Supplement to Post-Render Warp with Late Input Sampling Improves Aiming Under High Latency Conditions

JOOHWAN KIM, PYARELAL KNOWLES, JOSEF SPJUT, BEN BOUDAOUD, and MORGAN MCGUIRE, NVIDIA

ACM Reference Format:
Joohwan Kim, Pyarelal Knowles, Josef Spjut, Ben Boudaoud, and Morgan McGuire. 2020. Supplement to Post-Render Warp with Late Input Sampling Improves Aiming Under High Latency Conditions. In *HPG '20: High Performance Graphics*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SOFTWARE MODIFICATIONS

We modified the open source FPSci project [1–3] to enable us to run the experiment described in the main document. This section describes the functional changes we made and makes some comparisons to a real world implementation. Additionally, we also include as supplemental material C++ source code for the most important changes to reproduce our work. Furthermore, the full source code needed to repeat the study is available in the `hpg2020` tag of the FPSci github repository [3].

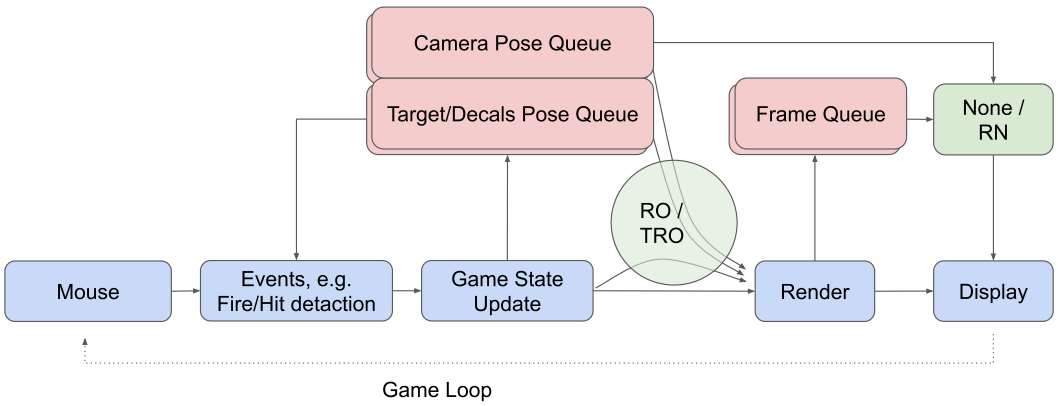

Fig. 1. Modifications to the game loop to support warp implementations. Red boxes are added queues to buffer state and frames when needed. Green blocks represent the added functionality to modify the rendering and displayed frame based on chosen warp condition.

A primary requirement of this experiment is to simulate latency and then short circuit certain parts to warp or simulate warping the rendered images. Fig. 1 shows the addition of various queues to introduce artificial latency. For RN and the **105ms** delay cases the frame queue is used, which records entire framebuffers and displays them after a fixed number of frames that add 80ms. In the case of RN, the displayed frame is warped by drawing a texture mapped quad and transforming it by the difference between the current camera and equivalent pose for the old frame. We consider this a safe lower bound on what might be done since this naive implementation filled empty areas black

HPG '20, July 2020, Online

and allowed the not-so-noticeable image resampling artifacts to remain. The oracle cases, RO and TRO, bound the improvement from compensating for view rotation and translation respectively. Rather than delay frames, the game state is stored in a queue - also 80ms long - and delayed before rendering. For RO, recent camera rotation is used instead of the delayed one, and similarly recent translation is used as well for TRO. Care was taken to make sure effects such as decals and target explosions were also delayed in the game state queue.

In order for any warp implementation to function correctly, it is essential for the content that should be warped to be separated from the content that should remain in-place on the screen. For example, heads up display (HUD) elements are 2D content placed at specific location on the screen that should not move when the 3D geometry is rotated to the new viewer orientation. Furthermore some 3D content, like the weapon model for the first person viewer, would move along with the rotation. Thus effective warp implementations depend on separating one *3D layer* that will be warped and a *2D layer* that should be presented in-place without any warp applied. For our implementation, the HUD and 2D elements were drawn separately using game state queues. In a true cloud implementation, the renderer on the cloud side would need to change the video encoding to allow for separate layers, and the thin client would need to change its decoder to receive both layers, warp only the 3D layer based on the latest player input, and composite them. While this requirement is somewhat invasive to some game rendering implementations, we believe it to not be overly burdensome. Similar separation of the 2D and 3D content is already handled in some engines and techniques, such as DLSS. It is worth noting that a warp implementing both rotation and translation correction would be able to use the frame-to-frame motion vectors or a depth map to aid in proper image warping.

