

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

Daniel Bermuth

ISSE

University of Augsburg, Germany

daniel.bermuth@uni-a.de

Alexander Poeppel

ISSE

University of Augsburg

poeppel@isse.de

Wolfgang Reif

ISSE

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 a very important task in many advanced computer vision applications. From enhancing virtual reality experiences to improving the collaboration with humans in robotic systems, precise human pose estimation plays a crucial role in bridging the gap between digital systems and 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 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 these issues, multi-view approaches have become more common.

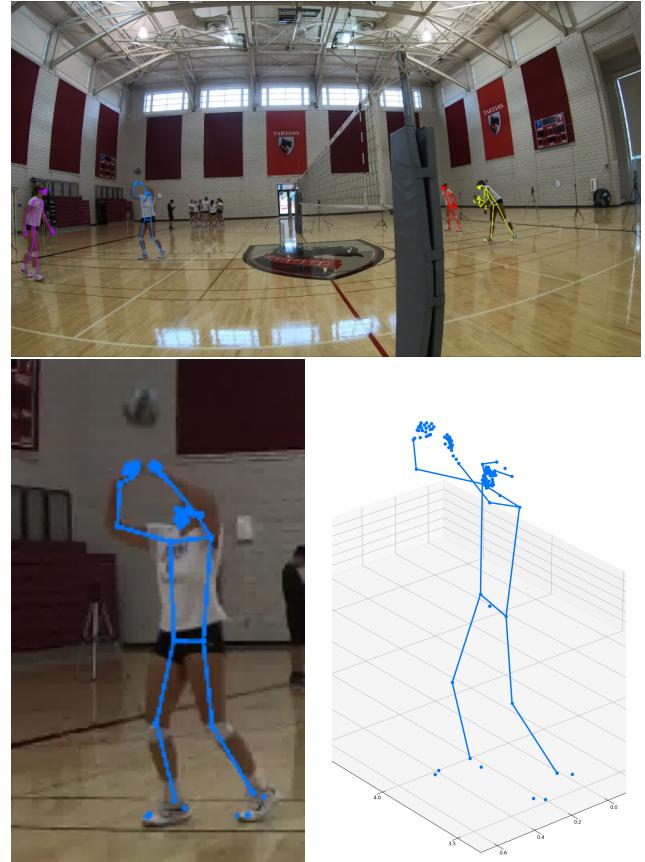


Figure 1. Example of a multi-person whole-body pose estimation from multiple camera views in a volleyball game (from the *egohumans* dataset [14]). On top, the full image of one camera with the projections of all the detected poses, on the bottom-left a zoom-in on one player, and on the bottom-right her detected 3D pose.

By capturing the scene from multiple angles simultaneously, these systems can overcome many of the limitations found in single-view methods.

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. 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 source-code of the method is 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 on 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* [20] was one of the first concepts, building upon the research of *Iskakov et al.* [11] and extending it to multi-person estimations. This method projects joint heatmaps from 2D images into a 3D voxelized space, then first estimates a central point for each individual, which is subsequently utilized to generate and extract a focused cube surrounding each person. Afterward, a secondary neural network computes the joint positions. *Faster-VoxelPose* [23] enhanced this technique by restructuring the 3D voxel network into several 2D and 1D projections, thereby boosting efficiency. *TesseTrack* [15] and *TEMPO* [6] incorporated a temporal dimension into the voxel space to refine poses across sequential frames. Alternative approaches such as *PRGnet* [22] employ a graph-based methodology or directly infer 3D poses from 2D features, like in *MvP* [21]. *Self-Pose3d* [18] 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* [8] 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* [19] utilizes a graph-matching approach to assign poses through epipo-

lar geometry and incorporates temporal information to handle missing joint data. *PartAwarePose* [7] 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. This method involves randomly positioning 3D poses from an alternative dataset within 3D space and subsequently back-projecting them onto the camera perspectives of the target setup. These resulting 2D poses, after undergoing various augmentations, are then employed to learn the 3D reconstruction process. In the closed-source approach of *CloseMoCap* [16] 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* [12], similar as in *VoxelKeypointFusion*, 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

