UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

RapidPoseTriangulation: multi-view multi-person whole-body human pose triangulation in a millisecond

Daniel Bermuth, Alexander Poeppel, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Bermuth, Daniel, Alexander Poeppel, and Wolfgang Reif. 2026. "RapidPoseTriangulation: multi-view multi-person whole-body human pose triangulation in a millisecond." arXiv. arXiv. https://doi.org/10.48550/arXiv.2503.21692.

Nutzungsbedingungen / Terms of use: **CC BY-SA 4.0**

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
CC-BY-SA 4.0: Creative Commons: Namensnennung - Weitergabe unter gleichen Bedingungen
Weitere Informationen finden Sie unter: / For more information see:
https://creativecommons.org/licenses/by-sa/4.0/deed.de

RapidPoseTriangulation: Multi-view Multi-person Whole-body Human Pose Triangulation in a Millisecond

Daniel Bermuth
University of Augsburg, Germany
daniel.bermuth@uni-a.de

Alexander Poeppel
University of Augsburg
poeppel@isse.de

Wolfgang Reif
University of Augsburg
reif@isse.de

Abstract

The integration of multi-view imaging and pose estimation represents a significant advance in computer vision applications, offering new possibilities for understanding human movement and interactions. This work presents a new algorithm that improves multi-view multi-person pose estimation, focusing on fast triangulation speeds and good generalization capabilities. The approach extends to whole-body pose estimation, capturing details from facial expressions to finger movements across multiple individuals and viewpoints. Adaptability to different settings is demonstrated through strong performance across unseen datasets and configurations. To support further progress in this field, all of this work is publicly accessible.

1. Introduction

The ability to accurately detect and track human body positions and movements is an important task in many advanced computer vision applications. From enhancing virtual reality experiences to improving the collaboration with humans in robotic systems, precise human poses play a crucial role at connecting digital systems with the physical world.

Human pose estimation, which involves determining the spatial locations of different body joints, has seen significant progress in recent years. While traditional approaches relied on wearable markers for precise tracking, the field has increasingly moved towards marker-less methods using standard cameras. This shift offers greater flexibility and usefulness, especially in scenarios where attaching markers is impractical or impossible, such as in public spaces or specialized environments like operating rooms.

However, marker-less pose estimation, especially with only a single viewpoint, presents its own set of challenges. Occlusions, varying lighting conditions, and the complexity of human movements can all impact the accuracy and reliability of these systems. To address external and self-occlusion issues, and to some extent also varying lighting, multi-view approaches have become more common. By capturing the scene from multiple angles simultaneously, these systems can overcome many of those limitations.

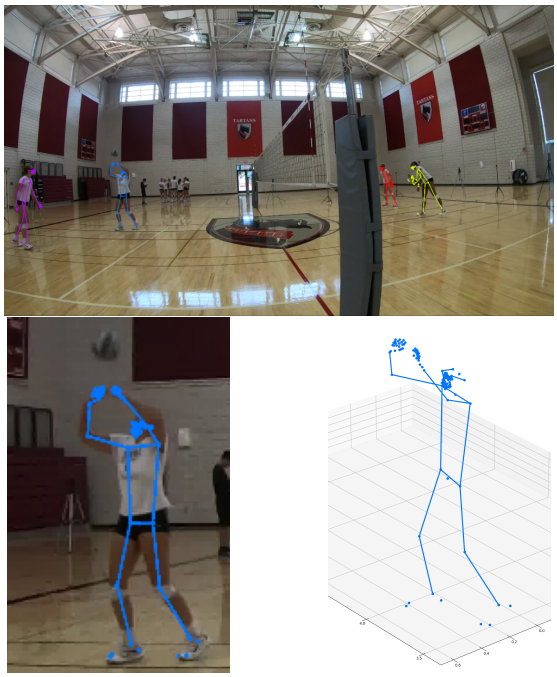


Figure 1. Example of a multi-person whole-body pose estimation from multiple camera views in a volleyball game (from the *egohumans* dataset [16]). On top, the full image of one camera with the projections of all the detected poses, on the bottom-left a zoom-in on one player, and on the bottom-right her detected 3D pose.

Recent advances in deep learning have dramatically improved the accuracy of pose estimation from individual camera views. However, the task of efficiently combining these multiple 2D estimates into accurate 3D poses, particularly for multiple people in real-time scenarios, remains an active area of research. Current approaches often show notable performance drops when applied to setups they were not trained on, and are normally too slow for real-time usage [2]. Furthermore, as applications become more advanced, there is an increasing demand for a more detailed whole-body pose estimation, including facial expressions and especially hand gestures (see Figure 1).

This work addresses these challenges by introducing a novel algorithm for multi-view multi-person human pose estimation. The approach is designed to be both fast and capable of generalizing well across different datasets and application setups. Unlike most previous works, the algorithm is also able to estimate whole-body poses. In a broad evaluation across multiple datasets, it is shown that the proposed algorithm is more reliable than existing methods, while also being significantly faster, which makes it especially suitable for real-time applications.

The demonstration that a lightweight algebraic method can outperform many modern learned approaches also raises an important question: Is the current trend toward increasingly complex and novelty-oriented learnable architectures truly the most effective way to improve performance in multi-view pose estimation?

To further support future research in all directions, the source-code of the presented algorithm is made available at: https://gitlab.com/Percipiote/

2. Related Work

Most approaches address the problem of human pose estimation in two separate steps. At first, 2D poses are predicted for each image, and in the second step, these are fused to estimate 3D poses. The basic concepts of the different approaches in the second step can be categorized by whether they use algorithmic strategies or learning-based methods. Some of them also make use of temporal information from previous frames to improve the results.

On the side of the learning-based methods, *VoxelPose* [24] was one of the first concepts, building upon the voxel-based triangulation research of *Iskakov et al.* [13] and extending it from single-person to multi-person estimations. Its method projects joint heatmaps from 2D images into a 3D voxelized space, estimates a central point for each individual using a convolution network, extracts a focused cube surrounding each person center, and computes the joint positions using a second net. *Faster-VoxelPose* [28] enhanced this technique by restructuring the 3D voxel network into several 2D and 1D projections, thereby boosting efficiency. *TesseTrack* [18] and *TEMPO* [8] added a temporal dimension into the voxel space to refine poses across sequential frames. Alternative approaches such as *PRGnet* [26] employ a graph-based methodology or directly infer 3D poses from 2D features, like in *MvP* [25]. *SelfPose3d* [21] adopts the framework of *VoxelPose* and trains both 2D and 3D networks in a self-supervised manner using randomly augmented 2D poses.

Regarding algorithmic methodologies, *mvpose* [10] solves the estimation problem in two steps. Initially, it identifies corresponding 2D poses across images based on geometric and visual clues, and then triangulates the matching pose groups to calculate the final output. *mv3dpose* [23] utilizes a graph-matching approach to assign poses through epipolar geometry and incorporates temporal information to handle missing joint data. *PartAwarePose* [9] accelerates the pose-matching process by leveraging poses from preceding frames and applying a joint-based filter to solve keypoint inaccuracies resulting from occlusions. *VoxelKeypointFusion* [2] employs a voxel-based triangulation concept to predict 3D joint proposals from overlapping viewpoints. It then uses the joint reprojections to assign them to individuals in the original 2D views, to match them to individual 3D persons before merging them into a final result.

