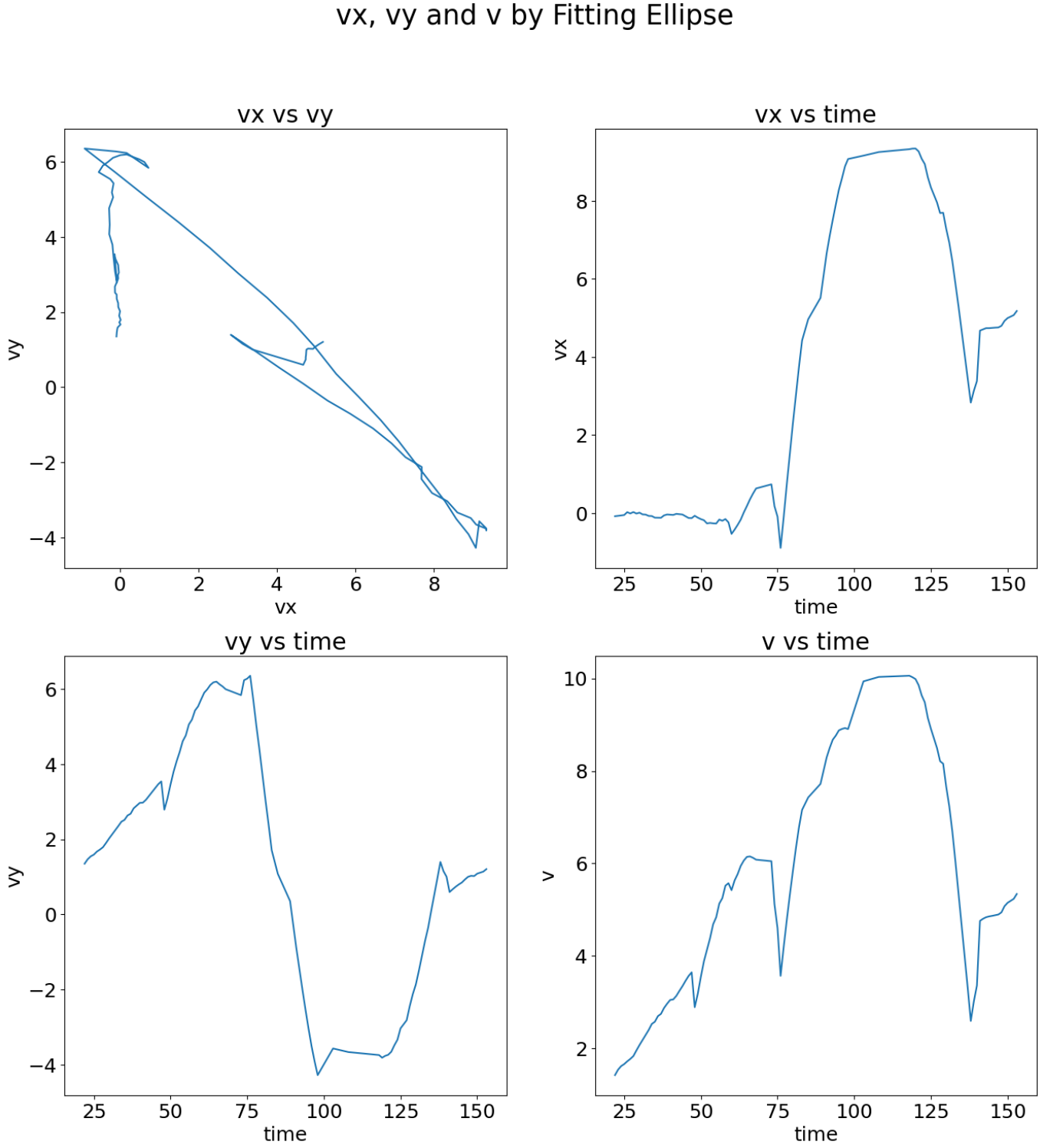
In [18]:

```
cols = ["x", "y", "v"]
plot_all(df_ap, title="x, y and v by Fitting Ellipse", cols = cols)
```



