計算機視覺作業

干皓丞，2101212850, 信息工程學院

2021 年 11 月 29 日

1　作業目標與作業狀況

作業可以從 GitHub 下的 kancheng/kan-cs-report-in-2021 專案找到，作業程式碼與文件目錄為 kan-cs-report-in-2021/CV/cv-final。實際執行的環境與實驗設備為 Google 的 Colab 、MacBook Pro (Retina, 15-inch, Mid 2014)、Acer Aspire R7 與 HP Victus (Nvidia GeForce RTX 3060)。

對之前的論文清單做延伸，內容包括但不限於：論文內容、程式碼復現、實驗創新等，形成一份報告。

再對過往 8 篇論文逐一復現的狀況，而第 8 篇的"Binary TTC: A Temporal Geofence for Autonomous Navigation" 復現相對穩定，但是由於原本研究者的環境是 Linux ，將原本執行的 Linux Shell 的 *.sh 檔案改寫成 PowerShell 的 *.ps1 的寫法，讓成果可以穩定執行。

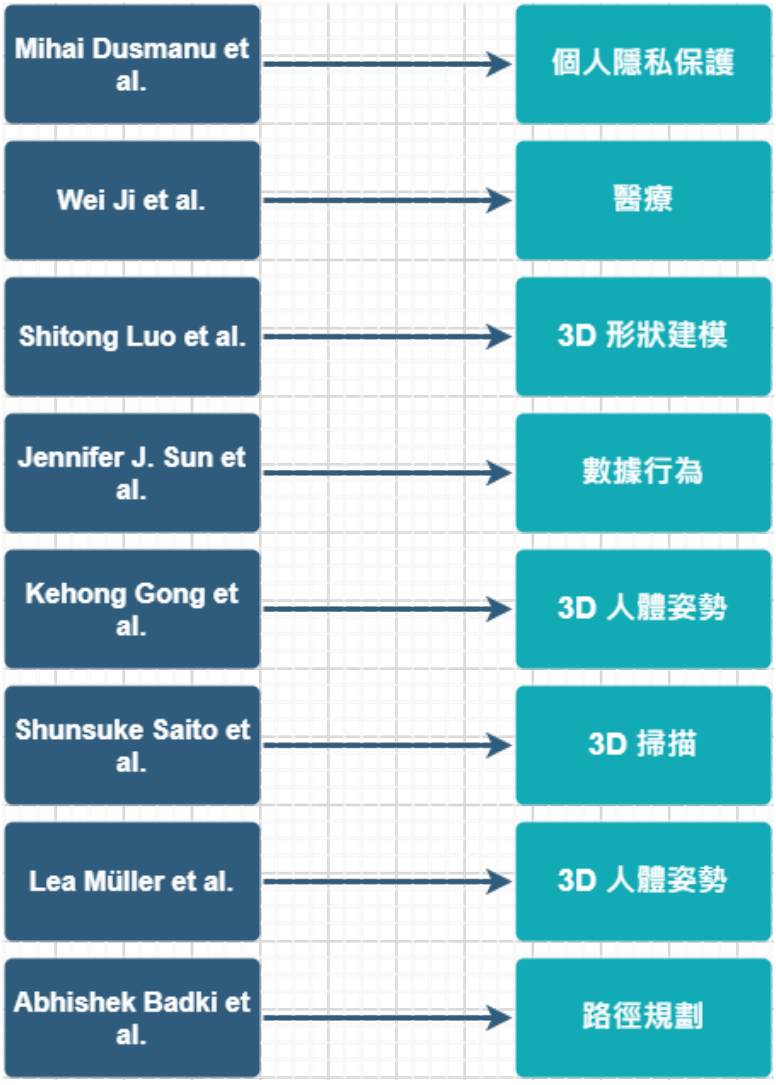


Fig. 1. 論文清單

2 Binary TTC

接觸時間 (TTC)，即物體與觀察者平面碰撞的時間，是路徑規劃的強大工具：它可能比場景中物體的深度、速度和加速度提供更多信息——即使對於人類也是如此。而 TTC 具有多種優勢，包括只需要一個單目、未校準的相機。然而，要做到回歸每個像素的 TTC 並不簡單，大多數現有方法對場景做出了過度簡化的假設。研究者們通過一系列更簡單的二元分類估計 TTC 來解決這一挑戰，他們以低延遲預測觀察者是否會在特定時間內與障礙物發生碰撞，這通常比知道精確的每像素 TTC 更重要。

