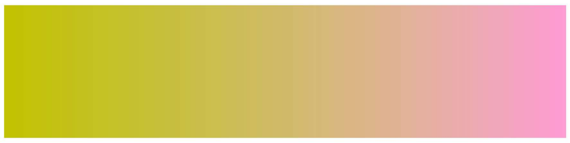
```
import numpy as np
import matplotlib.pyplot as plt
```

Colour Vulnerabilities

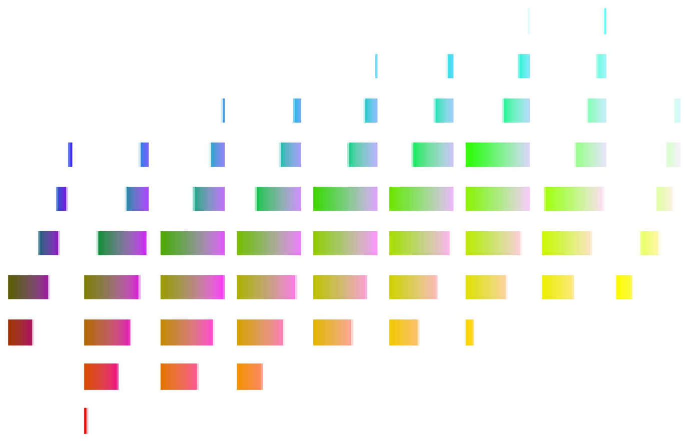
```
i = 0
y = 0
x = 0.5

colors = []
for b in range(256):
  b = b/255
  if b<0.03928:
    b = b/12.92
  else:
    b = ((b+0.055)/1.055)**2.4
  r = 0.693037*b + 1.07782*x + 0.770856*y/320
  g = -0.306963*b + 1.07782*x - 0.229144*y/320
  if b < 0.03928/12.92:
    B = round(b*12.92*255,0)
  else:
    B = round((b**(1/2.4)*1.055-0.055)*255,0)
  if g < 0.03928/12.92:
    G = round(g*12.92*255,0)
  else:
    G = round((g**(1/2.4)*1.055-0.055)*255,0)
  if r < 0.03928/12.92:
    R = round(r*12.92*255,0)
  else:
    R = round((r**(1/2.4)*1.055-0.055)*255,0)
  if max(R,G,B) < 256 and min(R,G,B)>-1:
    colors.append([R,G,B])
  else:
    colors.append([255,255,255])
colors = np.array([colors]*50, dtype=int)
plt.imshow(colors),plt.axis('off')
```

```
(<matplotlib.image.AxesImage at 0x7f2babe7cf90>, (-0.5, 255.5, 49.5, -0.5))
```



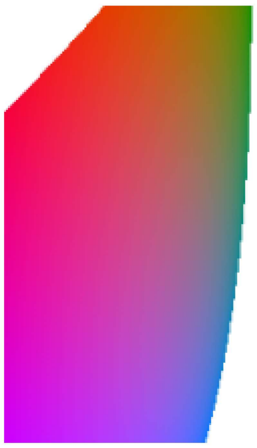
```
plt.imshow([[[194, 194, 0]]*1024]*768),plt.axis('off')
plt.show()
plt.imshow([[[255, 156, 214]]*1024]*768),plt.axis('off')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt

i = 0
ys = [-600, -500, -400, -300, -200, -100, 0, 100, 200, 300]
xs = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
for y in ys:
  for x in xs:
    i+=1
    colors = []
    for b in range(256):
      b = b/255
      if b<0.03928:
        b = b/12.92
      else:
        b = ((b+0.055)/1.055)**2.4
      r = 0.693037*b + 1.07782*x + 0.770856*y/320
      g = -0.306963*b + 1.07782*x - 0.229144*y/320
      if b < 0.03928/12.92:
        B = round(b*12.92*255,0)
      else:
        B = round((b**(1/2.4)*1.055-0.055)*255,0)
      if g < 0.03928/12.92:
        G = round(g*12.92*255,0)
      else:
        G = round((g**(1/2.4)*1.055-0.055)*255,0)
      if r < 0.03928/12.92:
        R = round(r*12.92*255,0)
      else:
        R = round((r**(1/2.4)*1.055-0.055)*255,0)
      if max(R,G,B) < 256 and min(R,G,B)>-1:
          colors.append([R,G,B])
        else:
          colors.append([255,255,255])
      colors = np.array([colors]*100, dtype=int)
      plt.subplot(len(ys), len(xs), i)
      plt.imshow(colors),plt.axis('off')
```