Regarding the generalization of learning-based approaches to new datasets, a direct transfer is typically not evaluated. However, *VoxelPose*, *Faster-VoxelPose*, *MvP*, and *TEMPO* have implemented a finetuning strategy utilizing synthetic data, though only the first two have also published the source-code for this. The idea is to randomly place 3D poses from another dataset in 3D space, and back-project them onto the camera perspectives of the target setup. After adding some augmentations, the network is then trained to learn the 3D reconstruction process. In the closed-source approach of *CloseMoCap* [19] this concept was further improved by using a larger 3D pose dataset and additional augmentations. Algorithmic approaches generally evaluated their transferability to a limited number of other datasets, a process that is considerably more straightforward as it doesn't require any training.

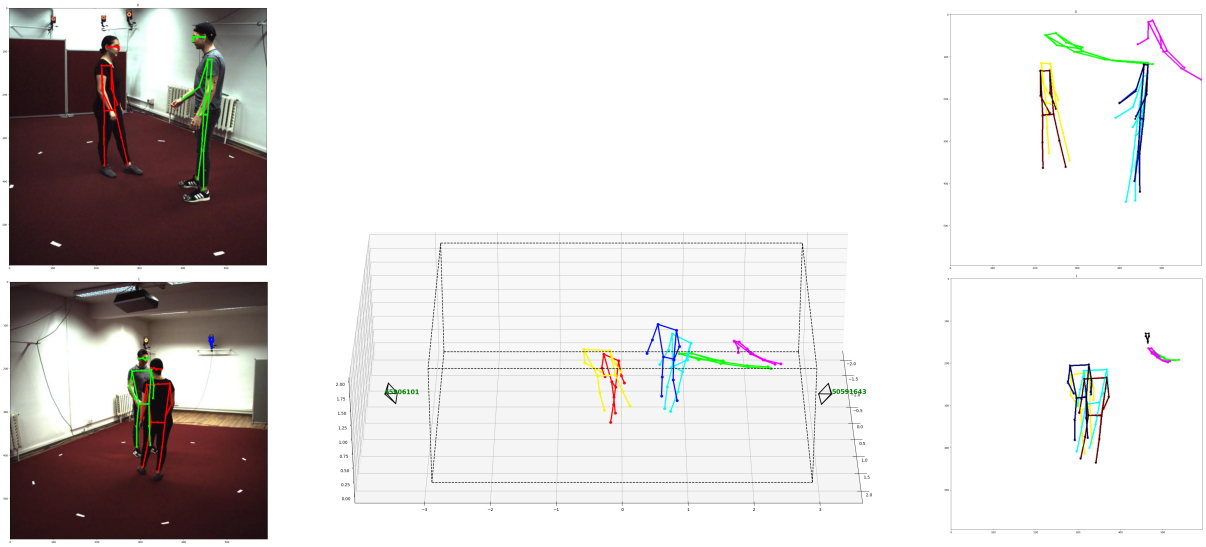
3. RapidPoseTriangulation

The new algorithm called *RapidPoseTriangulation* follows a simple and learning-free triangulation concept.

3.1. Algorithm

Similar to most other approaches, the algorithm can be split into two stages as well, with the first one predicting the 2D poses for each image. For this any 2D pose estimator can be integrated, here *RTMPose* [14] was used. The second stage can be split into the following steps:

1. Create all possible pairs with poses from other views
2. Filter pairs using the previous 3D poses
3. Select the core joints
4. Triangulate each pair to a 3D proposal
5. Drop proposals outside the room
6. Reproject into 2D views
7. Calculate reprojection error to original 2D poses
8. Drop pairs/proposals with large errors
9. Group the remaining 3D proposals in 3D space
10. Triangulate again, but now with all joints
11. Merge the proposals groups into a single 3D pose
12. Drop clearly invalid persons
13. Optional: Assign persons to tracks over time
14. Optional: Clip joint movements to a maximum speed



(a) (b) (c)

Figure 2. Obtaining 3D proposals. The process starts with 2D detections for each view (a, images from *chi3d* [11], note the small blue colored false positive in the second image). Then, in step (1), all possible pairs between the views are created. In this case this leads to six pairs from the two detections above with the three below. The core joints of all pairs are triangulated into 3D proposals (b, steps (3,4)). Then they are reprojected into the 2D views (c, step (6)), and a distance-based error to the original 2D poses (visualized in black in c) is calculated. As can be seen in the image, the green and pink proposals clearly do not match to their original 2D poses, and get a very high error. The yellow and light-blue poses resulted from the flipped (man with woman) pairs and also have a notable error. All pairs with errors above a threshold are dropped in step (8). Only the remaining red and dark-blue pairs with low enough errors are used for the further steps (9-12).

In step (1), for each 2D pose, all possible pairs with poses from other views are created, since no association to real persons is known yet. In the next step (2), the pairs are filtered using the previous 3D poses, if available. For this, the 3D poses from the last frame are projected into the 2D views and a pixel-based distance threshold is used. The idea is to reduce the total number of pairs that need to be triangulated. In a pair of 2D persons in two different views, either both parts can match to one of the reprojected 3D poses, only one part is matched, or none of them. If both pairs match, it is likely a valid pair belonging to that 3D person, and it is kept. If only one part matches, a match is unlikely, because the 3D pose should be matchable in both views, so the second part is likely a pose from a different 3D person, and the pair is dropped. If none is matching, it is likely that a new 3D person is present in the current frame, and the pair is kept. In step (3) the core joints are extracted, which are the shoulders, hips, elbows, wrists, knees, and ankles. They are enough to associate the following 3D proposals to a person, and this reduces the computational complexity in the next steps. In step (4) each pair is triangulated to a 3D proposal (see Figure 2b). In the following step (5) clearly invalid proposals outside the observed room are dropped. In step (6) the remaining proposals are reprojected into the 2D views. Step (7) calculates a pixel-distance-based reprojection error to the pair's original 2D poses, and also prunes clearly invalid limbs. In step (8) pairs/proposals with a large error are dropped, as they are considered invalid matches. In step (9) the remaining 3D proposals are grouped in 3D space. Each of the 2D view pairs created a 3D proposal, and if they are close in 3D space, they likely belong to the same person, and are therefore grouped together. Grouping in 3D makes use of physical constraints, which allows a simpler and faster implementation. In step (10) the remaining pairs, which normally are less than in step (4), are triangulated again, but this time with all joints. In step (11) for each group the calculated 3D proposals are merged into a single 3D pose. To filter outliers, first for each joint an average location is calculated, and then for each proposal, the distance of this joint to the average is calculated. If the distance is too great, the joint is dropped. From the remaining joint proposals, the *top-k* closest to the average are selected, to drop outliers again, and the final joint location is calculated as the average of those. In general, this late fusion step allows for stronger outlier filtering, because multiple proposals can be used, but also works if a person is only visible in two views. In step (12) clearly invalid persons, with too few keypoints, that are too small or large, or outside the room, are dropped, and missing joints are filled with their next neighbors. The general idea behind the algorithmic structure is to use only the minimal necessary information at each step, drop bad proposals early on, and move expensive operations, like the full-body keypoint calculations, to later steps.

