# 2025 NCHC Open Hackthon

## Optimizing Motion Capture and Athlete Reconstruction Systems through Multi-View 3D Reconstruction

Team 9 Paw Patrol

OpenACC
More Science, Less Programming

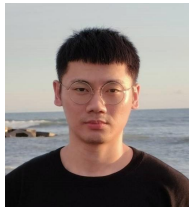
# Paw Patrol

- **Team Members from NTHU IGVI Lab**

Tsung-Hsun Tsai

Calvin Ku

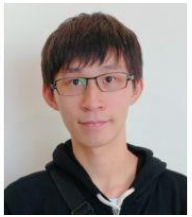
Pu Ching



Shang-Ching Liu

Prof. Hung-Kuo Chu

- **Mentor from Nvidia**



Johnson Sun

Zih-Syuan Jhan

Chen-Hao Huang

Tito Aribowo

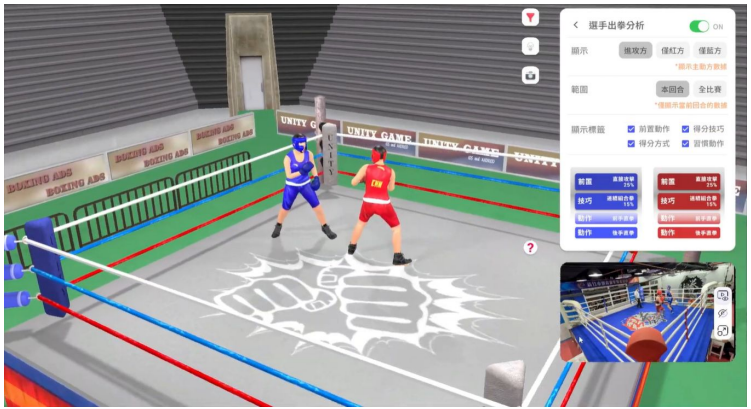
Prof. Min-Chun Hu

# Optimizing Motion Capture and Athlete Reconstruction Systems through Multi-View 3D Reconstruction

## Introduction and Motivation:

**Human motion capture** and 3D reconstruction are key technologies for understanding and analyzing human movements in sports. Reconstructing accurate **human meshes** from multi-view images provides detailed 3D geometry beyond traditional 2D/3D pose estimation, enabling biomechanical analysis and realistic digital human generation.

However, current reconstruction pipelines are **computationally expensive** and **too slow** for real-time or large-scale applications.




# Original Reconstruction Pipeline

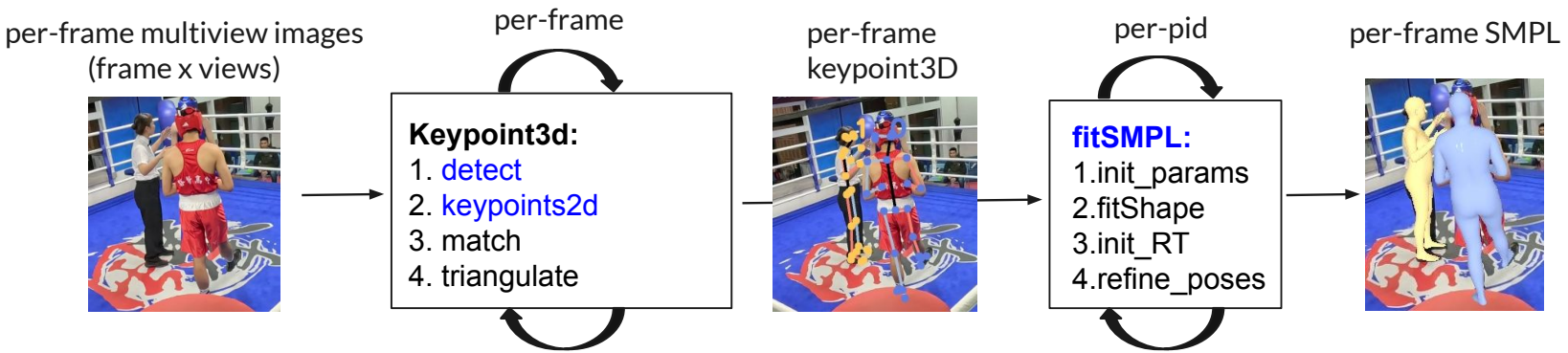
The pipeline (based on existing mocap framework——Easymocap) performs multi-view 3D human pose reconstruction using a combination of deep learning for human detection and 2D keypoint estimation, followed by optimization-based human mesh (SMPL) fitting for each detected person.

```
[Timing] Total time per step stage:
detect                : 1233.250s
keypoints2d           : 2629.981s
vis_2d                : 0.137s
match                 : 96.607s
```