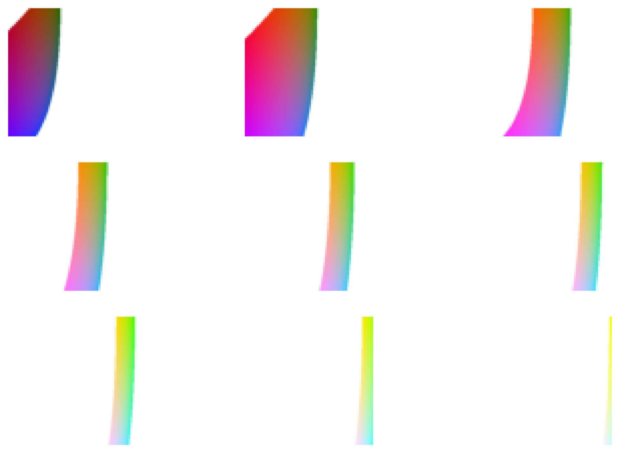
```
import numpy as np
import matplotlib.pyplot as plt

i = 0
x = 0.2

colors = []
for b in range(256):
  b = b/255
  if b<0.03928:
    b = b/12.92
  else:
    b = ((b+0.055)/1.055)**2.4
  colors_inner = []
  for g in range(256):
    g = g/255
    if g<0.03928:
      g = g/12.92
    else:
      g = ((g+0.055)/1.055)**2.4
    r = -0.339605*b - 3.36406*g + 4.70367*x
    if b < 0.03928/12.92:
      B = round(b*12.92*255,0)
    else:
      B = round((b**(1/2.4)*1.055-0.055)*255,0)
    if g < 0.03928/12.92:
      G = round(g*12.92*255,0)
    else:
      G = round((g**(1/2.4)*1.055-0.055)*255,0)
    if r < 0.03928/12.92:
      R = round(r*12.92*255,0)
    else:
      R = round((r**(1/2.4)*1.055-0.055)*255,0)
    if max(R,G,B) > 255 or min(R,G,B)<0:
      colors_inner.append([255, 255, 255])
      elif R/(R+G+B+0.0000001) > 0.8:
        colors_inner.append([255, 255, 255])
      else:
        colors_inner.append([R,G,B])
    colors.append(colors_inner.copy())
  colors = np.array(colors, dtype=int)
  plt.imshow(colors),plt.axis('off')
```

```
(<matplotlib.image.AxesImage at 0x7f2ba6bf9350>, (-0.5, 255.5, 255.5, -0.5))
```