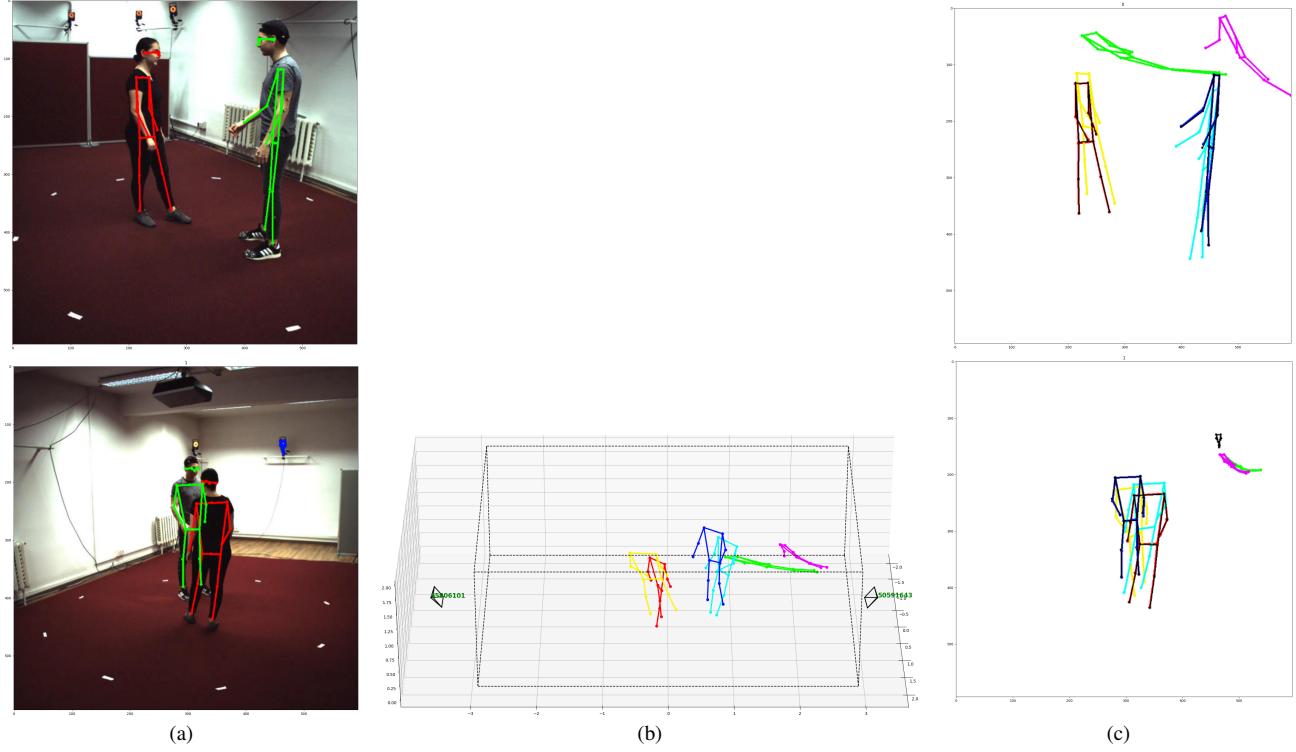


Figure 2. Obtaining 3D proposals. The process starts with 2D detections for each view (a, images from *chi3d* [9]). Then all possible pairs between the views are created. In this case this leads to six pairs from the two detections above with the three below. The filter step (2) with previous poses is skipped here. 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.

3.2. Method Comparison

There exist several differences to the other algorithmic approaches. In comparison to *mvpose* [8] no person identification by appearance similarities is required, the association is done only by geometric matching. *mv3dpose* [19] matches and groups 2D poses by epipolar geometry instead of the 3D triangulation and grouping used here. *PartAware-Pose* [7] 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* [25] 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* [26] 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.* [5] 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

Method	Time (μs)
Pose undistortion	14
Pair creation (1)	4
Pair filtering (2)	10
Triangulate and Score (3-7)	26
Grouping (9)	4
Triangulate and Score (10)	27
Merge (11)	12
Post-process (12)	8
Total	108

Table 1. Average time for different steps in microseconds.