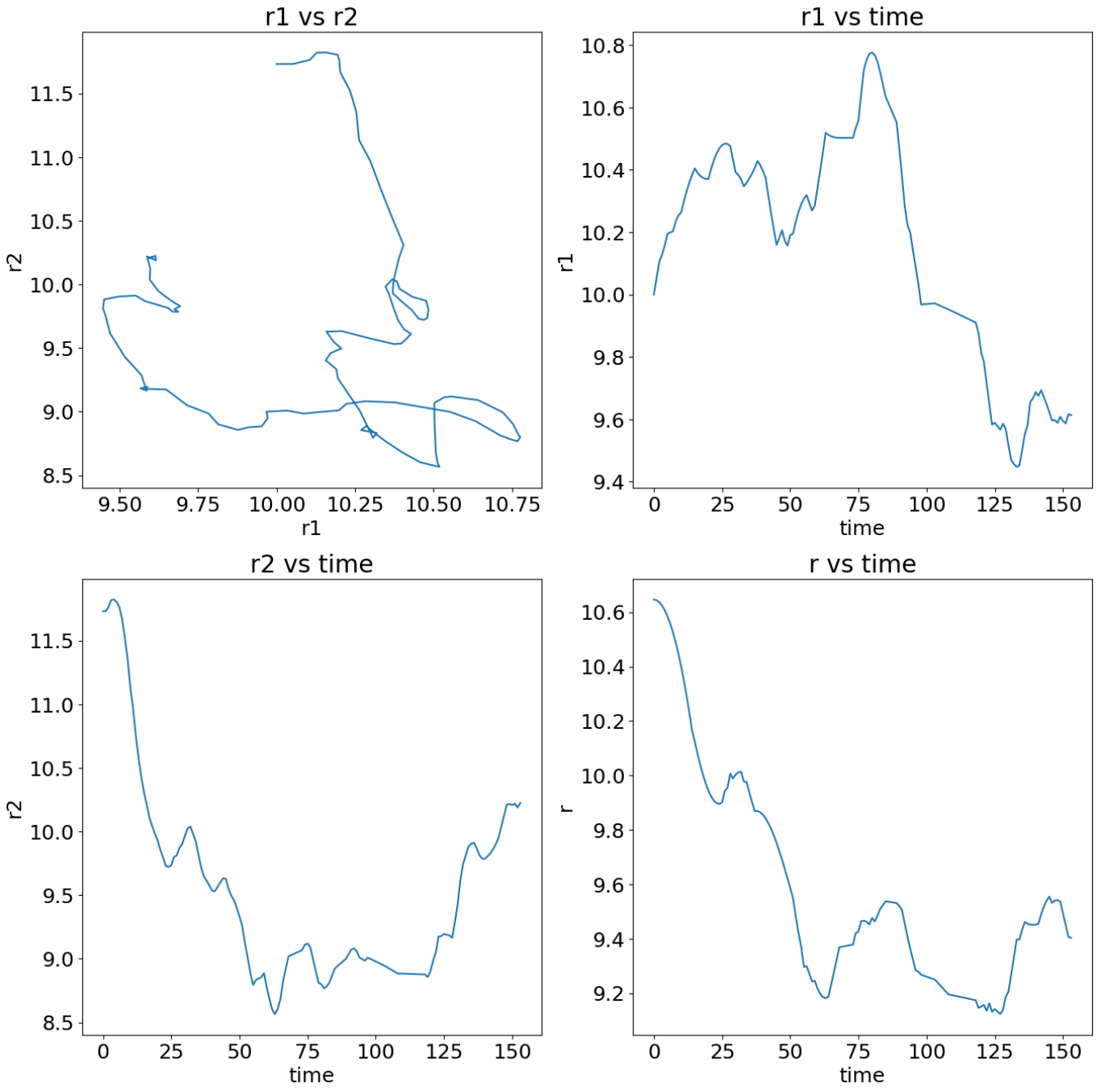
In [19]:

```
cols = ["vx", "vy", "v"]
plot_all(df_si, title="vx, vy and v by Fitting Ellipse", cols = cols)
```



In [20]:

```
cols = ["r1", "r2", "r"]
plot_all(df_si, title="r1, r2 and r by Fitting Ellipse", cols = cols)
```


r1, r2 and r by Fitting Ellipse
r1 vs r2
r2
r1
r1 vs time
r1
time
r2 vs time
r2
time
r vs time
r
time