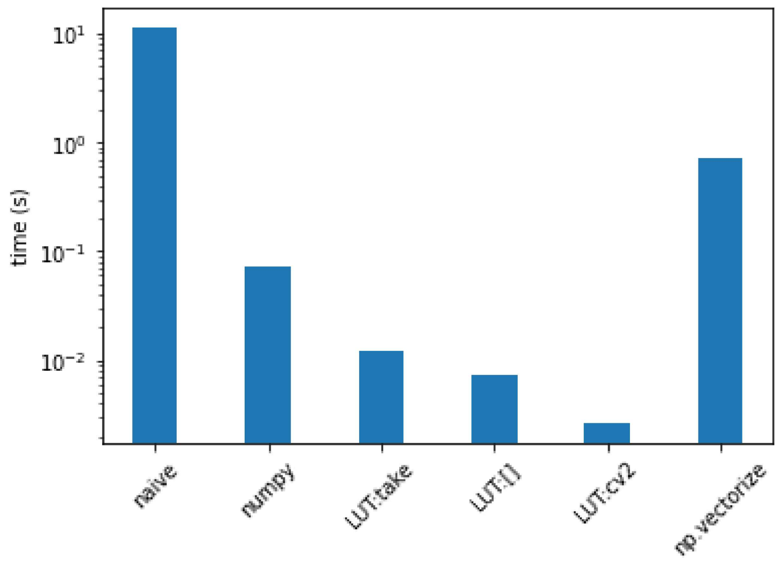
```
plt.imshow([[[8, 144, 0]]*1024]*768),plt.axis('off')
plt.show()
plt.imshow([[[204, 0, 255]]*1024]*768),plt.axis('off')
plt.show()
plt.imshow([[[9, 116, 255]]*1024]*768),plt.axis('off')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt

i = 0
xs = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

for x in xs:
  i+=1
  colors = []
  for b in range(256):
    b = b/255
    if b<0.03928:
      b = b/12.92
    else:
      b = ((b+0.055)/1.055)**2.4
    colors_inner = []
    for g in range(256):
      g = g/255
      if g<0.03928:
        g = g/12.92
      else:
        g = ((g+0.055)/1.055)**2.4
      r = -0.339605*b - 3.36406*g + 4.70367*x
      if b < 0.03928/12.92:
        B = round(b*12.92*255,0)
      else:
        B = round((b**(1/2.4)*1.055-0.055)*255,0)
      if g < 0.03928/12.92:
        G = round(g*12.92*255,0)
      else:
        G = round((g**(1/2.4)*1.055-0.055)*255,0)
      if r < 0.03928/12.92:
        R = round(r*12.92*255,0)
      else:
        R = round((r**(1/2.4)*1.055-0.055)*255,0)
      if max(R,G,B) > 255 or min(R,G,B)<0:
        colors_inner.append([255, 255, 255])
      elif R/(R+G+B+0.0000001) > 0.8:
        colors_inner.append([255, 255, 255])
      else:
        colors_inner.append([R,G,B])
    colors.append(colors_inner.copy())
  colors = np.array(colors, dtype=int)
  plt.subplot(3, 3, i)
  plt.imshow(colors),plt.axis('off')
```



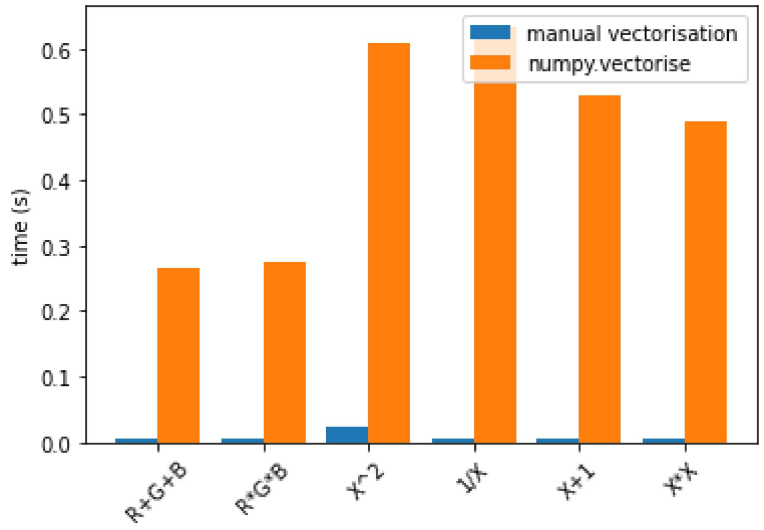
Performance of numerical operations

```
names = ["naive", "numpy", "LUT:take", "LUT:[]", "LUT:cv2", "np.vectorize"]
ys = [11.52, 0.072, 0.0124, 0.0073, 0.0026, 0.72]
X_axis = np.arange(len(names))
plt.bar(X_axis, ys, 0.4)
plt.xticks(X_axis, names)
plt.xticks(rotation=45)
plt.ylabel("time (s)")
plt.yscale('log')
plt.show()
# plt.bar(X_axis[1:-1], ys[1:-1], 0.4)
# plt.xticks(X_axis[1:5], names[1:5])
# plt.show()
```