Optionally, in steps (13-14) the results can be tracked and smoothed over time. A simple distance-based threshold is used to assign poses to existing tracks. Just clipping the

joint movements to a maximum speed was preferred to more complex smoothing filters, because it does not introduce response latency to sudden direction changes, which could hurt real-time-position accuracy.

3.2. Method Comparison

There exist several differences to the other algorithmic approaches. In comparison to *mvpose* [10] no person identification by appearance similarities is required, the association is done only by geometric matching. *mv3dpose* [23] matches and groups 2D poses by epipolar geometry instead of the 3D triangulation and grouping used here. *PartAwarePose* [9] groups the 2D poses by assigning them to the 3D poses from the previous view, and uses a temporal filter to predict future locations from joints using previous observations to filter joints, which on the other hand can lead to prediction errors and costs additional computation time. *4DAssociation* [29] follows a bottom-up graph-based concept to match body-parts and connections over space and time to assemble the 3D poses afterward. *Bridgeman et al* [4] group the 2D poses to persons by first calculating correspondence costs for all pairs of 2D poses, using the minimal distance between two ray projections, calculate 3D poses and then iteratively calculate costs from the point-to-ray distance to update the 2D-3D matches. The results are further smoothed by a temporal filter. The source-code was not published. *QuickPose* [30] splits the detected 2D skeletons into all plausible part-graphs, calculates a ray-based distance between joints in different views, and uses a clustering algorithm to find matching parts. *Chen et al.* [6] iteratively match 2D poses from one view to the already existing 3D persons, and gradually update the matched 3D poses, before continuing with the next view. The last two algorithms are the fastest ones yet, according to the times stated in their papers, but their source-codes were not published, therefore a detailed comparison is not possible. In comparison to *VoxelKeypointFusion* [2] no voxel space is used, which results in much faster calculations, and its person association follows a bottom-up instead of the top-down approach used here.

3.3. Performance

Due to the relatively simple concept, the algorithm is very fast. In a test on the *shelf* [1] dataset, which shows up to four persons in five views, the algorithm requires an average triangulation time of only $0.1\,ms$. The per-step times are listed in Table 1.

While the initial version of the algorithm used *OpenCV* functions for most calculations, the current version uses standard *C++* functions only, because the overhead of the *OpenCV* functions was higher than the actual calculation time for the relatively small number of points. Instead of *OpenCV's SVD*-based triangulation, a simple mid-point triangulation was implemented, which was much faster and almost as accurate in the experiments. In summary the switch to standard *C++* functions led to a significant relative speedup of around $0.1\,ms$. Using only the core joints, and repeatedly filtering invalid poses helps to keep the subsequent steps efficient. The runtime is further influenced by the number of joints, which are by default 20. In the whole-body case that uses 136 joints, the average runtime increases to $0.4\,ms$. Depending on the integration of the algorithm, data conversion time can lead to further delays. The algorithm itself is implemented in *C++*, but an interface to *Python* (using *SWIG*) is provided as well.

Method	Time (µs)
Pose undistortion	14
Pair creation (1)	4
Pair filtering (2)	10
Triangulate and Score (3-7)	28
Grouping (9)	7
Triangulate and Score (10)	28
Merge (11)	12
Post-process (12)	7
Tracking (13-14)	1
Total	115

Table 1. Average time for different steps in microseconds.

As shown in Table 2 the proposed solution is multiple times faster than all the other algorithms, which proves its significant efficiency improvements. Note that all times reported in the different papers were measured on different hardware (see appendix), here a single core of an *AMD-7900X* was used, but the main speedup is clearly caused by the more efficient algorithm.

Method	Time (ms)
mvpose [10]	105
VoxelPose	103
mv3dpose	63
PlaneSweepPose	57
VoxelKeypointFusion [2]	48
Faster-VoxelPose	38
4DAssociation [30]	32
PartAwarePose [9]	10
Bridgeman et al. [4]	9.1
Chen et al. [6]	3.1
QuickPose [30]	2.9
RapidPoseTriangulation	0.1

Table 2. Time from 2D to 3D poses on *shelf* [1] in milliseconds.

4. Dataset Generalization

The most important part for machine learning models is their generalization to previously unknown setups. Therefore the proposed approach was evaluated on a variety of datasets. The datasets used for the evaluation were *human36m* [12], *shelf* [1], *campus* [1], *mvor* [20], and *panoptic* [15], similar as in other works. Additionally, the datasets *chi3d* [11], *tsinghua* [29] and *egohumans* [16] were added as well to allow an even broader evaluation. The metrics used are the same as in *VoxelKeypointFusion* [2], and evaluate the same 13 keypoints (2 shoulders, 2 hips, 2 elbows, 2 wrists, 2 knees, 2 ankles, 1 nose/head) across each dataset. The metrics follow common standards and are described in more

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
Faster-VoxelPose (synthetic)	91.3	75.5	98.8	88.3	75.0	**100**	0.2	99.5	36.2
VoxelKeypointFusion	**96.9**	81.7	**100**	64.3	**95.0**	**100**	**0**	**100**	8.7
RapidPoseTriangulation	96.8	**88.2**	99.9	**60.6**	94.5	**100**	**0**	**100**	**115**

Table 3. Transfer to *human36m* [12], all other results from [2].

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
Faster-VoxelPose	99.1	88.3	**100**	59.8	**99.4**	**100**	50.7	66.0	17.6
Faster-VoxelPose (synthetic)	99.0	87.9	**100**	58.8	99.0	**100**	48.8	67.7	17.4
MvP (synthetic)	98.6	94.1	99.7	51.8	97.1	**100**	82.2	30.2	8.5
mv3dpose	97.1	91.4	98.4	55.8	94.8	98.5	44.3	71.2	1.6
VoxelKeypointFusion	98.8	93.3	**100**	51.3	98.3	**100**	49.1	67.4	5.8
RapidPoseTriangulation	**99.2**	**94.4**	**100**	**47.5**	98.7	**100**	**41.3**	**74.0**	**66.6**

Table 4. Transfer to *shelf* [1], all other results from [2].

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
Faster-VoxelPose (synthetic)	65.9	41.3	74.4	104	39.6	74.7	**2.1**	84.8	25.3
mvpose	91.3	70.2	99.9	80.4	82.7	**100**	25.4	85.5	0.5
mv3dpose	84.1	64.4	93.4	135	62.5	94.9	10.1	**92.4**	2.8
PartAwarePose	93.2	78.4	98.9	**74.7**	**93.9**	98.9	22.7	86.8	5.8
VoxelKeypointFusion	91.1	74.6	99.9	84.4	80.3	**100**	35.5	78.4	7.8
RapidPoseTriangulation	**95.2**	**79.9**	**100**	75.2	**93.9**	**100**	15.9	91.4	**142**

Table 5. Transfer to *campus* [1], all other results from [2].