Keypoint3d Porfiling

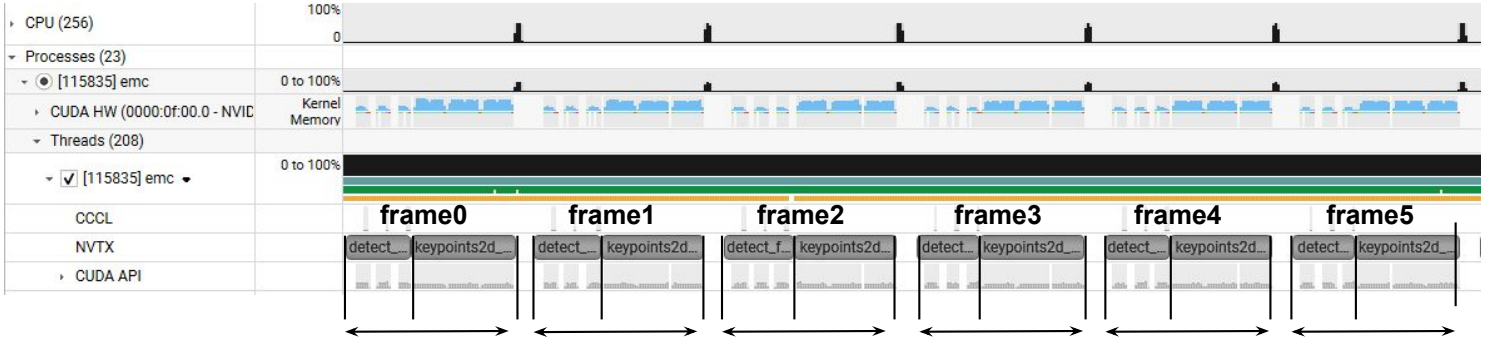
```
[Timing] Total time per final stage:
write_raw             : 7.738s
collect               : 0.096s
load_body_model       : 0.000s
fitting_each_person   : 4338.863s
write                 : 111.342s
```