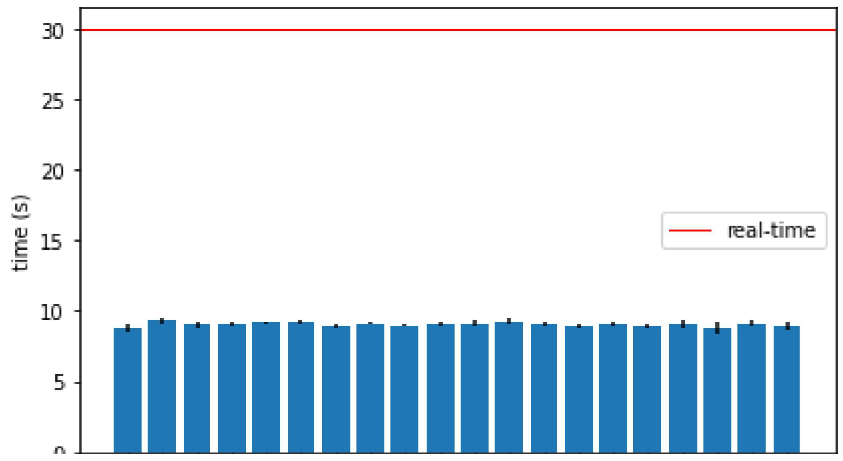
```
names = ["R+G+B", "R*G*B", "X^2", "1/X", "X+1", "X*X"]
xs = [0.26534008979797363, 0.273817777633667, 0.6099138259887695, 0.634380578994751, 0.529
ys = [0.005125522613525391, 0.006552219390869141, 0.023012399673461914, 0.0040125846862792
```

```
X_axis = np.arange(len(names))
plt.bar(X_axis - 0.2, ys, 0.4, label="manual vectorisation")
plt.bar(X_axis + 0.2, xs, 0.4, label="numpy.vectorise")
plt.xticks(X_axis, names)
plt.xticks(rotation=45)
plt.ylabel("time (s)")
plt.legend()
plt.show()
```