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
SimpleDepthPose [3]	**74.0**	**62.0**	**94.3**	113	**54.1**	**96.6**	23.5	**85.4**	**37.2**
Faster-VoxelPose (synthetic) [2]	37.9	28.1	45.5	**109**	29.4	46.1	**7.7**	61.5	30.0
VoxelKeypointFusion [2]	54.5	43.9	75.1	128	35.9	76.6	24.2	76.2	11.3
RapidPoseTriangulation	**59.0**	**44.6**	**76.7**	120	**39.9**	**77.9**	22.1	**77.9**	**121**

Table 6. Transfer to *mvor* [20] with depth images (first) and without (others).

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
Faster-VoxelPose	99.4	98.6	**99.9**	20.5	99.7	**99.9**	**1.0**	**99.5**	18.0
PRGnet	**99.5**	**99.1**	**99.9**	17.1	**99.9**	**99.9**	2.0	99.0	6.8
TEMPO	98.1	97.4	98.5	**16.8**	98.4	98.4	2.4	98.0	5.1
VoxelKeypointFusion	97.1	94.0	99.7	47.8	97.3	**99.9**	2.4	98.7	4.2
RapidPoseTriangulation	**99.1**	**96.5**	**99.8**	**30.6**	**99.3**	99.8	1.8	**99.0**	**57.5**

Table 7. Replication of *panoptic* [15] results and transfer without depth, all other results from [2].

detail in [2]. The *FPS* were measured on an *Nvidia-3090* as well. Due to brevity and readability reasons, for the datasets already evaluated in *VoxelKeypointFusion* [2], only the best results/algorithms are compared here.

4.1. Standard datasets

On the commonly used datasets, *RapidPoseTriangulation* shows a similar or better generalization performance than most state-of-the-art algorithms, while reaching an outstanding speed at the same time. See Tables 3, 4, 5, 6, 7.

Much of the *MPJPE* error in the *human36m* dataset results from different skeleton definitions. The original labels for example have higher hips, the nose more inside the head, and the ankles more at the heels, than the *COCO* [17] skeleton definition of the 2D pose estimation network. From a human perspective though, the predictions would often match better to the images than the original labels.

Regarding the *campus & shelf* datasets, a visual inspection of samples with large errors showed that a majority of them are caused by label errors, and not by the algorithm itself. Therefore it needs to be noted that the comparability of the results of well-performing algorithms seems to be somewhat limited.

On *mvor* the algorithm suffers, similar to the other RGB-only approaches, from high occlusions and few viewpoints.

Many errors in the *panoptic* dataset involve the lower body joints, especially the ankles. While most upper body joints have an average error between 15-25 mm, the hips and knees have around 35-40 mm, and the ankles around 50 mm. This is mainly caused by the position of the cameras, which often cut off part of the legs. The 2D pose estimation network, however, still predicts those joints at the lower edge of the image (so for example an ankle is predicted at the height of the knee). This leads to a large error in the triangulation step because the 3D joint is then estimated with a wrong depth, which cannot always be detected as an outlier. For future optimization, this might be better handled if the 2D pose estimation network could predict the visibility of the keypoints as well, so the algorithm could prefer directly visible keypoints over estimated ones.

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
VoxelPose (synthetic)	97.4	88.4	99.4	72.2	90.5	**100**	0.5	99.7	15.2
Faster-VoxelPose (synthetic)	97.3	88.2	99.2	69.3	94.9	99.6	0.8	99.4	31.9
PRGnet	95.8	82.9	99.1	80.1	85.0	**100**	10.2	94.6	5.0
TEMPO	89.0	81.1	90.7	66.5	86.4	91.0	6.6	92.2	8.9
SelfPose3d	79.7	71.5	87.2	97.8	66.5	89.7	79.8	32.9	5.1
mvpose	98.5	85.2	**100**	66.7	98.3	**100**	0.5	99.7	0.2
mv3dpose	58.9	53.8	61.1	71.8	55.9	61.4	2.2	75.4	2.6
PartAwarePose	89.1	81.6	92.9	77.8	84.0	94.0	8.7	92.5	3.4
VoxelKeypointFusion	94.8	81.7	99.1	68.5	95.5	99.4	0.1	99.6	8.0
RapidPoseTriangulation	**99.0**	**90.5**	99.9	**60.3**	**98.6**	**100**	**0**	**100**	**97.3**

Table 8. Transfer to *chi3d* [11]

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
VoxelPose (synthetic)	97.8	94.1	99.1	58.9	94.1	99.5	5.7	96.8	5.1
Faster-VoxelPose (synthetic)	94.8	87.9	98.6	66.3	89.7	99.0	9.2	94.8	7.7
PRGnet	96.8	88.7	99.9	67.8	93.2	**100**	10.5	94.5	6.3
TEMPO	89.2	85.7	90.4	57.3	89.4	90.3	1.4	94.3	5.0
SelfPose3d	87.2	83.1	89.2	65.9	83.0	90.2	51.2	63.3	4.9
mvpose	91.4	81.5	99.3	79.2	88.6	99.7	4.7	97.4	0.4
mv3dpose	68.6	64.8	71.0	63.3	66.8	71.3	14.6	77.7	1.2
PartAwarePose	89.9	81.7	94.3	79.7	78.6	94.9	6.7	94.1	0.8
VoxelKeypointFusion	98.0	95.9	99.5	**51.1**	95.1	99.8	0.9	99.4	3.1
RapidPoseTriangulation	**98.7**	**96.0**	**99.8**	52.8	**95.7**	**100**	**0**	**100**	**48.4**

Table 9. Transfer to *tsinghua* [29]

4.2. Additional datasets

To allow an even broader comparison, two further datasets were added, following the same scheme as in *VoxelKeypointFusion* [2], and using their *skelda* dataset library.

The first one is the *chi3d* [11] dataset, which shows two persons interacting with each other. It has the challenge that the persons are often very close to each other, which makes it hard to distinguish between them. The setup is similar to *human36m* and also contains four cameras. The results in Table 8 show that generally most persons are detected, with a similar performance as in *human36m*. Regarding *RapidPoseTriangulation*, it could be seen that in some cases the 2D pose estimation struggled to correctly detect the joints between very close persons, and the 3D triangulation then did not get enough inputs to correctly triangulate all joints and could miss a person. One could reduce the reprojection matching threshold to better handle such problematic 2D detections, but at the cost of increasing the number of false positives, or alternatively use the tracking information to fill-up missed persons with their last positions. The authors of *CloseMoCap* [19] also reported very good transfer results on this dataset, reaching a *PCK@50* of 94.3 on average. They also evaluated *VoxelPose*, *Faster-VoxelPose* and *mvpose* on the same dataset and reached much better results for them as well. Additionally, they reported per-joint errors on a subset of the dataset and found that the hip joints had the best results, whereas here the hip joints are among the worst results. In the labels, however, it can be seen that the hips are positioned higher on the body than in the *COCO* skeleton definition, similar as already in *human36m*. So in conclusion, the mismatch is likely caused by parts of the evaluation, or the use of different labels, somewhere in *CloseMoCap*, but because their source-code was not published, this assumption cannot be verified.

The second dataset is *tsinghua* [29], which shows people moving around in a room, while they are being recorded from six different cameras. To prevent multiple trainings for the synthetic models, only the first of the two sequences was evaluated. The results in Table 9 show that almost all models perform relatively well on this dataset, but still many of them miss a few persons.

