Toward User-specific Tracking by Detection of Human Shapes in Multi-Cameras Supplementary Material

Chun-Hao Huang[1], Edmond Boyer[2], Bibiana do Canto Angonese[1], Nassir Navab[1], Slobodan Ilic[3]

[1] Technische Universität München, [2] LJK-INRIA Grenoble Rhône-Alpes, [3] Siemens AG

{huangc,slobodan.ilic,bibiana.canto,navab}@in.tum.de, edmond.boyer@inria.fr

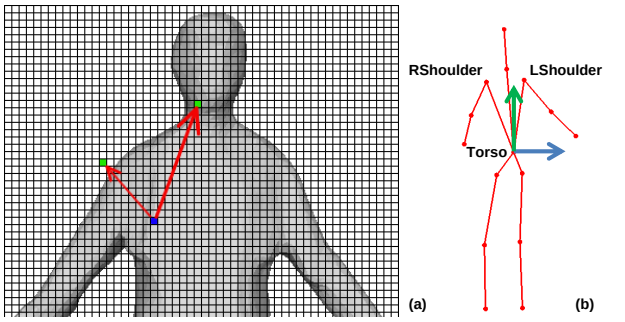

Figure 1. (a) illustrations of the offset pair for features $\mathbf{f}$. (b) illustrations of two reference vectors from the skeleton.

The supplementary material for the paper *Toward User-specific Tracking by Detection of Human Shapes in Multi-Cameras* consists of this document and the accompanying video. It provides details of feature vectors $\mathbf{f}$, as well as the way of re-orienting meshes.

1. Details of volumetric features f

In this section, we explain how to construct our volumetric features $\mathbf{f}$. As depicted in Fig. 1(a), $\mathbf{f}$ operates on the normal field $\mathbf{N}$, characterizing the local geometries of the current voxel $\mathbf{v}$ (blue).[1] Let κ denotes the index of feature channel. The first 6 channels $(f_0, \ldots, f_5)$ involve two neighboring voxels (green), while the last two dimensions (f_6, f_7) involve only the current voxel. Neighboring voxels are selected based on a pair of offsets $\psi = (\mathbf{o}_1, \mathbf{o}_2) \in \Omega_3 \times \Omega_3$ (red vectors), either aligned with a local coordinate frame or not. Same as in [2], each offset has 50% probability to be $\mathbf{0}$, where we encode the geometry of the current voxel $\mathbf{v}$ itself.

Although the length of $\mathbf{f}$ is only 8, during training in practice there are $8 \times N_\psi$ possibilities to try out, which is approximately 120k in our experiments. In each branch node, the offset pair ψ and the feature channel κ that maximizes the information gain (Eq. 3 in the main paper) are saved. During testing, one does not have to prepare the whole one-hundred-thousand-dimensional vector to traverse the forest, since the calculation of each dimension is independent.

[1] Since we intend to describe surface geometries, current voxels $\mathbf{v}$ are always surface voxels $\mathbf{v}_{\text{suf}}$, as in Sec. 4.2 of the main paper. In this supplementary document, however, we drop the subscript in order to keep notations uncluttered.

$\mathbf{v}_2$ \ $\mathbf{v}_1$	$\mathbf{v}_{\text{out}}$	$\mathbf{v}_{\text{suf}}$	$\mathbf{v}_{\text{in}}$
$\mathbf{v}_{\text{out}}$	$\lvert\mathbf{n}\rvert_* + 8$	$\mathbf{n}_1 \cdot \mathbf{n} + 2$	$\lvert\mathbf{n}\rvert_* - 4$
$\mathbf{v}_{\text{suf}}$	$\mathbf{n} \cdot \mathbf{n}_2 + 6$	$\mathbf{n}_1 \cdot \mathbf{n}_2$	$\mathbf{n} \cdot \mathbf{n}_2 - 6$
$\mathbf{v}_{\text{in}}$	$\lvert\mathbf{n}\rvert_* + 4$	$\mathbf{n}_1 \cdot \mathbf{n} - 2$	$\lvert\mathbf{n}\rvert_* - 8$

Table 1. The response table of f_0. $\mathbf{n} = \mathbf{N}(\mathbf{v})$, $\mathbf{n}_1 = \mathbf{N}(\mathbf{v}_1)$ and $\mathbf{n}_2 = \mathbf{N}(\mathbf{v}_2)$. $\lvert\mathbf{n}\rvert_*$ represents a randomly chosen dimension of $\mathbf{n}$.

1.1. Extended dot product of normals

Let $\mathbf{v}_1$ and $\mathbf{v}_2$ denote the two neighboring voxels. We first consider the second-order information between them, which can be roughly measured by the dot product of normals: $\mathbf{N}(\mathbf{v}_1) \cdot \mathbf{N}(\mathbf{v}_2) \in [-1, 1]$. Nevertheless, this operation is only valid when both neighboring voxels lie on the surface, which is not always the case. Moreover, it is known that higher order information is less reliable in 3D data. We therefore include the zeroth-order information (surface occupancies), and extend the dot product to a more general response table as in Table 1.