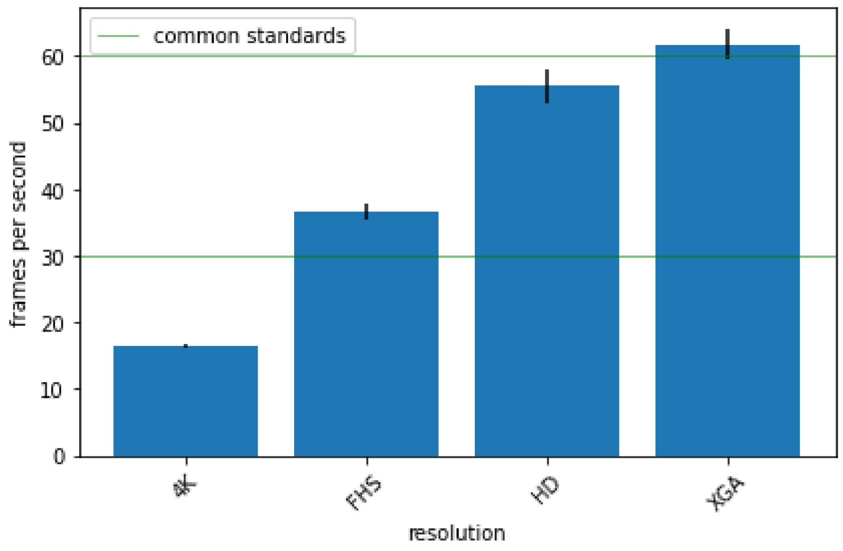


Evaluation

```
fig, ax = plt.subplots()

videos = ['Video '+str(x+1) for x in range(20)]
x_pos = np.arange(len(videos))
CTEs = [8.77917, 9.30953, 8.9981, 9.08952, 9.18551, 9.22973, 8.98588, 9.12478, 8.99393, 9.
error = [0.22254, 0.24678, 0.14766, 0.11119, 0.06957, 0.15746, 0.13573, 0.12824, 0.13694,

ax.bar(x_pos, CTEs, yerr=error)
ax.set_ylabel('time (s)')
ax.set_xticks(x_pos)
ax.set_xticklabels(videos)
plt.axhline(y=30,linewidth=1, color='red', label='real-time')
plt.xticks(rotation=45)
plt.legend()

# Save the figure and show
plt.tight_layout()
plt.show()
```

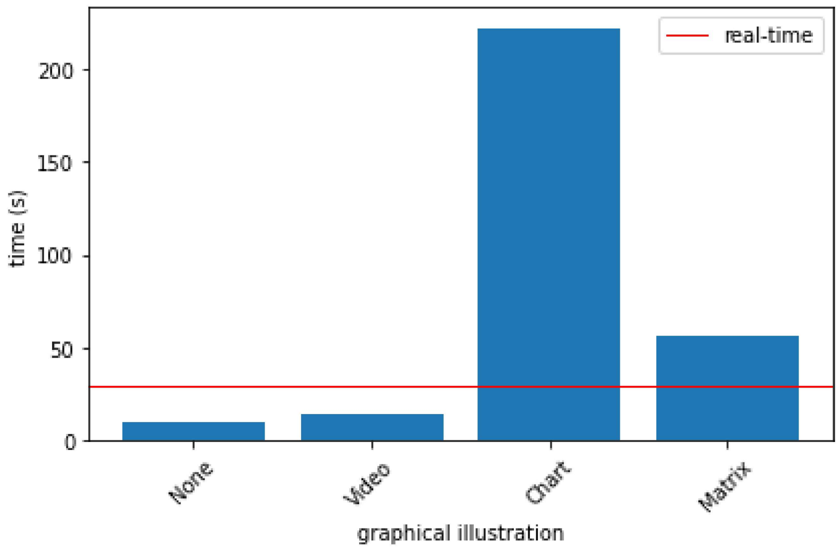


```
fig, ax = plt.subplots()

videos = ['4K', 'FHS', 'HD', 'XGA']
x_pos = np.arange(len(videos))
CTEs = [16.4, 36.68, 55.44, 61.68]
error = [0.3286, 1.1479, 2.659, 2.327]

ax.bar(x_pos, CTEs, yerr=error)
ax.set_ylabel('frames per second')
ax.set_xlabel('resolution')
ax.set_xticks(x_pos)
ax.set_xticklabels(videos)
plt.xticks(rotation=45)
plt.axhline(y=30,linewidth=0.5, color='green')
plt.axhline(y=60,linewidth=0.5, color='green', label='common standards')
plt.legend()

# Save the figure and show
plt.tight_layout()
plt.show()
```