OpenCV's SVD-based triangulation, a simple mid-point triangulation was implemented, which was much faster and almost as accurate in the experiments. 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.

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, here a single core of an *AMD-7900X* was used.

Method	Time (ms)
<i>mvpose</i> [8]	105
<i>VoxelPose</i>	103
<i>mv3dpose</i>	63
<i>VoxelKeypointFusion</i> [2]	48
Faster-VoxelPose	38
<i>4DAssociation</i> [26]	32
<i>PartAwarePose</i> [7]	10
<i>Bridgeman et al.</i> [4]	9.1
<i>Chen et al.</i> [5]	3.1
<i>QuickPose</i> [26]	2.9
<i>RapidPoseTriangulation</i>	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* [10], *shelf* [1], *campus* [1], *mvor* [17], and *panoptic* [13], similar as in other works. Additionally, the datasets *chi3d* [9], *tsinghua* [24] and *egohumans* [14] 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 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.

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
VoxelKeypointFusion	96.9	81.7	100	64.3	95.0	100	0

RapidPoseTriangulation	96.8	88.2	99.9	60.5	94.5	100	0	100	112
------------------------	------	-------------	------	-------------	------	------------	---	------------	------------

Table 3. Transfer to *human36m* [10], 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
Faster-VoxelPose (synthetic)	99.0	87.9	100	58.8	99.0	100	48.8
MvP (synthetic)	98.6	94.1	99.7	51.8	97.1	100	82.2
mv3dpose	97.1	91.4	98.4	55.8	94.8	98.5	44.3
VoxelKeypointFusion	98.8	93.3	100	51.3	98.3	100	49.1

RapidPoseTriangulation	99.2	94.3	100	47.6	98.7	100	42.3	73.2	63.5
------------------------	-------------	-------------	------------	-------------	------	------------	-------------	-------------	-------------

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
mvpose	91.3	70.2	99.9	80.4	82.7	100	25.4
mv3dpose	84.1	64.4	93.4	135	62.5	94.9	10.1
PartAwarePose	93.2	78.4	98.9	74.7	93.9	98.9	22.7
VoxelKeypointFusion	91.1	74.6	99.9	84.4	80.3	100	35.5

RapidPoseTriangulation	95.0	79.9	100	75.6	92.3	100	15.7	91.5	139
------------------------	-------------	-------------	------------	------	------	------------	------	------	------------

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
Faster-VoxelPose (synthetic) [2]	37.9	28.1	45.5	109	29.4	46.1	7.7
VoxelKeypointFusion [2]	54.5	43.9	75.1	128	35.9	76.6	24.2

RapidPoseTriangulation	58.0	43.8	75.5	120	39.3	76.7	27.5	74.6	118
------------------------	-------------	------	-------------	-----	-------------	-------------	------	------	------------

Table 6. Transfer to *mvor* [17] 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
PRGnet	99.5	99.1	99.9	17.1	99.9	99.9	2.0
TEMPO	98.1	97.4	98.5	16.8	98.4	98.4	2.4

VoxelKeypointFusion	97.1	94.0	99.7	47.8	97.3	99.9	2.4	98.7	4.2
---------------------	------	------	-------------	------	------	-------------	-----	-------------	-----

RapidPoseTriangulation	99.0	96.7	99.7	30.5	99.3	99.7	2.5	98.6	55.7
------------------------	-------------	-------------	-------------	-------------	-------------	------	-----	------	-------------

Table 7. Replication of *panoptic* [13] results and transfer without depth, all other results from [2].

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.

A lot 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* 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.

A lot of 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 not always can be detected as 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.

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.