Under the assumption that the general goal of all developed pose estimators is their usage in practical applications, their generalization to unseen datasets is very important. As can be seen in the experiments above, many models have setups in which they show good, and others in which they show bad performance. For a simpler algorithm comparison, the average performance on all six datasets was calculated. As can be easily seen in Table 10, *RapidPoseTriangulation* shows the overall best performance.

Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	FPS
VoxelPose (synthetic)	81.4	68.7	90.5	100	64.6	92.1	26.7	79.3	10.7
Faster-VoxelPose (synthetic)	81.0	68.1	86.1	82.6	71.3	86.6	**11.5**	84.6	24.8
PRGnet	63.5	52.4	68.4	134	51.3	69.3	28.8	63.3	9.9
TEMPO	69.1	59.9	72.2	79.2	61.9	72.4	21.2	69.7	9.9
SelfPose3d	68.0	55.6	77.4	115	52.4	79.0	68.3	42.5	7.0
mvpose	83.4	70.5	90.0	81.6	76.8	90.4	19.6	83.3	0.4
mv3dpose	61.0	53.9	64.4	110	54.6	65.0	21.5	66.4	2.3
PartAwarePose	79.1	71.0	83.9	91.5	74.0	84.6	17.7	79.9	3.9
VoxelKeypointFusion	89.0	78.5	95.6	74.6	83.4	96.0	18.3	86.8	7.5
RapidPoseTriangulation	**91.3**	**82.3**	**96.0**	**69.4**	**86.9**	**96.3**	13.2	**90.5**	**98.4**

Table 10. Averaged generalization on all six unseen datasets (*human36m*, *shelf*, *campus*, *mvor*, *chi3d*, *tsinghua*).

Subset	Method	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	Time: Demosaic-2D-3D		
legoassemble	VoxelPose (synthetic)	89.9	74.4	93.8	94.7	65.3	96.1	71.2	44.4		392	
	Faster-VoxelPose (synthetic)	75.4	64.1	94.2	107	54.0	95.3	**0**	97.6		239	
	VoxelKeypointFusion	99.4	97.2	99.4	37.0	99.4	99.7	**0**	99.9		205	271
	RapidPoseTriangulation	**100**	**99.0**	**100**	**25.3**	**100**	**100**	**0**	**100**	11.2	**23.3**	**0.3**
tennis	VoxelPose (synthetic)	66.9	29.5	86.5	128	3.9	87.0	1.3	92.5		1077	
	VoxelKeypointFusion	98.2	90.6	100	64.6	98.1	**100**	**0**	**100**		434	679
	RapidPoseTriangulation	**99.9**	**99.5**	**100**	**22.7**	**99.7**	**100**	**0**	**100**	26.8	**50.1**	**0.6**
tagging	VoxelKeypointFusion	89.1	84.2	92.4	67.1	84.3	94.3	8.0	93.1		207	379
	RapidPoseTriangulation	**97.5**	**92.9**	**99.4**	**47.9**	**90.0**	**100**	**4.1**	**97.9**	11.0	**26.5**	**0.3**
fencing	VoxelKeypointFusion	**100**	97.7	**100**	37.5	**100**	**100**	69.0	47.3		335	420
	RapidPoseTriangulation	**100**	**99.6**	**100**	**23.9**	**100**	**100**	**50.0**	**66.7**	12.3	**27.9**	**0.4**
basketball	VoxelKeypointFusion	96.3	92.2	97.6	53.5	94.8	98.2	50.9	65.5		212	804
	RapidPoseTriangulation	**99.6**	**96.3**	**100**	**40.8**	**99.7**	**100**	**24.8**	**85.8**	11.0	**27.1**	**0.4**
volleyball	VoxelKeypointFusion	97.1	94.1	98.0	48.5	96.1	98.6	25.2	85.0		449	1004
	RapidPoseTriangulation	**100**	**99.2**	**100**	**27.6**	**100**	**100**	**5.8**	**97.0**	18.0	**56.4**	**1.1**
badminton	VoxelKeypointFusion	99.8	99.0	**100**	36.5	**100**	**100**	16.2	91.2		391	846
	RapidPoseTriangulation	**99.9**	**99.5**	**100**	**24.3**	**100**	**100**	**0.0**	**100**	17.5	**42.1**	**0.7**

Table 11. Transfer to *egohumans* [16]

4.3. Using more cameras

The previous results demonstrate that the number of cameras has a large impact on the performance of the algorithms. Especially *mvor* shows that using only three cameras in the presence of strong occlusions leads to a large decrease in performance. From an application point of view, it is therefore recommended to use at least five or even more cameras, if this is possible.

To evaluate the impact of a high number of cameras, the *egohumans* [16] dataset was selected. It shows between 2-8 people in different indoor and outdoor settings (like building with lego bricks, fencing, or playing tag, basketball, volleyball, badminton, or tennis). They are watched by 8-20 cameras. The supervised space is up to $25 \times 14\,m$ in size, which is much larger compared to the other datasets. The last recording of each setting was selected as test set, and each recording was subsampled into parts of $3s$ consecutive motions with larger gaps in between as well.

Because the cameras use fisheye lenses, the images are more strongly distorted than in the other datasets. Therefore, a different undistortion method is required. Since this requires larger changes in the pipeline of most algorithms, only a subset of the better performing algorithms was tested on this dataset. In some cases the poses can be undistorted after the 2D estimation step, in other cases, the images need to be undistorted instead. *VoxelPose* and *Faster-VoxelPose*, for which the images are undistorted beforehand, both have problems with a direct transfer to the *legoassemble* subset, but again show much better performance after synthetic training. The same holds true for *VoxelPose* on the *tennis* subset, *Faster-VoxelPose* on the other hand does not work at all here. The synthetic training always failed with an out-of-memory error, even at the lowest possible batch-size. For *mvpose* the number of views was too high, the six cameras from *tsinghua* already took up the complete GPU memory. *TEMPO* [8] described an evaluation on this dataset, in which they reported an average transfer error of $120\,mm$ (or $36.7\,mm$ after training) on this dataset, but they did not provide further information about the evaluation in the paper or their code. *VoxelKeypointFusion* proves its generalization capabilities with a very good performance, which is only surpassed by *RapidPoseTriangulation*.

4.4. About real-time performance

Regarding real-time performance with many cameras, the results show that while the 3D part of *RapidPoseTriangulation* is very fast and takes at most $1\,ms$, the majority of the time is currently spent on 2D pose estimation.

In comparison to the original implementation of *RTMPose*, the preprocessing pipeline and network interfacing are already optimized. Some parts of the preprocessing were moved to the GPU, and the image is transferred as *uint8* type to decrease transfer time. The box cropping and padding were optimized, and for image resizing nearest-neighbor instead of linear interpolation is used for further speedups. Everything was implemented in *C++* as well. Cropping and resizing directly on the GPU did not show notable speedups, as transferring the full image counterweighted its benefits.