fitSMPL Porfiling

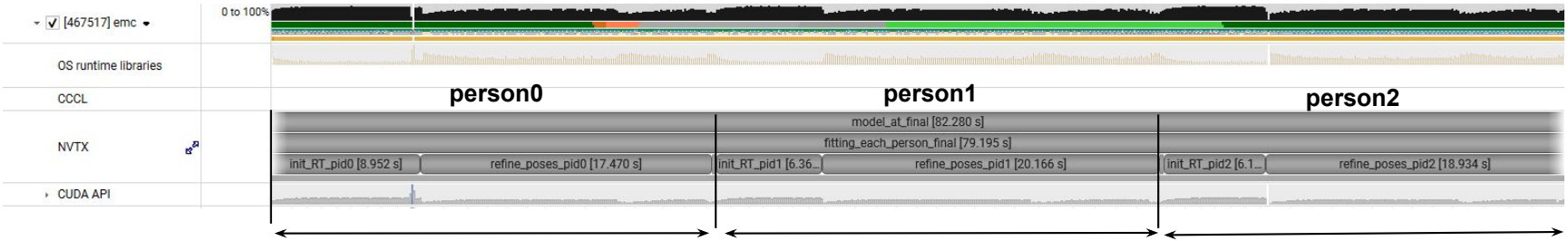



**Sequential estimation!**

# Problem to Solve




**per-frame sequencial process !**



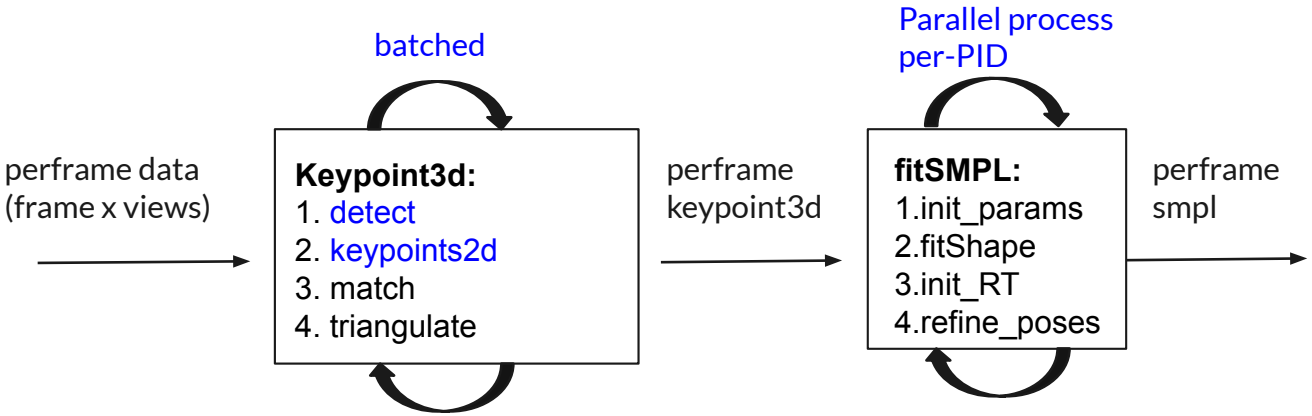

**per-subject sequencial process !**

# Goal and Strategy

- What was your goal for coming here?

  Accelerate the multi-view 3D human-pose reconstruction pipeline without affecting final reconstruction quality.

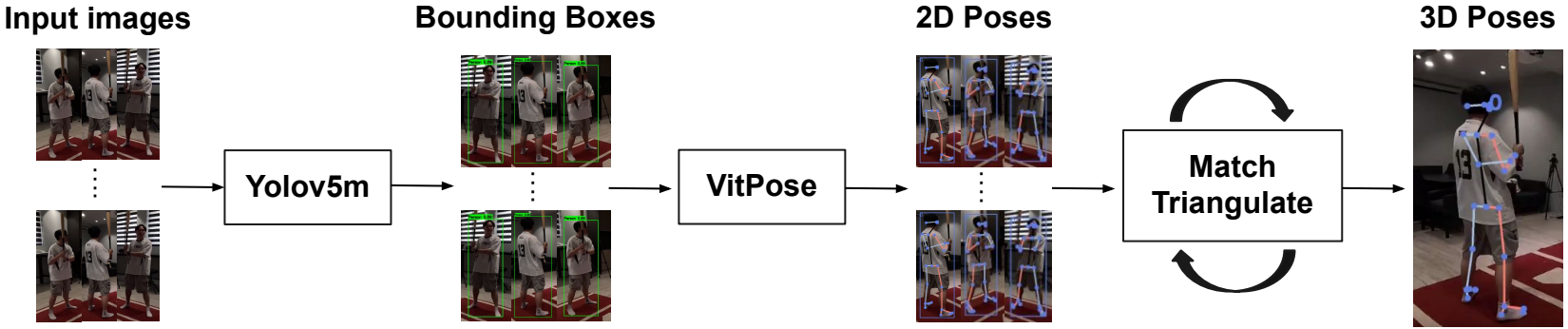
- What was your initial strategy?
  1. Batched process for detect and keypoint2d
  2. Parallel processing the fitsmpl stage
  3. Optimization of the SMPL fitting stage

