My GPU partition:
salloc -–partition=gpu-v100 -–gpus=1 -–time=3:00:00
I used 4 threads per block, however, I have 3 blocks for 3 dimensions (row, column and frame indexes)

#	Width	Height	Num_frames	Serial Runtime (s)	CUDA Runtime (s)	Speed up
1	1024	576	10	2.9519	0.002579	1144
2	1280	720	15	6.8507	0.005820	1177
3	1920	1080	20	20.3719	0.016258	1253
4	2560	1440	25	45.0304	0.036545	1232
5	3200	1800	30	84.2178	0.067618	1245
6	3840	2160	35	141.0836	0.116907	1206
7	1024	576	40	11.5371	0.009685	1191
8	1920	1080	45	45.3812	0.040395	1123
9	2560	1440	50	89.3217	0.078210	1142
10	4096	2160	60	256.3924	0.216558	1183

Note that the size of the width and height does not increase in the order from 1 - 10, I recycled the input for 7-9 with different numbers of frames.

Bonus Assignment: Comparing runtime after changing the variables to type float

#	Width	Height	Num_frames	CUDA with float runtime (s)	CUDA with double runtime (s)
1	1024	576	10	0.002732	0.002579
2	1280	720	15	0.004827	0.005820
3	1920	1080	20	0.054845	0.016258
4	2560	1440	25	0.028917	0.036545
5	3200	1800	30	0.157634	0.067618
6	3840	2160	35	0.085083	0.116907
7	1024	576	40	0.007265	0.009685
8	1920	1080	45	0.073306	0.040395
9	2560	1440	50	0.057238	0.078210
10	4096	2160	60	0.150980	0.216558

In GPU computing, using float instead of double typically enhances performance, especially for large data sets like high-resolution images or numerous video frames. This improvement occurs because floats require less memory and GPUs can process them faster due to their simpler, less precise nature. However, there are cases in the table where using floats can actually degrade performance. I believe that in these cases the task requires high precision of the data. In such cases, the lower precision of floats can lead to inefficiencies or the need for additional compensatory computations, which may offset the typical benefits.