To further reduce the input latency in real applications, the *Bayer* format, which is the default layout of image sensors in most cameras, is used directly. Instead of having three RGB channels, this format only has one channel, in which every other pixel has a single red, green, or blue color only. The conversion into RGB is now integrated into the preprocessing pipeline. For the dataset evaluation, the RGB images were first converted back to *Bayer* encoding. Interestingly, this back-and-forth conversion notably improved the *MPJPE* score in many datasets, presumably because the demosaicing algorithm of *OpenCV* is better than the default one of many cameras. The demosaicing time is already included in the *FPS* values of the previous tables. In the case of *FullHD* resolution, it is around $0.2\,ms$ per image. With *4K* inputs, like in the *egohumans* dataset, it increases to a notable $1.4\,ms$ per image, so demosaicing after cropping and resizing, or directly using *Bayer* images as network input, might be future optimization possibilities.

Method	PCP	PCK@100/500		MPJPE (All—Body—Face—Hands)	Recall@100/500		Invalid	FPS
VoxelKeypointFusion [2]	99.7	**96.8**	99.8	**40.5** — 37.3 — **38.3** — **40.7**	98.5	**100**	0.5	4.9
RapidPoseTriangulation	**99.8**	88.3	**100**	45.7 — **29.1** — 54.3 — 41.8	**100**	**100**	**0**	**82.6**

Table 12. Whole-body estimation on *h3wb*.

In real-time applications, it is also important to weigh algorithmic accuracy versus its latency. Because persons normally move around, estimations with a high latency result in a larger error to the real position of the human. For example, a person moving with $2\,m/s$ would move $20\,mm$ in only $10\,ms$, which is already more than the average *MPJPE* difference between the better-performing algorithms from Table 10. Algorithmic speed therefore can bring a significant real-time-position accuracy improvement.

Another important latency factor in real-time systems is the transfer time of the image from the camera to the computer, especially if multiple cameras feed one host. Transferring a single *FullHD* image with $1920 \times 1080 \times 1$ pixels in *Bayer* encoding over a $1\,Gb/s$ network already takes around $17\,ms$. And this has to be stacked for every further camera. While compressing the image could reduce this time, it would require extra processing time and reduce the image quality. A different solution could be adding a small computer to each camera, thus creating a smart-edge device, which estimates the poses for only one image and sends only a few keypoint coordinates to the main computer. Because only a single image is processed per time-step, this also would allow less powerful hardware. A *Raspberry Pi* with the *AI-Hat* could already be sufficient. If the price is not a concern, each camera could be equipped with one standard GPU though. The RTX-3090 used before takes about $1\,ms$ per image and $1\,ms$ per person to estimate their 2D pose. With an RTX-4080 this was even 45% faster, and with an RTX-5090 (which has about $2.5\times$ more FLOPS than the RTX-3090), it should be possible to achieve full image-to-pose 3D prediction in only one millisecond.

5. Whole-body Estimation

As already stated in *VoxelKeypointFusion* [2], a significant benefit of algorithmic approaches over the learning-based ones, is their ability to handle input keypoints that were not part of the training dataset, such as *whole-body* joints.

To evaluate the performance, the *h3wb* dataset [31] was used, which extended a part of the *human36m* dataset with whole-body keypoints (17 body, 6 foot, 68 face, 42 hand). For simplification of the labeling process, only frames with good visibility of the actor were used. Therefore, the dataset is somewhat simpler than the original one (the default model of *RapidPoseTriangulation* reaches a *MPJPE* of $23.7\,mm$, and the authors estimated around $17\,mm$ labeling error). The whole-body results can be found in Table 12. Since this dataset is normally used for single-view 2D-to-3D pose lifting, no other comparable results were found.

One problem of *VoxelKeypointFusion* are the artifacts of the voxelization process, which, due to the discretization, push some keypoints closer together than they really are. This was especially visible at the finger keypoints, which were often agglomerated together instead of showing distinct finger lines. *RapidPoseTriangulation* does not have this problem anymore, since it directly triangulates in continuous coordinates. On the other hand, smaller errors in the 2D keypoint prediction can lead to larger errors in 3D. For example, if a keypoint is visible from three views, *VoxelKeypointFusion* merges them all at once using heatmap-beams. So if there are small 2D position errors, the beams of all three views still partially overlap. *RapidPoseTriangulation* on the other hand triangulates each pair of views separately, and merges them afterward, but the errors do not necessarily cancel themselves out again. This is more severe if the keypoints can only be seen in very few views, such as in this dataset, since it is not possible to detect and drop outliers before the merging step.

A large difference can be seen in the inference speed, especially in the 3D part. *VoxelKeypointFusion* took on average $122\,ms$ for the whole-body prediction. *RapidPoseTriangulation* on the other hand only took $0.1\,ms$ for the whole-body prediction, which is around $1100\times$ faster.

6. Conclusion

This work presented *RapidPoseTriangulation*, a new algorithm for 3D human pose triangulation. Its basic idea is to triangulate pairs of 2D poses from different views separately, and group and merge them to the final poses directly in 3D space. This relatively simple concept makes it very efficient, and much faster than any previous algorithm.

In a broad evaluation of the algorithm on several datasets, it showed state-of-the-art performance. It scales well with an increased number of cameras and is especially suited for real-time applications. Besides that, the results also show that a relatively simple algebraic approach still can outperform many recent fully differentiable or end-to-end learned multi-view methods. This suggests that the current trend of increasing model complexity does not necessarily translate into better exploitation of the geometric structure.

In summary, it is one of the best and by far the fastest method for multi-view multi-person human pose estimation that is currently available. In contrast to most prior works, it is also able to predict whole-body poses, which can be used to implement more advanced follow-up tasks. By making the source-code publicly available, it is hoped that this work can contribute to the development of more intuitive and safer human-computer or human-robot interactions.

References

[1] Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic. 3D pictorial structures for multiple human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1669–1676, 2014. 4, 5

[2] Daniel Bermuth, Alexander Poeppel, and Wolfgang Reif. VoxelKeypointFusion: Generalizable Multi-View Multi-Person Pose Estimation. *arXiv preprint arXiv:2410.18723*, 2024. 1, 2, 4, 5, 6, 8, 12

[3] Daniel Bermuth, Alexander Poeppel, and Wolfgang Reif. SimpleDepthPose: Fast and Reliable Human Pose Estimation with RGBD-Images. *arXiv preprint arXiv:2501.18478*, 2025. 5

[4] Lewis Bridgeman, Marco Volino, Jean-Yves Guillemaut, and Adrian Hilton. Multi-person 3d pose estimation and tracking in sports. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 4

[5] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019. 12

[6] Long Chen, Haizhou Ai, Rui Chen, Zijie Zhuang, and Shuang Liu. Cross-view tracking for multi-human 3d pose estimation at over 100 fps. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[7] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018. 12

[8] Rohan Choudhury, Kris M Kitani, and László A Jeni. TEMPO: Efficient multi-view pose estimation, tracking, and forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14750–14760, 2023. 2, 7

[9] Hau Chu, Jia-Hong Lee, Yao-Chih Lee, Ching-Hsien Hsu, Jia-Da Li, and Chu-Song Chen. Part-aware measurement for robust multi-view multi-human 3d pose estimation and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1472–1481, 2021. 2, 4

[10] Junting Dong, Wen Jiang, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Fast and Robust Multi-Person 3D Pose Estimation from Multiple Views. 2019. 2, 4