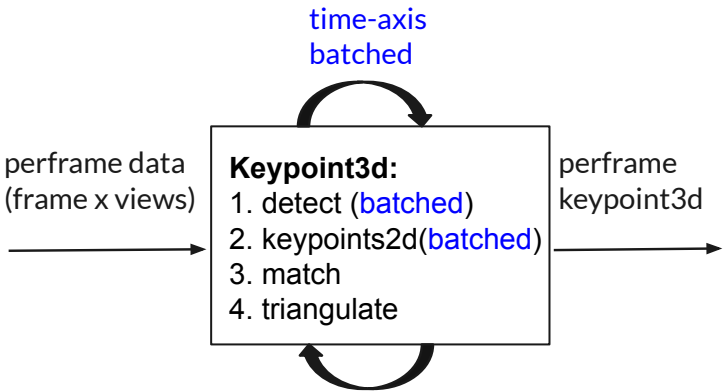


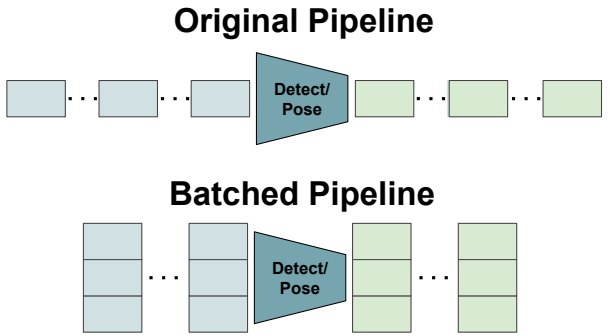
# Evaluation

- What were you accomplished?

1. Batched process for Detect and Keypoint2d



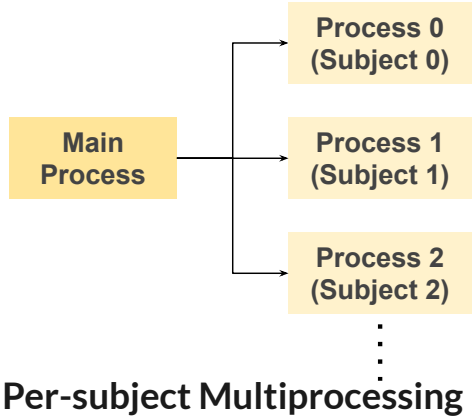
# Evaluation

- What were you accomplished?

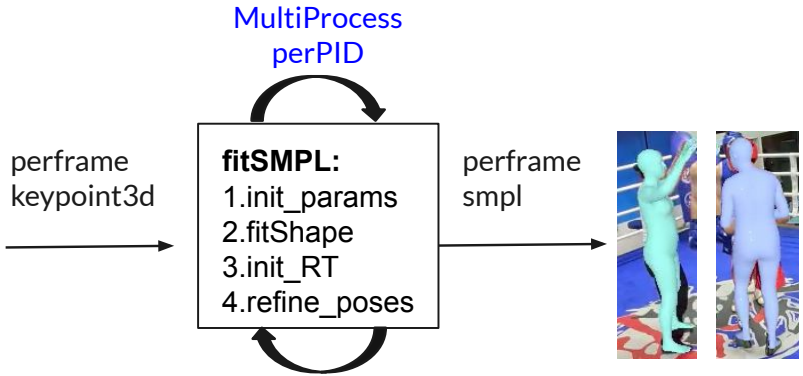
    2. Multiprocessing in SMPL fitting

```python
# Each worker preloads SMPL fitting stages
def init_worker(stage_args):
    STAGES = {k: load_model(v) for k, v in stage_args.items()}

# Each worker optimizes one person through all stages
def worker(pid, data):
    for stage in STAGES.values():
        data = stage.optimize(data)
    return pid, data

# Main process: persistent pool for parallel execution
pool = Pool(initializer=init_worker, initargs=(stages,))
results = pool.map(worker, persons)
```

Pseudo Code




Per-subject Multiprocessing




Fitting Pipeline

# Evaluation

**fitSMPL:**
1.init_params
**2.fitShape**
**3.init_RT**
**4.refine_poses**

- What were you accomplished?