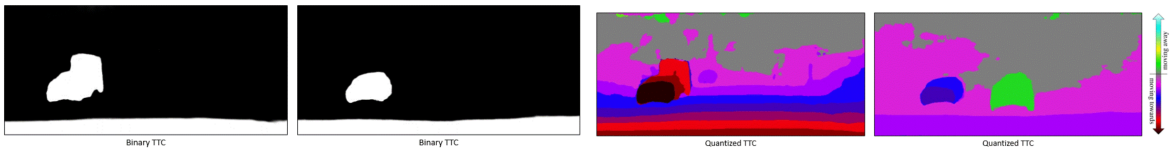
對於這種情況，該研究的方法在 6.4 毫秒內提供了時間地理圍欄 (temporal geofence)，此方法比現有方法快了 25 倍。當計算預算允許時，該研究的方法還可以通過任意精細的量化，包含連續值來估計每一個像素的 TTC。此方法是第一個以足夠高的幀速率提供 TTC 資訊 (binary or coarsely quantized) 以供實際使用的方法。

另外專案測試資料與包含論文中展示的測試結果整理如下。

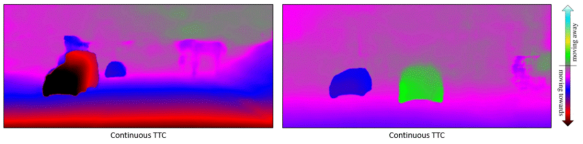
(i) 測試 - img-ref.png (ii) 測試 - img-src.png

Fig. 2. 測試論文資料

(i) 測試 - Reference image (ii) 測試 - Source image



(iii) 測試 - Binary TTC (iv) 測試 - Quantized TTC



(v) 測試 - Continuous TTC

Fig. 3. 測試論文資料

3 Linux Shell 改寫成 PowerShell

run_continuous_demo.sh 改寫成 run_continuous_demo.ps1。

```
set CUDA_VISIBLE_DEVICES=1 ; python run_continuous_estimation.py `
    --attributes ttc of `
    --pretrained '../weights/cont_ttc_kitti15_trainval.pth.tar' `
    --alpha_min 0.5 `
    --alpha_max 1.3 `
    --alpha_size 18 `
    --shift_min -12 `
    --shift_max 12 `
    --shift_delta 1 `
    --bittcnet_crop_height 192 `
    --bittcnet_crop_width 576 `
    --ref_img_path '../data/img_ref.png' `
    --src_img_path '../data/img_src.png'
```

run_binary_demo.sh 改寫成 run_binary_demo.ps1。

```
set CUDA_VISIBLE_DEVICES=1 ; python run_binary_estimation.py `
    --attribute ttc `
    --pretrained '../weights/binary_ttc_kitti15_train.pth.tar' `
    --fps 10 `
    --alpha_vals 0.7 0.75 0.8 0.85 0.9 0.95 0.98 1.02 1.10 `
    --ref_img_path '../data/img_ref.png' `
    --src_img_path '../data/img_src.png'

set CUDA_VISIBLE_DEVICES=1 ; python run_binary_estimation.py `
    --attribute of `
    --pretrained '../weights/binary_ttc_kitti15_train.pth.tar' `
    --shifts_x -84 -64 -48 -36 -24 -12 0 12 24 36 48 64 84 `
    --shifts_y -36 -24 -12 0 6 12 18 24 30 48 60 72 84 `
    --ref_img_path '../data/img_ref.png' `
    --src_img_path '../data/img_src.png'
```

4 測試結果與 Demo

(i) 測試 - 000-thr0.70-img-in-0.jpg (ii) 測試 - 000-thr0.70-img-in-1.jpg (iii) 測試 - 000-thr0.70-seg-out.jpg