[11] Mihai Fieraru, Mihai Zanfir, Elisabeta Oneata, Alin-Ionut Popa, Vlad Olaru, and Cristian Sminchisescu. Three-dimensional reconstruction of human interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7214–7223, 2020. 3, 4, 6

[12] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. 4, 5

[13] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7718–7727, 2019. 2

[14] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose. *arXiv preprint arXiv:2303.07399*, 2023. 2, 12

[15] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015. 4, 5

[16] Rawal Khirodkar, Aayush Bansal, Lingni Ma, Richard Newcombe, Minh Vo, and Kris Kitani. Ego-Humans: An Ego-Centric 3D Multi-Human Benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19807–19819, 2023. 1, 4, 7

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[18] N Dinesh Reddy, Laurent Guigues, Leonid Pishchulin, Jayan Eledath, and Srinivasa G Narasimhan. Tessetrack: End-to-end learnable multi-person articulated 3d pose tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15190–15200, 2021. 2

[19] Qing Shuai, Zhiyuan Yu, Zhize Zhou, Lixin Fan, Haijun Yang, Can Yang, and Xiaowei Zhou. Reconstructing Close Human Interactions from Multiple Views. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023. 2, 6

[20] Vinkle Srivastav, Thibaut Issenhuth, Abdolrahim Kadkhodamohammadi, Michel de Mathelin, Afshin Gangi, and Nicolas Padoy. MVOR: A multi-view RGB-D operating room dataset for 2D and 3D human pose estimation. *arXiv preprint arXiv:1808.08180*, 2018. 4, 5

[21] Vinkle Srivastav, Keqi Chen, and Nicolas Padoy. SelfPose3d: Self-Supervised Multi-Person Multi-View 3d Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2502–2512, 2024. 2

[22] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019. 12

[23] Julian Tanke and Juergen Gall. Iterative Greedy Matching for 3D Human Pose Tracking from Multiple Views. In *German Conference on Pattern Recognition*, 2019. 2, 4

[24] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[25] Tao Wang, Jianfeng Zhang, Yujun Cai, Shuicheng Yan, and Jiashi Feng. Direct Multi-view Multi-person 3D Human Pose Estimation. *Advances in Neural Information Processing Systems*, 2021. 2

[26] Size Wu, Sheng Jin, Wentao Liu, Lei Bai, Chen Qian, Dong Liu, and Wanli Ouyang. Graph-based 3d multi-person pose estimation using multi-view images. In *ICCV*, 2021. 2

[27] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of*

the European conference on computer vision (ECCV), pages 466–481, 2018. 12

[28] Hang Ye, Wentao Zhu, Chunyu Wang, Rujie Wu, and Yizhou Wang. Faster VoxelPose: Real-time 3D Human Pose Estimation by Orthographic Projection. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[29] Yuxiang Zhang, Liang An, Tao Yu, xiu Li, Kun Li, and Yebin Liu. 4D Association Graph for Realtime Multi-person Motion Capture Using Multiple Video Cameras. In *IEEE International Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2020. 4, 6

[30] Zhize Zhou, Qing Shuai, Yize Wang, Qi Fang, Xiaopeng Ji, Fashuai Li, Hujun Bao, and Xiaowei Zhou. Quickpose: Real-time multi-view multi-person pose estimation in crowded scenes. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 4

[31] Yue Zhu, Nermin Samet, and David Picard. H3WB: Human3.6M 3D WholeBody Dataset and Benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20166–20177, 2023. 8

A. Ablation Studies

The following section contains the ablation studies that were left out of the main paper for space reasons.

A.1. Number of cameras

For completeness, an experiment with *panoptic* using 3, 5, 7, 10, or 31 cameras was conducted. As expected, the results show that more cameras lead to better results. The frame-rate also slows down, with the *2D*-part having the most impact, and the *3D*-part having a slight exponential increase, even though its impact is still very small.

For real-world applications, this experiment is not very meaningful though, since the required number of cameras is highly dependent on the targeted environment. Depending on the setup, 3 cameras can be enough or much too few, as in *campus* versus *mvor*. *VP* and *F-VP* both conducted an experiment for 3 cameras on *panoptic* in which they had only very small performance drops, but their performance on *campus* and *mvor* showed a large drop, even though they also have 3 cameras. For a real-world application, it is therefore better to first select a similar dataset and then evaluate the impact of the number of cameras and their best positioning. Since it is not possible to evaluate all combinations of datasets, cameras, and positions, the recommendation for developers is to use the provided source-code and run a personalized evaluation on the most similar dataset.

A.2. Algorithmic ablations

In general, the algorithm is very robust against small changes, and only a few of them have a notable impact on the results. Ignoring the 2D confidence scores instead to integrating them into the 3D keypoint score, results in a slightly higher *MPJPE*. Removing the pair pre-filtering step, which removes pairs of poses before the triangulation, has minimal impact on most scores, but notably increases the triangulation time, since more pairs have to be calculated. The two outlier removal steps, before merging the proposals of one group to a single person, notably improve the accuracy. Lowering the error threshold, which filters poor triangulations before merging, slightly increases *MPJPE* and the invalid count. The postprocessing steps of dropping clearly invalid persons or replacing clearly wrong joints with their neighbors also improves the *MPJPE* and reduces the invalid person count in some datasets. Using only the triangulation error or only the reprojection error for the 3D keypoint score calculation, instead of a combination of both, has a slightly negative impact on the *MPJPE*. Using only the triangulation score is slightly faster though, because the more complex reprojection can be skipped. Especially in datasets with more cameras, it is recommended to increase the minimal required group size. This results in dropping invalid 3D proposals which do not originate from the same person, but have a similar pose in two or more views. The higher the number of cameras the higher the chance for it. Since they are from different persons, they are triangulated to some mathematically valid positions somewhere in the 3D space. Proposals from the same person on the other hand start to build clusters around their real position, by which they can be recognized as such. The tracking step has only a minimal impact on most datasets, since it does not smooth the movement trajectories. At fast movements it might slightly reduce the position accuracy because it can slow down some position changes. On the other hand it can prevent physically impossible fast jitter movements. In case a person is not visible for a few frames, it can also prevent the person from being lost completely, since their last position is returned then, but it will also keep persons moving out of the room alive for a few extra frames. Therefore tracking is considered as an optional extension.

Number of cameras	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	Time-3D	FPS
3	90.4	83.7	93.5	58.2	85.2	94.5	1.1	96.7	0.1	95.8
5	99.1	96.5	99.8	30.6	99.3	99.8	1.8	99.0	0.2	57.5
7	99.5	97.2	99.9	27.9	99.9	99.9	3.7	98.1	0.3	40.9
10	99.7	98.0	99.9	23.9	99.9	99.9	3.0	98.4	0.5	29.6
31	99.8	98.7	99.9	19.3	99.9	99.9	2.8	98.6	5.0	9.1

Table 13. Using different numbers of cameras on *panoptic*.