3. Reduced LBFGS history size (100->10)

$$f(x) = \text{Loss}(\text{SMPL}(x), \text{observations})$$

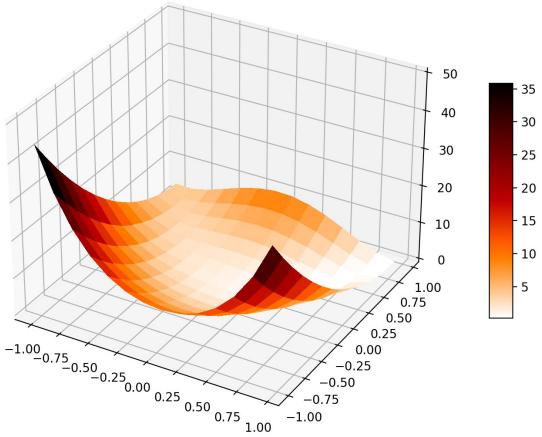
**Compute gradient**

**Compute search direction using approximate inverse Hessian**

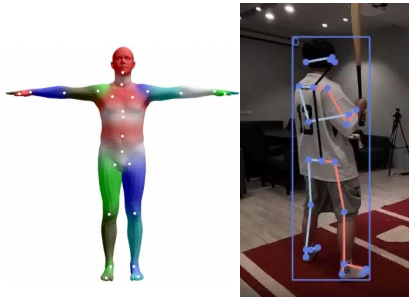
**Time consuming !**

**Line search along direction**

**Update history**



```
q = flat_grad.neg()
for i in range(num_old - 1, -1, -1):
    al[i] = old_stps[i].dot(q) * ro[i]
    # q.add_(-al[i], old_dirs[i])
    q.add_(old_dirs[i], alpha=-al[i])

# multiply by initial Hessian
# r/d is the final direction
d = r = torch.mul(q, H_diag)
for i in range(num_old):
    be_i = old_dirs[i].dot(r) * ro[i]
    # r.add_(al[i] - be_i, old_stps[i])
    r.add_(old_stps[i], alpha=(al[i] - be_i))
```



# Experiments

We considered two cases

1. **Single-person (simplest case)**

- ❏ Evaluate improvements in each stage: **Detection、Keypoints2D、SMPL**
- ❏ Serves as the **baseline** for per-stage efficiency.

2. **Multi-person (general case)**

- ❏ Each stage's workload scales with the number of people, making the original pipeline much slower.
- ❏ Tested **per-stage** and **overall** performance gains after optimization.

baseball0 3views

boxing 3views

# Results and Final Profile

**Strategies**

1. **batched detect and keypoint2d**
2. **Parallel process the fitsmpl stage**
3. **reduce lbfgs history size**