Fig. 4. 000-thr0.70

(i) 測試 - 001-thr0.75-img-in-0.jpg (ii) 測試 - 001-thr0.75-img-in-1.jpg (iii) 測試 - 001-thr0.75-seg-out.jpg

Fig. 5. 001-thr0.75

(i) 測試 - 002-thr0.80-img-in-0.jpg (ii) 測試 - 002-thr0.80-img-in-1.jpg (iii) 測試 - 002-thr0.80-seg-out.jpg

Fig. 6. 002-thr0.80

(i) 測試 - 003-thr0.85-img-in-0.jpg (ii) 測試 - 003-thr0.85-img-in-1.jpg (iii) 測試 - 003-thr0.85-seg-out.jpg

Fig. 7. 003-thr0.85

(i) 測試 - 004-thr0.90-img-in-0.jpg (ii) 測試 - 004-thr0.90-img-in-1.jpg (iii) 測試 - 004-thr0.90-seg-out.jpg

Fig. 8. 004-thr0.90

(i) 測試 - 005-thr0.95-img-in-0.jpg (ii) 測試 - 005-thr0.95-img-in-1.jpg (iii) 測試 - 005-thr0.95-seg-out.jpg

Fig. 9. 005-thr0.95

(i) 測試 - 006-thr0.98-img-in-0.jpg (ii) 測試 - 006-thr0.98-img-in-1.jpg (iii) 測試 - 006-thr0.98-seg-out.jpg

Fig. 10. 006-thr0.98

(i) 測試 - 007-thr1.02-img-in-0.jpg (ii) 測試 - 007-thr1.02-img-in-1.jpg (iii) 測試 - 007-thr1.02-seg-out.jpg

Fig. 11. 007-thr1.02

(i) 測試 - 008-thr1.10-img-in-0.jpg (ii) 測試 - 008-thr1.10-img-in-1.jpg (iii) 測試 - 008-thr1.10-seg-out.jpg

Fig. 12. 008-thr1.10

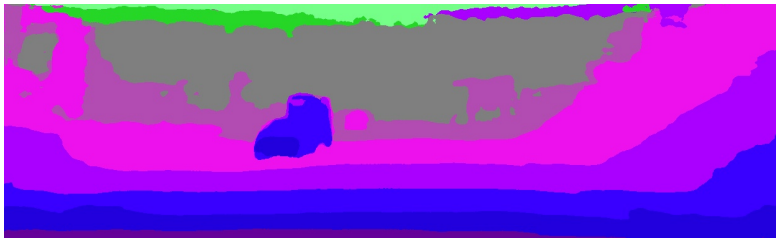
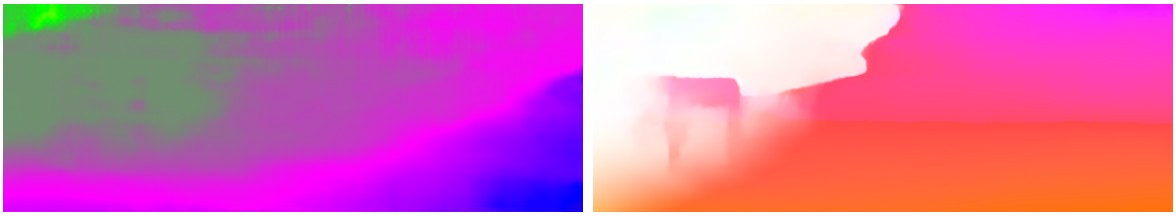