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
Faster-VoxelPose (synthetic)	97.3	88.2	99.2	69.3	94.9	99.6	0.8
PRGnet	95.8	82.9	99.1	80.1	85.0	100	10.2
TEMPO	89.0	81.1	90.7	66.5	86.4	91.0	6.6
SelfPose3d	79.7	71.5	87.2	97.8	66.5	89.7	79.8
mvpose	98.5	85.2	100	66.7	98.3	100	0.5
mv3dpose	58.9	53.8	61.1	71.8	55.9	61.4	2.2
PartAwarePose	89.1	81.6	92.9	77.8	84.0	94.0	8.7
VoxelKeypointFusion	94.8	81.7	99.1	68.5	95.5	99.4	0.1
RapidPoseTriangulation	98.7	90.2	99.5	59.8	98.7	99.6	0
RapidPoseTriangulation	98.7	90.2	99.5	59.8	98.7	99.6	94.6

Table 8. Transfer to *chi3d* [9]

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
Faster-VoxelPose (synthetic)	94.8	87.9	98.6	66.3	89.7	99.0	9.2
PRGnet	96.8	88.7	99.9	67.8	93.2	100	10.5
TEMPO	89.2	85.7	90.4	57.3	89.4	90.3	1.4
SelfPose3d	87.2	83.1	89.2	65.9	83.0	90.2	51.2
mvpose	91.4	81.5	99.3	79.2	88.6	99.7	4.7
mv3dpose	68.6	64.8	71.0	63.3	66.8	71.3	14.6
PartAwarePose	89.9	81.7	94.3	79.7	78.6	94.9	6.7
VoxelKeypointFusion	98.0	95.9	99.5	51.1	95.1	99.8	0.9
RapidPoseTriangulation	98.7	96.0	99.8	52.8	95.7	100	0.5
RapidPoseTriangulation	98.7	96.0	99.8	52.8	95.7	100	46.9

Table 9. Transfer to *tsinghua* [24]

The first one is the *chi3d* [9] 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 the one of *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 a few cases in which the 2D pose estimation struggled to correctly detect the joints between very close person, the 3D triangulation did not get enough inputs to correctly triangulate the joints and could miss a person. One could reduce the reprojection matching threshold to better handle such problematic 2D detections, but this would also increase the number of false positives. The authors of *CloseMoCap* [16] also reported very good transfer results on this dataset, reaching a *PCK@50* of 94.3 on average. That would be better than any *PCK@100* score reached here. They also evaluated *VoxelPose*, *Faster-VoxelPose* and *mvpose* on the same dataset and reached much better results for them as well. Besides that, 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, it though can be seen that the hips are

positioned higher on the body than in the *COCO* skeleton definition. So in conclusion the mismatch is likely caused by parts of the evaluation somewhere in *CloseMoCap*, but even though it was promised in its paper, the source-code was not published, so this assumption can not be verified.

The second dataset is *tsinghua* [24], 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 well on this dataset.

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.

4.3. Using more cameras

From the previous results, it can be seen that the number of cameras has a large impact on the performance of the

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
Faster-VoxelPose (synthetic)	81.0	68.1	86.1	82.6	71.3	86.6	11.5
PRGnet	63.5	52.4	68.4	134	51.3	69.3	28.8
TEMPO	69.1	59.9	72.2	79.2	61.9	72.4	21.2
SelfPose3d	68.0	55.6	77.4	115	52.4	79.0	68.3
mvpose	83.4	70.5	90.0	81.6	76.8	90.4	19.6
mv3dpose	61.0	53.9	64.4	110	54.6	65.0	21.5
PartAwarePose	79.1	71.0	83.9	91.5	74.0	84.6	17.7
VoxelKeypointFusion	89.0	78.5	95.6	74.6	83.4	96.0	18.3
RapidPoseTriangulation	91.1	82.1	95.8	69.4	86.5	96.0	14.3
RapidPoseTriangulation	91.1	82.1	95.8	69.4	86.5	96.0	89.8
RapidPoseTriangulation	91.1	82.1	95.8	69.4	86.5	96.0	95.7