## Case1: baseball0 795 frames 3views - Single-person

**Overall x8.5, Detect x12.5, Keypoint2d x20.8, fitsmpl x4.6** in one person case (campare with Baseline on RTX 2060)

**Overall x2.2, Detect x3.7, Keypoint2d x5.3, fitsmpl x1.5** in one person case (campare with Baseline on A100)

| Settings (test on single-person data) | CPU device | device | Detect | Keypoint2d | Fitsmpl | total time |
|---|---|---|---|---|---|---|
| **Baseline** | Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz | RTX 2060 | 143.2 s | 239.3 s | 94.0 s | 655.0 s |
| **Baseline** | AMD EPYC 7742 64-Core Processor | A100 | 42.9 s (3.3x) | 61.0 s (3.9x) | 30.1 s (3.1x) | 167.3 s (3.9x) |
| **Stratagy1 (batch size = 3*32)** | AMD EPYC 7742 64-Core Processor | A100 | **11.5 s (12.5x)** | **11.5 s (20.8x)** | 30.1 s (3.1x) | 86.1 s (7.6x) |
| **Stratagy1 (batch size = 3*32) + Stratagy3** | AMD EPYC 7742 64-Core Processor | A100 | **11.5 s (12.5x)** | **11.5 s (20.8x)** | **20.5 s (4.6x)** | **77.2 s (8.5x)** |

# Results and Final Profile

**Strategies**

1. **batched detect and keypoint2d**
2. **Parallel process the fitsmpl stage**
3. **reduce lbfgs history size**

## Case2: boxing 600 frames 3views - Multi-person

**Overall x6.4, Detect x13.9, Keypoint2d x8.5, fitsmpl x6.2** in multi-person case (campare with Baseline on RTX 2060)

**Overall x2.5, Detect x7.6, Keypoint2d x2.8, fitsmpl x2.7** in multi-person case (campare with Baseline on A100)

| Settings<br>(test on multi-person data 600 frames) | CPU device | device | Detect | Keypoint2d | Fitsmpl | Total time |
|---|---|---|---|---|---|---|
| **Baseline** | Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz | RTX 2060 | 130.2 s | 316.4 s | 738.5 s | 1364.6 s |
| **Baseline** | AMD EPYC 7742 64-Core Processor | A100 | 71.1 s (1.8x) | 103.4 s (3.1x) | 321.2 s (2.3x) | 529.9 s (2.6x) |
| **Stratagy1 (batch size = 3*32)<br>+ Stratagy3** | AMD EPYC 7742 64-Core Processor | A100 | 9.4 s (13.9x) | 37.4 s (8.5x) | 222.8 s (3.3x) | 310.2 s (4.4x) |
| **Stratagy1 (batch size = 3*32)<br>+ Stratagy2 (Multi-Threading version)<br>+ Stratagy3** | AMD EPYC 7742 64-Core Processor | A100 | 9.4 s (13.9x) | 37.4 s (8.5x) | 146.4 s (5.0x) | 228.4 s (6.0x) |
| **Stratagy1 (batch size = 3*32)<br>+ Stratagy2 (Multi-Processing version)<br>+ Stratagy3** | AMD EPYC 7742 64-Core Processor | A100 | **9.4 s (13.9x)** | **37.4 s (8.5x)** | **119.6 s (6.2x)** | **211.6 s (6.4x)** |

# Final Thoughts

- Was this Open Hackathon worth it?

  Yes. **Special thanks to mentor, Johnson Sun**

- What did you learn?

  Profiling Cycle

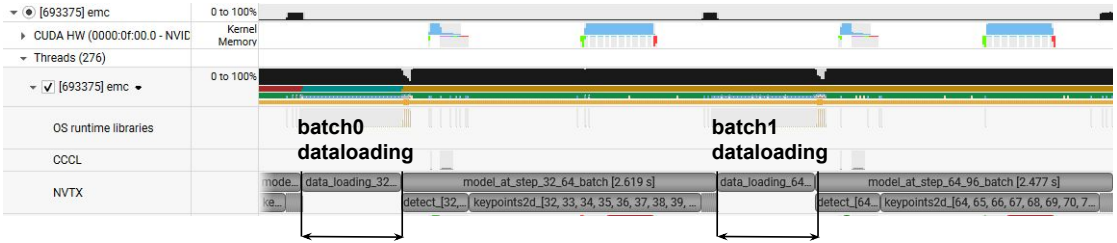
- Future Works

  Optimization of data loading

  TensorRT (detect + keypoints)