Fig. 13. quant-ttc-out.jpg



(i) 測試 - eta-out.png (ii) 測試 - of-out.png

Fig. 14. cont_ttc-of

5 附件

執行 run_continuous_demo.ps1 的結果。

```
(bittc) PS D:\py-test\BiTTC-master\test\src> .\run_continuous_demo.ps1
==> ALL PARAMETERS
ref_img_path : ../data/img_ref.png
src_img_path : ../data/img_src.png
attributes : ['ttc', 'of']
pretrained : ../weights/cont_ttc_kitti15_trainval.pth.tar
arch : bittcnet_continuous_of_ttc_2d
bittcnet_featnet_arch : featextractnetspp
bittcnet_segnet_arch : segnet2d
segnet_num_imgs : 2
segnet_num_segs : 3
segnet_is_deep : True
regrefinenet_out_planes : 32
bittcnet_max_scale : 1.5
alpha_min : 0.5
alpha_max : 1.3
alpha_size : 18
shift_min : -12.0
shift_max : 12.0
shift_delta : 1
bittcnet_crop_height : 192
bittcnet_crop_width : 576
dtype : torch.float32
device : cuda

Results will be dumped in ./results/21_11_30-00_31_21_cont_ttc-of

Padding will be added to ensure input image size divisible by 192
New input image size is 384 x 1344
Crop size will be 192 x 576
=> using pre-trained model 'bittcnet_continuous_of_ttc_2d'
==> USING BiTTCNetContinuousOFTTC2D
bittcnet_featnet_arch : featextractnetspp
bittcnet_segnet_arch : segnet2d
bittcnet_max_scale : 1.5
bittcnet_crop_height : 192
bittcnet_crop_width : 576
bittcnet_full_height : 384
bittcnet_full_width : 1344
==> USING FeatExtractNetSPP
bittcnet_featnet_arch : featextractnetspp
==> USING SegNet2D
==> USING RegRefineNet
regrefinenet_out_planes : 32
```

```
C:\Users\zxdfg\.conda\envs\bittc\lib\site-packages\torch\nn\functional.py:4003: UserWarning: Default gr
  warnings.warn(

Results generated

Visualizing results

DONE!
```