Ablation	Dataset	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1	Time-3D
default	shelf	99.2	94.4	100	47.5	98.7	100	41.3	74.0	0.1
score without 2D confidence	shelf	99.2	94.4	100	47.7	98.7	100	41.3	74.0	0.1
no pair pre-filtering	shelf	99.2	94.4	100	47.6	98.7	100	40.4	74.6	0.2
keep topk outliers	shelf	98.9	93.8	100	49.8	97.9	100	41.3	74.0	0.1
keep topk+distance outliers	shelf	98.9	93.7	100	49.9	97.9	100	41.3	74.0	0.1
min_match_score=0.8	shelf	99.2	94.5	100	47.7	97.9	100	46.9	69.3	0.1
default	panoptic	99.1	96.5	99.8	30.6	99.3	99.8	1.8	99.0	0.2
no postprocessing steps	panoptic	99.0	96.3	99.8	31.6	99.2	99.8	1.8	99.0	0.1
default	badminton	99.9	99.5	100	24.3	100	100	0.0	100	0.7
only triangulation error	badminton	99.9	99.6	100	24.6	100	100	0.0	100	0.5
only reprojection error	badminton	100	99.5	100	24.4	100	100	0.0	100	0.7
min_group_size=1	badminton	99.4	98.9	99.8	27.1	99.3	100	39.0	75.8	1.0
default	tagging	97.5	92.9	99.4	47.9	90.0	100	4.1	97.9	0.3
no tracking	tagging	97.5	93.0	99.3	46.7	91.8	99.8	3.9	97.9	0.3

Table 14. Different algorithmic ablations.

B. Further Remarks

The following section contains some further remarks about the algorithmic performance.

B.1. Computation Comparison

The computation efficiency comparison was made as fair as technically possible. Results taken from [2] in Table 3 onwards used either RTX3090 (same as here), or GTX1080 (*mvpose, mv3dpose, PartAwarePose*) because they did not work on newer GPUs. In Table 2, the methods *mvpose, VoxelPose, Faster-VoxelPose, PlaneSweepPose, VoxelKeypointFusion* use a GPU, the rest is on CPU. The fastest three in the table ([4,5,26]) are closed source, so comparison on the same hardware is not possible. [4] used an Intel-i7 $3.2\,GHz$, with parallelization. [26] and *PartAwarePose* used a $3.7\,GHz$ CPU, [5] a $2.2\,GHz$ CPU, the AMD7900X used here has $4.7\,GHz$, all those works used a single core. *RapidPoseTriangulation* is at least $30\times$ faster, which is much more than the GHz difference, so the main speedup certainly is from the algorithm itself.

B.2. Different 2D Estimators

The referenced previous works use a wide variety of 2D detectors, and all results from Table 3 onwards are with the originally used detectors. *VoxelPose*, *PRGnet*, *MvP*, *SelfPose3d* all use *ResNet* [27], *Faster-VP* and *TEMPO* switched to the newer *HRNet* [22]. *mvpose* uses *CPN* [7], *mv3dpose* uses *OpenPose* [5], *PartAwarePose* uses *HRNet* [22]. *VoxelKeypointFusion* uses *RTMPose* [14], same as this work. In general, all previous works that switched their pose detector to newer models did not investigate this impact on performance. Speaking from experience, it is reasonable that this is not a standard practice in the field, because it takes a lot of effort to adjust often poorly documented code to inject different poses into it. For this reason, this work only conducted a small experiment with *mv3dpose* and *PartAwarePose* using exactly the same 2D poses as *RapidPoseTriangulation*, instead of the ones from their original *OpenPose* and *HRNet* pipelines (Table 15). For *mv3dpose* the results show improvements the localization accuracy (*MPJPE*), but a notable drop in person detection (*Recall@500*). The results of *PartAwarePose* show only marginal changes, mainly a slightly better *MPJPE* (mostly caused by better hips, while many joints even got worse). All scores are still clearly behind *RapidPoseTriangulation*, so the detector change is not the main reason for the better performance.

B.3. Additional Reasoning

The (voxel-based) learned methods all have generalization problems, they work well on the same dataset (*panoptic*), but show notable loss in setup switches. The synthetic training reduced this somewhat in many cases, but not to the full extend. They also do not allow using arbitrary keypoints, but require matching training datasets instead. This was already evaluated in more detail in [2]. And as *VoxelKeypointFusion* has shown, the problems do not come from the voxel representation itself. But even if those problems could be solved, those methods are rather slow (as shown in Table 2), which makes them less suitable for real-time applications, and, as *VoxelKeypointFusion* has shown, adding a lot more keypoints makes them even slower.

The algebraic methods often fail in person association. For example in Table 2+3 from [2] *mv3dpose* and *PartAwarePose* are not able to detect all persons (Recall@500), and as shown here in Table 15, this is not due to the 2D detector. The same problem can be seen in Table 5 from [2] or Table 9. The better performance of *RapidPoseTriangulation* in this regard can be explained by the different matching strategy. Unlike the others, *RPT* triangulates all possible view-pairs to 3D poses, and then drops those with large errors (steps 1-8). So if a person is seen in only two images it is very likely to be reconstructed. This makes *RapidPoseTriangulation* so reliable, which is why this approach was chosen. Grouping all those 3D proposals to persons (step 9) is very robust as well, because it is highly unlikely that two persons have very similar 3D poses while standing at basically the same spot. Because there are no projection caused similarities this is more reliable than 2D pose matching across views, especially in more crowded scenes, which have a higher probability of similar looking persons, and therefore more false matches. And compared to appearance based matching, it has no problems with similar looking persons (like in *MVOR*), and it is orders of magnitudes faster. While a larger number of persons also affects the likelihood of wrong 3D proposals generated in *RapidPoseTriangulation*, they can easily be filtered out by setting a higher minimal group size.

Regarding the joint accuracy (PCK@100), having multiple 3D proposals per person from the different view-pairs allows for a very efficient but robust outlier detection. At its core it's based on majority voting, which removes very bad proposals, without them influencing the final position, which would not be possible with iterative matching or (weighted) averaging like in other works.

Method	Dataset	PCP	PCK@100/500		MPJPE	Recall@100/500		Invalid	F1
mv3dpose	shelf	92.2 (-4.9)	88.2 (-3.4)	98.4 (-4.8)	51.1 (-4.7)	90.6 (-4.2)	93.7 (-4.8)	**40.6** (-3.7)	72.7 (+1.5)
PartAwarePose	shelf	98.3 (0.0)	93.5 (+0.8)	99.0 (0.0)	48.1 (-3.3)	97.5 (-1.0)	99.2 (0.0)	49.4 (+2.0)	67.0 (-1.7)
RapidPoseTriangulation	shelf	**99.2**	**94.4**	**100**	**47.5**	**98.7**	**100**	41.3	**74.0**
mv3dpose	campus	87.1 (+3.0)	71.4 (+7.0)	91.4 (-2.0)	75.7 (-59)	81.6 (+19)	91.8 (-3.1)	17.5 (+7.4)	86.9 (-5.5)
PartAwarePose	campus	94.2 (+1.0)	78.4 (0.0)	98.9 (0.0)	**74.6** (-0.1)	92.0 (-1.9)	98.9 (0.0)	17.1 (-5.6)	90.2 (+3.4)
RapidPoseTriangulation	campus	**95.2**	**79.9**	**100**	75.2	**93.9**	**100**	**15.9**	**91.4**

Table 15. Replacing the original 2D pose inputs with those of *RTMPose*, and the change to the original results.