  Optimization of LBFGS approximate inverse Hessian process

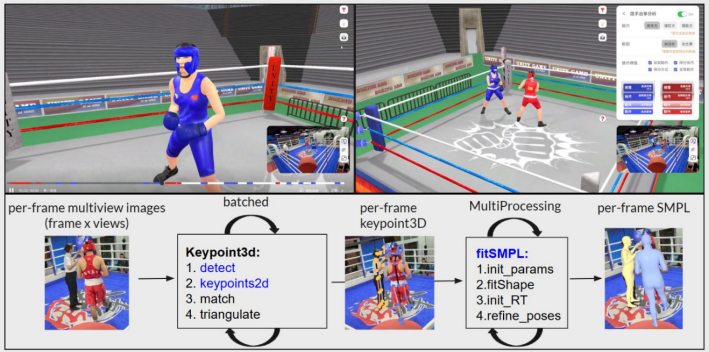
  Use neural network–based initialization to provide better SMPL parameters and reduce optimization iterations.



# Create a storyline for publication on NCHC's website.

## 3D人體姿勢重建

Team 9 : Prof. Hung-Kuo Chu, Prof. Min-Chun Hu, Tsung-Hun Tsai, Calvin Ku, Pu Ching, Shang-Ching Liu, Zih-Syuan Jhan, Chen-Hao Huang, Tito Aribowo




**Paw-Patrol 團隊成員來自清華大學資工系「朱宏國和胡敏君老師」帶領的IGVI實驗室，將3D人體姿勢重建加速了 8.5 倍！ — NVIDIA Mentor: Johnson Sun**

隨著運動科技與人工智慧的快速發展，人體動作捕捉與三維重建已成為理解與分析運動表現的關鍵技術。透過多視角影像重建出精準的人體模型，不僅能獲得比傳統姿勢估計更完整的幾何資訊，還能應用於運動表現分析與擬真人體生成等領域。

然而，現有的多視角重建流程往往計算量龐大、處理速度緩慢，難以滿足即時互動或大規模應用的需求。為了實現更快速的人體重建流程，我們團隊對整個系統進行了優化，並在 EasyMocap 的基礎上加以改良。將其改寫為批次 (Batch) 處理並結合多程序 (Multiprocess) 加速，有效提升運算效率。同時，我們調整了 SMPL 人體模型擬合的優化器參數，以進一步加快人體模型的生成速度。透過 GPU 加速，最終整體流程獲得約 8.5 倍的加速效果。

Table1: Speedup in single person case

| Settings | device | Detect | Keypoint2d | Fitsmpl | total time |
|---|---|---|---|---|---|
| **Baseline** | RTX 2060 | 143.2 s | 239.3 s | 94.0 s | 655.0 s |
| **Baseline** | A100 | 42.9 s (3.3x) | 61.0 s (3.9x) | 30.1 s (3.1x) | 167.3 s (3.9x) |
| **Optimized** | A100 | **11.5 s (12.5x)** | **11.5 s (20.8x)** | **20.5 s (4.6x)** | **77.2 s (8.5x)** |

Table2: Speedup in multi person case

| Settings | device | Detect | Keypoint2d | Fitsmpl | Total time |
|---|---|---|---|---|---|
| **Baseline** | RTX 2060 | 130.2 s | 316.4 s | 738.5 s | 1364.6 s |
| **Baseline** | A100 | 71.1 s (1.8x) | 103.4 s (3.1x) | 321.2 s (2.3x) | 529.9 s (2.6x) |
| **Optimized** | A100 | **9.4 s (13.9x)** | **37.4 s (8.5x)** | **119.6 s (6.2x)** | **211.6 s (6.4x)** |

Poster Link:
https://www.canva.com/design/DAG4Ym4iswc/5tBFc2CT-FuCZ6W1ZERdHQ/edit?utm_content=DAG4Ym4iswc&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton


Thank You
OpenACC
More Science, Less Programming