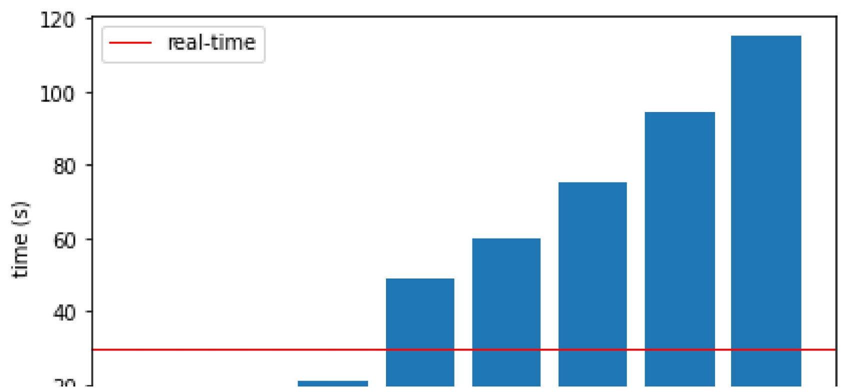


```
fig, ax = plt.subplots()

videos = ["None", "Video", "Chart", "Matrix"]
x_pos = np.arange(len(videos))
```

```
CTEs = [9.92178862094879, 14.8111939907073, 221.601954555511, 56.2163502931594]

ax.bar(x_pos, CTEs)
ax.set_ylabel('time (s)')
ax.set_xlabel('graphical illustration')
ax.set_xticks(x_pos)
ax.set_xticklabels(videos)
plt.axhline(y=30,linewidth=1, color='red', label='real-time')
plt.xticks(rotation=45)
plt.legend()

# Save the figure and show
plt.tight_layout()
plt.show()
```



```
fig, ax = plt.subplots()

videos = ["qHD", "FHD", "3K", "4K", "5K", "6K", "7K", "8K"]
x_pos = np.arange(len(videos))
CTEs = [9.21627497673034, 13.7190344333648, 20.8664650917053, 49.1172392368316, 60.1508309

ax.bar(x_pos, CTEs)
ax.set_ylabel('time (s)')
ax.set_xlabel('resolution')
ax.set_xticks(x_pos)
ax.set_xticklabels(videos)
plt.axhline(y=30,linewidth=1, color='red', label='real-time')
plt.xticks(rotation=45)
plt.legend()

# Save the figure and show
plt.tight_layout()
plt.show()
```



```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] [True, True, True, True, True, True, True, True]
[0.9866666666666667, 0.9866666666666667, 1.0, 0.9888888888888889, 0.9833333333333333, 0.98
[0.9988888888888889, 0.9988888888888889, 1.0, 0.9855555555555555, 0.9833333333333333, 0.91
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.7122222222222222, 1.0] [True, True, True, True, True, Tru
[0.9944444444444445, 0.9944444444444445, 1.0, 0.9944444444444445, 0.9911111111111112, 0.87
[1.0, 1.0, 1.0, 1.0, 1.0, 0.9866666666666667, 0.9277777777777778, 0.9277777777777778] [Tru
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] [True, True, True, True, True, True, True, True]
[0.9677777777777777, 0.9855555555555555, 1.0, 0.9922222222222222, 0.9155555555555556, 0.89
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] [True, True, True, True, True, True, True, True]
[1.0, 1.0, 1.0, 0.9988888888888889, 0.9733333333333334, 0.98, 0.9633333333333334, 0.863333
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] [True, True, True, True, True, True, True, True]
[0.8922222222222222, 1.0, 1.0, 1.0, 0.9633333333333334, 0.8266666666666667, 0.711111111111
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] [True, True, True, True, True, True, True, True]
[0.9988888888888889, 1.0, 1.0, 0.9977777777777778, 0.9855555555555555, 0.9277777777777778,
```