Let $\mathbf{n}$, $\mathbf{n}_1$, and $\mathbf{n}_2$ denotes the normals of the current voxel $\mathbf{v}$, $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. Whenever one of the neighboring voxels is not a surface voxel, we replace its normal with the one from the current voxel, $\mathbf{n}$, and perform dot product. If neither $\mathbf{v}_1$ nor $\mathbf{v}_2$ lies on the surface, we consider a randomly chosen dimension of the $\mathbf{n}$ as results. Afterwards, we add a scalar constant to the result, distinguishing different cases of surface occupancies. The first dimension of the feature $f_0(\mathbf{v}_1, \mathbf{v}_2)$ follows Table 1.

1.2. Difference of VNF in local cuboids

To further exploit the normal field $\mathbf{N}$, we open a cuboid around $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively, subtract two cuboids, and

sum up the results over the cuboid:

$$\mathbf{s}(\mathbf{v}_1, \mathbf{v}_2) = \sum_{\mathbf{i} \in V} \left(\mathbf{N}(\mathbf{v}_1 + \mathbf{i}) - \mathbf{N}(\mathbf{v}_2 + \mathbf{i}) \right), \qquad (1)$$

where V represents the set of neighborhood indices. In our implementation it has the size of $5 \times 5 \times 5$. Eq. 1 leads to a vector $\mathbf{s} = (s_x, s_y, s_z) \in \Omega_3$. The next five dimensions of the feature $\mathbf{f}$ can thereby be defined accordingly:

$$f_1(\mathbf{v}_1, \mathbf{v}_2) = s_x, \qquad (2a)$$
$$f_2(\mathbf{v}_1, \mathbf{v}_2) = s_y, \qquad (2b)$$
$$f_3(\mathbf{v}_1, \mathbf{v}_2) = s_z, \qquad (2c)$$
$$f_4(\mathbf{v}_1, \mathbf{v}_2) = \|\mathbf{s}\|_2, \qquad (2d)$$
$$f_5(\mathbf{v}_1, \mathbf{v}_2) = s_x + s_y + s_z. \qquad (2e)$$

Recall that we use $\pm(2, 2, 2)$ in the normal field $\mathbf{N}$ to indicate voxels outside and inside the mesh respectively. These indicators are included in Eq. 1 and 2, and hence $f_1 \dots f_5$ by definition also contain the zeroth-order information of shapes.

1.3. Normalized heights and lengths

The last two feature dimensions take only the current voxel $\mathbf{v}$ into account, *i.e.* $f_6(\mathbf{v})$ and $f_7(\mathbf{v})$. They are the normalized height (f_6) and normalized length (f_7), respectively. Let $\hat{\mathbf{v}}$ denotes the normalized voxel coordinate w.r.t. the bounding box of the mesh, and the operator $|\cdot|_*$ takes the $*$-coordinates of the vector. Typically f_6 corresponds to $|\hat{\mathbf{v}}|_z$, and f_7 corresponds to $|\hat{\mathbf{v}}|_x$. Note anyway that this could vary due to different recording settings.

2. Re-orient meshes

In this section we describe how to orient meshes into a canonical direction by using the underling skeletons. We consider two bone vectors, *Torso-LShoulder*, and *Torso-RShoulder* since they usually remain stable even when limbs have undesirable deformations. As illustrated in Fig. 1(b), the first reference vector (blue) is the cross product of two bones, while the second one (green) is the sum of them. We align the two reference vectors respectively to x-axis and z-axis, canceling the rotations of the subjects.

3. Error vs. ICP-itr.

In Table 2, we point out the advantage of our method over surICP. With the correspondences from VNF-forest framework, ICP attains similar accuracy but requires less iterations to converge. This demonstrates that, compared with using results of previous frames as initializations, our method is capable of providing better ones.

	ours + ICP		surICP [1]	
	error	# ICP-itr.	error	# ICP-itr.
Crane	8015	17	8138	20
Jumping	7976	16	7648	21
Bouncing	7569	34	7826	44
Handstand	9767	27	9963	60

Table 2. Average silhouette overlap error in pixels, and the average ICP-iterations (itr.) of 4 sequences.

References

[1] C. Cagniart, E. Boyer, and S. Ilic. Probabilistic deformable surface tracking from multiple videos. In *ECCV*. Springer, 2010.

[2] J. Shotton, A. Fitzgibbon, M. Cook, and A. Blake. Real-time Human Pose Recognition in Parts from Single Depth Images . In *CVPR*. IEEE, 2011.