Appendix

A. Visibility

In figure 6, we present qualitative examples of the visibility heatmaps produced by the query image branch as these are used to predict visibility for our model. As can be seen, the visibility prediction heads do a good job at identifying which points are visible and occluded in the target image. Specifically, in the first row, the rider in the foreground (middle) occludes the people seen in the query image (left) and this is correctly predicted by the black region in the centre of the frame (right). Similarly, in the last row, the points on the cart (left) are predicted correctly as visible (right). Conversely, the part of the scene that disappears due to the camera pan to the right is clearly classified as occluded with a low probability. It should also be noted that the visibility head correctly captures the visibility due to the camera motion. In the first, third and fourth rows, the part of the scene visible in the second image is easily distinguished in the visibility map. In the second row, only the column is common between frames and is predicted as visible.

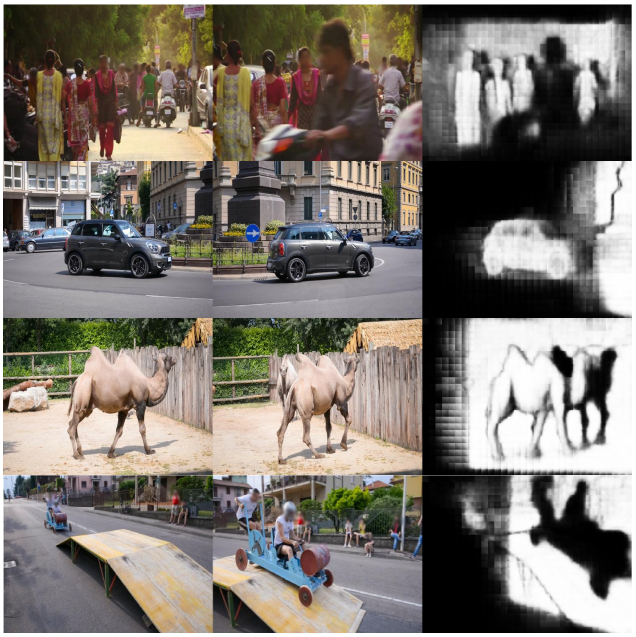

Figure 6. Visualisation of the visibility heatmaps produced by our added visibility prediction head in the query branch. Each head predicts whether each pixel is visible or not in the other branch's image. Here white is high probability and black is low.

B. Resolution and Bilinear Interpolation

In training, MASt3R and PointSt3R have multiple resolution options, with the x-axis kept constant at 512 pixels. We conduct an ablation of the affect of changing input resolution at inference on 2D tracking for TAP-Vid-DAVIS. As expected, the largest input resolution performs best and this happens to be a common native resolution for point trackers. We also report how results change between using the nearest pixel's feature to the query point and when using bilinear interpolation to query the correspondence feature. As expected bilinear interpolation produces more accurate results, with an increase of 1.1%.

Resolution	Sampling	δ_{avg} ↑
$(512, 160)$	Nearest	63.5
$(512, 256)$	Nearest	70.0
$(512, 288)$	Nearest	70.7
$(512, 336)$	Nearest	71.9
$(512, 384)$	Nearest	<u>72.7</u>
$(512, 384)$	Bilinear	**73.8**

Table 7. Tracking accuracy δ_{avg} ↑ on TAP-Vid-DAVIS [8] when changing resolution and sampling strategy. "Nearest" is simply selecting the nearest pixel's feature to the query coordinate. "Bilinear" uses bilinear interpolation to retrieve the feature for matching.

C. Percentage of Dynamic Correspondences

We include the full table of results in table 8 for the figure of the ablation of percentage of dynamic correspondences in each batch. It should be noted that our chosen model was 95% for its consistent performance across datasets and splits.

Model	Time (s)
CoTracker3 [14]	0.3
PointSt3R	1.7

Table 9. Inference time for CoTracker3 [14] and PointSt3R for a window for 16 frames and 5 query points.

	All Points					Static Points				Dynamic Points			
Model	TAP-Vid-DAVIS	RoboTAP	RGB-S	EgoPoints	PO	RoboTAP	RGB-S	EgoPoints	PO	RoboTAP	RGB-S	EgoPoints	PO
MASt3R [18]	38.5	71.6	73.2	53.5	45.6	99.2	89.3	79.4	68.8	42.5	39.4	12.3	25.2
PointSt3R - 0%	20.2	56.0	64.4	48.2	38.8	99.6	96.6	73.6	78.9	13.5	22.2	8.5	7.4
PointSt3R - 25%	68.1	76.6	83.0	58.3	55.0	99.7	99.9	78.4	79.2	60.3	52.6	24.6	33.9
PointSt3R - 50%	71.2	77.9	84.2	59.9	57.3	99.7	99.9	78.9	78.4	65.0	55.1	27.8	38.6
PointSt3R - 75%	72.9	77.8	85.9	60.4	57.6	99.7	99.7	78.5	75.2	67.5	58.5	29.3	41.7
PointSt3R - 80%	73.1	77.9	85.9	60.4	57.7	99.8	99.8	78.4	74.9	67.8	58.5	29.5	42.2
PointSt3R - 85%	73.3	78.2	86.4	60.8	57.3	99.6	99.7	78.7	73.6	68.1	59.8	30.3	42.7
PointSt3R - 90%	73.5	78.4	86.8	61.3	57.2	99.5	99.7	78.7	72.4	68.6	60.9	31.6	43.4
PointSt3R - 95%	73.8	78.6	87.0	61.3	56.2	99.4	99.2	78.7	70.2	69.4	61.6	31.4	43.4
PointSt3R - 99%	73.9	78.6	86.3	61.3	53.9	99.5	97.0	78.6	65.1	69.2	62.2	31.7	43.7
PointSt3R - 100%	73.8	76.2	75.0	57.7	31.4	86.1	68.4	71.9	13.7	69.8	62.3	32.3	43.8

Table 8. Tracking accuracy δ_{avg} ↑ for PointSt3R models with different percentage of dynamic correspondences per batch, compared with the MASt3R baseline.

D. Computational Cost

PointSt3R uses global cosine similarity and produces results in a pairwise fashion. It is not particularly optimised. We compare the runtime to CoTracker3 [14] which uses a window of 16 frames, so we chose to feed PointSt3R a batch of 16 image pairs and performed cosine similarity to produce tracks for 5 query points and then compared the runtime. As can be seen from table 9, PointSt3R is 5x slower than CoTracker3, which is expected due to the pairwise method used.

We highlight that our attempt is not to produce a competitive point tracker in performance, but to evaluate the capabilities of current 3D grounding foundation models for the task of point tracking. Our results also highlight that the temporal context can be removed from point trackers without impacting the performance much.

E. 3D Tracking - Implementation Details

For the 3D tracking results seen in the main paper, we employ ZoeDepth [2] NK model. When the ground truth intrinsics are not used, we use the demo.py, available with the SpaTracker [41] published code, that estimates the intrinsics for videos without ground truth by setting $f_x = f_y = W$ and $c_x = W/2, c_y = H/2$.