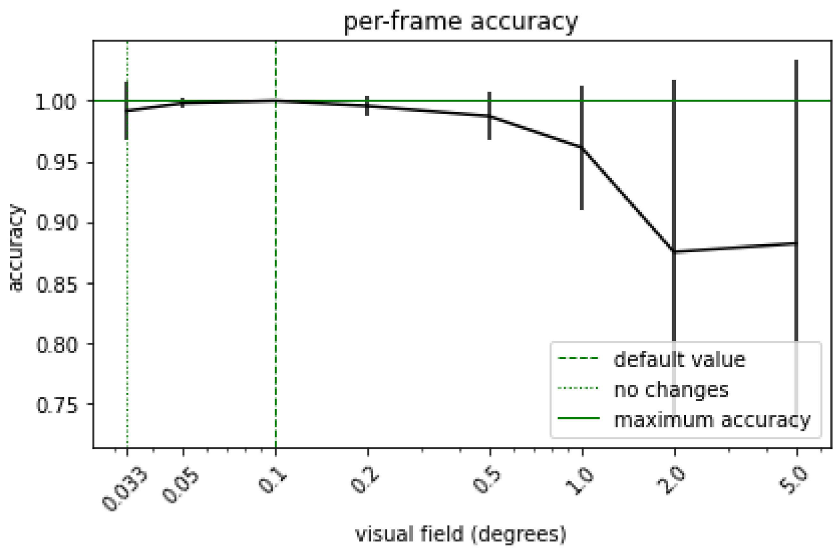
```
per_frame_accs = [[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0], [0.9866666666666667, 0.9866666
per_video_accs = [[True, True, True, True, True, True, True, True], [True, True, True, Tru
timings = [[146.7848584651947, 71.33189940452576, 11.34118366241455, 4.131996154785156, 2.
```

```
from statistics import mean, stdev
print(*map(mean, zip(*per_frame_accs)))
print(*map(stdev, zip(*per_frame_accs)))
print(*map(stdev, zip(*timings)))
```

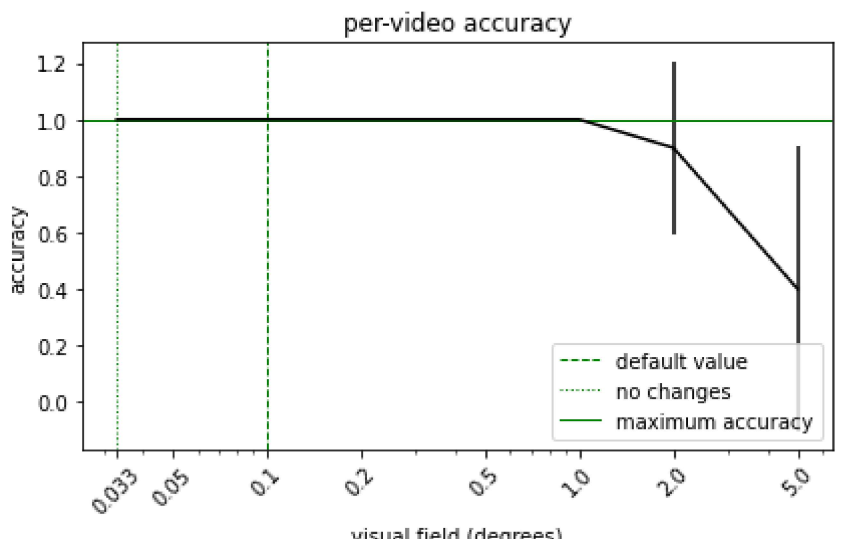
```
0.9916111111111111 0.9979444444444444 1.0 0.9957222222222222 0.9872222222222222 0.961
0.024591463782543753 0.004389647576737024 5 0.0 0.008456401199538928 0.02008914505865
113.20863495772342 36.231815520644616 7.973079156109293 3.1223397622315296 1.9566226
```

```
fig, ax = plt.subplots()

videos = [0.033, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
x_pos = np.arange(len(videos))
CTEs = [*map(mean, zip(*per_frame_accs))]
error = [*map(stdev, zip(*per_frame_accs))]
ax.errorbar(videos, CTEs, yerr=error, color='black')
ax.set_xscale('log')

ax.set_ylabel('accuracy')
ax.set_xlabel('visual field (degrees)')
```