執行 run_binary_demo.ps1 **的結果。**

```
(bittc) PS D:\py-test\BiTTC-master\test\src> .\run_binary_demo.ps1
==> ALL PARAMETERS
ref_img_path : ../data/img_ref.png
src_img_path : ../data/img_src.png
attribute : ttc
pretrained : ../weights/binary_ttc_kitti15_train.pth.tar
arch : bittcnet_binary_of_ttc_2d
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
bittcnet_segnet_arch : segnet2d
bittcnet_refinenet_arch : segrefinenet
segnet_num_imgs : 2
segnet_num_segs : 3
bittcnet_num_refinenets : 3
featextractnethr_out_planes : 16
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
bittcnet_max_scale : 1.5
alpha_vals : [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.98, 1.02, 1.1]
shifts_x : [-72, -60, -48, -36, -24, -12, 0, 12, 24, 36, 48, 60, 72]
shifts_y : [-18, -15, -12, -9, -6, -3, 0, 3, 6, 9, 12, 15, 18]
fps : 10.0
dtype : torch.float32
device : cuda

Results will be dumped in ./results/21_11_30-00_31_32_binary_ttc

Padding will be added to ensure input image size divisible by 192
New input image size is 384 x 1344


Computing binary TTC

alpha/motion-in-depth threshold values:
[0.7  0.75 0.8  0.85 0.9  0.95 0.98 1.02 1.1 ]
```

```
corresponding time-to-contact threshould values:
[ 0.33333334  0.4         0.50000006  0.6666668   0.9999998   1.9999996
  5.0000052  -5.0000052  -0.9999998 ]

=> using pre-trained model 'bittcnet_binary_of_ttc_2d'
==> USING BiTTCNetBinaryOFTTC2D
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
bittcnet_segnet_arch : segnet2d
bittcnet_refinenet_arch : segrefinenet
bittcnet_num_refinenets : 3
bittcnet_max_scale : 1.5
bittcnet_crop_height : 384
bittcnet_crop_width : 1344
==> USING FeatExtractNetSPP
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
featextractnethr_out_planes : 16
==> USING SegNet2D
==> USING FeatExtractNetHR
featextractnethr_out_planes : 16
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
C:\Users\zxdfg\.conda\envs\bittc\lib\site-packages\torch\nn\functional.py:4003: UserWarning: Default gr
  warnings.warn(

Results generated

Visualizing results

alpha threshold values are in increasing order
Computing quantized TTC map

DONE!
==> ALL PARAMETERS
ref_img_path : ../data/img_ref.png
src_img_path : ../data/img_src.png
```

```
attribute : of
pretrained : ../weights/binary_ttc_kitti15_train.pth.tar
arch : bittcnet_binary_of_ttc_2d
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
bittcnet_segnet_arch : segnet2d
bittcnet_refinenet_arch : segrefinenet
segnet_num_imgs : 2
segnet_num_segs : 3
bittcnet_num_refinenets : 3
featextractnethr_out_planes : 16
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
bittcnet_max_scale : 1.5
alpha_vals : [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.98, 1.02, 1.1]
shifts_x : [-84.0, -64.0, -48.0, -36.0, -24.0, -12.0, 0.0, 12.0, 24.0, 36.0, 48.0, 64.0, 84.0]
shifts_y : [-36.0, -24.0, -12.0, 0.0, 6.0, 12.0, 18.0, 24.0, 30.0, 48.0, 60.0, 72.0, 84.0]
fps : 10
dtype : torch.float32
device : cuda

Results will be dumped in ./results/21_11_30-00_31_40_binary_of

Padding will be added to ensure input image size divisible by 192
New input image size is 384 x 1344


Computing 2D binary OF

2D Binary optical flow segmentation will be computed with respect to following thresholds:
(-84.00, -36.00)
(-64.00, -24.00)
(-48.00, -12.00)
(-36.00, 0.00)
(-24.00, 6.00)
(-12.00, 12.00)
(0.00, 18.00)
(12.00, 24.00)
(24.00, 30.00)
(36.00, 48.00)
(48.00, 60.00)
(64.00, 72.00)
(84.00, 84.00)

=> using pre-trained model 'bittcnet_binary_of_ttc_2d'
==> USING BiTTCNetBinaryOFTTC2D
```

```
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
bittcnet_segnet_arch : segnet2d
bittcnet_refinenet_arch : segrefinenet
bittcnet_num_refinenets : 3
bittcnet_max_scale : 1.5
bittcnet_crop_height : 384
bittcnet_crop_width : 1344
==> USING FeatExtractNetSPP
bittcnet_featnet_arch : featextractnetspp
bittcnet_featnethr_arch : featextractnethr
featextractnethr_out_planes : 16
==> USING SegNet2D
==> USING FeatExtractNetHR
featextractnethr_out_planes : 16
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
==> USING SegRefineNet
segrefinenet_in_planes : 17
segrefinenet_out_planes : 8
segrefinenet_num_layers : 4
C:\Users\zxdfg\.conda\envs\bittc\lib\site-packages\torch\nn\functional.py:4003: UserWarning: Default gr
  warnings.warn(
Traceback (most recent call last):
  File "D:\py-test\BiTTC-master\test\src\run_binary_estimation.py", line 326, in <module>
    main()
  File "D:\py-test\BiTTC-master\test\src\run_binary_estimation.py", line 224, in main
    out = model(img_list, T_inv_list, T_list, is_compute_TTC)[1]
  File "C:\Users\zxdfg\.conda\envs\bittc\lib\site-packages\torch\nn\modules\module.py", line 1102, in _
    return forward_call(*input, **kwargs)
  File "D:\py-test\BiTTC-master\test\src\models\BiTTCNet.py", line 103, in forward
    seg_raw_low_res = self.segnet2D(dzv)
  File "C:\Users\zxdfg\.conda\envs\bittc\lib\site-packages\torch\nn\modules\module.py", line 1102, in _
    return forward_call(*input, **kwargs)
  File "D:\py-test\BiTTC-master\test\src\models\SegNet2D.py", line 118, in forward
    out_deconv1_1 = self.deconv1_1(torch.cat((out_conv0, out_deconv1), 1))
RuntimeError: CUDA out of memory. Tried to allocate 820.00 MiB (GPU 0; 6.00 GiB total capacity; 3.37 Gi
(bittc) PS D:\py-test\BiTTC-master\test\src>
```