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
legoassemle	VoxelPose (synthetic)	89.9	74.4	93.8	94.7	65.3	96.1	71.2
	Faster-VoxelPose (synthetic)	75.4	64.1	94.2	107	54.0	95.3	0
	VoxelKeypointFusion	99.4	97.2	99.4	37.0	99.4	99.7	0
	RapidPoseTriangulation	100	98.9	100	25.3	100	100	0
tennis	VoxelPose (synthetic)	66.9	29.5	86.5	128	3.9	87.0	1.3
	VoxelKeypointFusion	98.2	90.6	100	64.6	98.1	100	0
	RapidPoseTriangulation	99.9	99.5	100	22.9	99.7	100	0
tagging	VoxelKeypointFusion	89.1	84.2	92.4	67.1	84.3	94.3	8.0
	RapidPoseTriangulation	94.6	90.3	96.2	44.6	90.8	96.5	13.1
fencing	VoxelKeypointFusion	100	97.7	100	37.5	100	100	69.0
	RapidPoseTriangulation	100	99.6	100	24.0	100	100	50.0
basketball	VoxelKeypointFusion	96.3	92.2	97.6	53.5	94.8	98.2	50.9
	RapidPoseTriangulation	99.7	96.7	100	40.7	99.7	100	27.0
volleyball	VoxelKeypointFusion	97.1	94.1	98.0	48.5	96.1	98.6	25.2
	RapidPoseTriangulation	99.1	98.2	99.2	29.1	99.0	99.2	1.2
badminton	VoxelKeypointFusion	99.8	99.0	100	36.5	100	100	16.2
	RapidPoseTriangulation	99.9	99.5	100	24.7	100	100	0.0

Table 11. Transfer to *egohumans* [14]

algorithms. Especially *mvor* shows that using only three cameras while there are 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* [14] 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 $25 \times 14\text{ m}$ 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 distorted stronger 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 *legoassemle* 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 *mpose* the number of views was too high, the six cameras from *tsinghua* already took up the complete GPU memory. *TEMPO* [6] 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* shows a very good

performance, which is only surpassed by *RapidPoseTriangulation*.

4.4. About real-time performance

Regarding the real-time performance with many cameras, the results show, that while the 3D part of *RapidPoseTriangulation* is very fast and at most takes 1 ms, the majority of the time is currently spent for 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, it was counterweighted by an increased transfer time for the full image.

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	4.9
RapidPoseTriangulation	99.7	88.3	100	45.7 — 29.2 — 54.1 — 41.9	100	81.2

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 on the other hand take 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 require less potent hardware, a *Raspberry Pi* with the *AI-Hat* could already be enough. 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.

To evaluate the performance, the *h3wb* dataset [27] 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 pushes some keypoints closer together than they really are. This was especially visible at the finger keypoints, which were often clustered into agglomerations, instead of showing the clear lines of the fingers. *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* merged 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 be only seen in very few views, like 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

In this work a new algorithm for 3D human pose triangulation under the name of *RapidPoseTriangulation* was presented. It is based on the idea of triangulating pairs of 2D poses from different views separately and grouping and merging them to the final poses directly in 3D space. While the general approach is relatively simple, it is also much faster than any previous algorithm. Unlike most other methods, it is also able to predict whole-body poses.

In a broad evaluation of the algorithm on several datasets it showed state-of-the-art generalization 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 most modern fully differentiable methods in the task of exploiting multi-view information.

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. 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. [2](#) [4](#) [5](#) [8](#)
- [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] 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](#)
- [6] 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](#)
- [7] 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](#)
- [8] 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](#)
- [9] 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](#)
- [10] 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](#)
- [11] 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](#)
- [12] 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](#)
- [13] 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](#)
- [14] 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](#)
- [15] N Dinesh Reddy, Laurent Guigues, Leonid Pishchulin, Jayan Eledath, and Srinivasa G Narasimhan. Tesseltrack: 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](#)
- [16] 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](#)
- [17] 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](#)
- [18] 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](#)
- [19] 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](#)
- [20] 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](#)
- [21] 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](#)
- [22] 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](#)
- [23] Hang Ye, Wentao Zhu, Chunyu Wang, Ruijie Wu, and Yizhou Wang. Faster VoxelPose: Real-time 3D Human Pose Estimation by Orthographic Projection. In *European Conference on Computer Vision (ECCV)*, 2022. [2](#)
- [24] 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](#)
- [25] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1324–1333, 2020. [4](#)

- [26] 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
- [27] 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