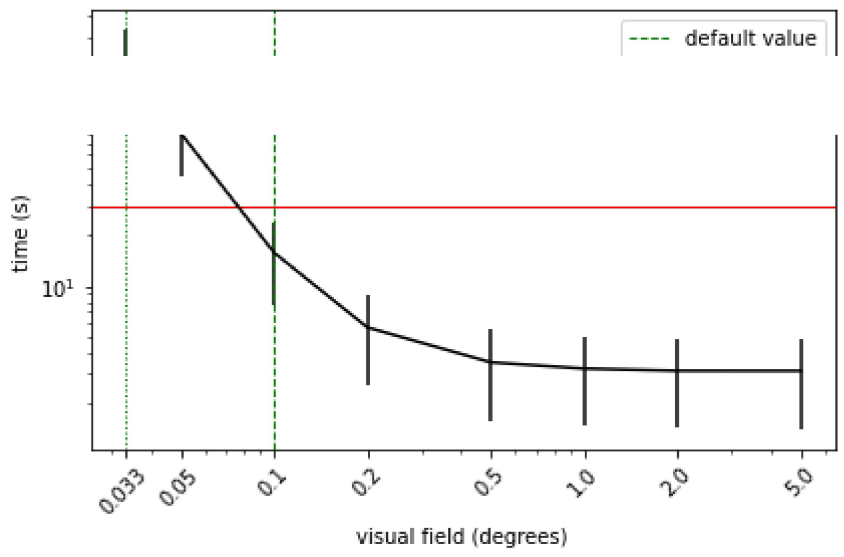
```
    ax.set_xticks(videos)
    ax.set_xticklabels(videos)
    plt.xticks(rotation=45)
    plt.axvline(x=0.1,linewidth=1, color='green', linestyle='--', label='default value')
    plt.axvline(x=0.033,linewidth=1, color='green', linestyle=':', label='no changes')
    plt.axhline(y=1,linewidth=1, color='green', label='maximum accuracy')
    plt.legend()
    plt.title('per-frame accuracy')

    plt.tight_layout()
    plt.show()
```



```
    fig, ax = plt.subplots()

    videos = [0.033, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
    x_pos = np.arange(len(videos))

    CTEs = [*map(mean, zip(*per_video_accs))]
    error = [*map(stdev, zip(*per_video_accs))]
    ax.errorbar(videos, CTEs, yerr=error, color='black')
    ax.set_xscale('log')

    ax.set_ylabel('accuracy')
    ax.set_xlabel('visual field (degrees)')
    ax.set_xticks(videos)
    ax.set_xticklabels(videos)
    plt.xticks(rotation=45)
    plt.axvline(x=0.1,linewidth=1, color='green', linestyle='--', label='default value')
    plt.axvline(x=0.033,linewidth=1, color='green', linestyle=':', label='no changes')
    plt.axhline(y=1,linewidth=1, color='green', label='maximum accuracy')
    plt.legend()
    plt.title('per-video accuracy')

    # Save the figure and show
    plt.tight_layout()
    plt.show()
```



```
fig, ax = plt.subplots()

videos = [0.033, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
x_pos = np.arange(len(videos))

CTEs = [*map(mean, zip(*timings))]
error = [*map(stdev, zip(*timings))]
ax.errorbar(videos, CTEs, yerr=error, color='black')
ax.set_xscale('log')
ax.set_yscale('log')

ax.set_ylabel('time (s)')
ax.set_xlabel('visual field (degrees)')
ax.set_xticks(videos)
ax.set_xticklabels(videos)
plt.xticks(rotation=45)
plt.axvline(x=0.1,linewidth=1, color='green', linestyle='--', label='default value')
plt.axvline(x=0.033,linewidth=1, color='green', linestyle=':', label='no changes')
plt.axhline(y=30,linewidth=1, color='red', label='real-time')
plt.axhline(y=0,linewidth=1, color='black')
plt.legend()

# Save the figure and show
plt.tight_layout()
plt.show()
```



✓ 0s completed at 15:17