Given that our implementation only emulates network latency by adding a fixed local delay, it's important to recognize the way in which we adjusted hit detection in our fire events in-game. In all cases, the combined state of camera and game simulation that matched the displayed frame when a click event occurred is used. Thus in cases where the game state was delayed by 80 ms for the displayed frame, that delayed game state was used for hit detection. This is equivalent rollback in multiplayer games. Fig. 1 shows the game state queue, which holds delayed target transforms, being reused for hit detection. In a hypothetical cloud implementation, this would imply either that the local state that was generated for the warped frame needs to be sent to the cloud (either cloud renderer or game server) to perform hit detection, or the thin client could perform this hit detection and merely send the results for validation. While the specific details of such implementations are beyond the scope of this paper, the concept of including the thin client as a contributor to game event decision making is an important contribution of this work.

2 ADDITIONAL DATA

We include in this supplement some additional user data and analysis that, while interesting, is not essential to the conclusions of the main document. We also provide as a supplement the raw data used to perform all of these anlyses in `latewarp.pkl` that can be loaded using the pandas module for Python. The analysis in the paper was based on per-user median task completion time taken from a number of task samples. These samples could be considered independently, and we include Figures 2, 4, and 3 as histograms of those samples.

2.1 Per-User Medians and Means

Since our analysis was based on the per-user median task completion times, we provide those individual medians in this supplement. We also provide means and standard deviations along side the medians in Tables 1-4.

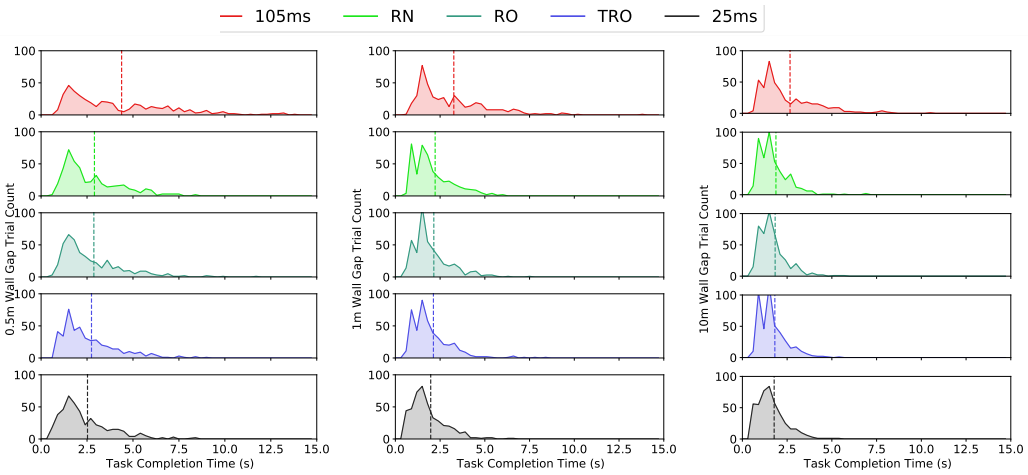

Fig. 2. Aggregate user performance for various wall gap, with distribution means shown as dashed lines.

2.2 Task Completion Time Distributional Shape

All task completion time distributions have an (expected) Poisson shape, with similar modes, and means largely influenced by the spread/heavy sided-ness of the samples. For this reason the spread of data tends to be better demonstrated by the means.

2.3 Analysis of User 3

User 3 was an interesting anomaly in our collected data. Unlike all other subjects, this user adapted their task strategy late into the experiment. This effectively resulted in a prolonged training effect for this user, who happened to complete the case with 105 ms of latency last.

As conditions progressed, User 3 chose to minimize mouse motion, and maximize keyboard motion to track the target. This sort of mixed-peripheral aiming strategy is a topic considered for future work.

Interestingly, this subject also substantially outperformed most other users for many wall gaps and warp conditions, particularly in the 105 ms of latency (no warp) case. This may imply that more skilled users require longer training/adaptation times to fully adapt to new experiment conditions.

REFERENCES

[1] Josef Spjut, Ben Boudaoud, Kamran Binaee, Jonghyun Kim, Alexander Majercik, Morgan McGuire, David Luebke, and Joohwan Kim. 2019. Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz. In *SIGGRAPH Asia 2019 Technical Briefs (SA '19)*. Association for Computing Machinery, New York, NY, USA, 110–113. https://doi.org/10.1145/3355088.3365170

[2] Josef Spjut, Ben Boudaoud, Kamran Binaee, Alexander Majercik, Morgan McGuire, and Joohwan Kim. 2019. FirstPersonScience: Quantifying Psychophysics for First Person Shooter Tasks. In *UCI Esports Conference*.

[3] Josef Spjut, Ben Boudaoud, Pyarelal Knowles, Zander Majercik, Morgan McGuire, and Joohwan Kim. 2019. FirstPersonScience. https://github.com/NVlabs/abstract-fps

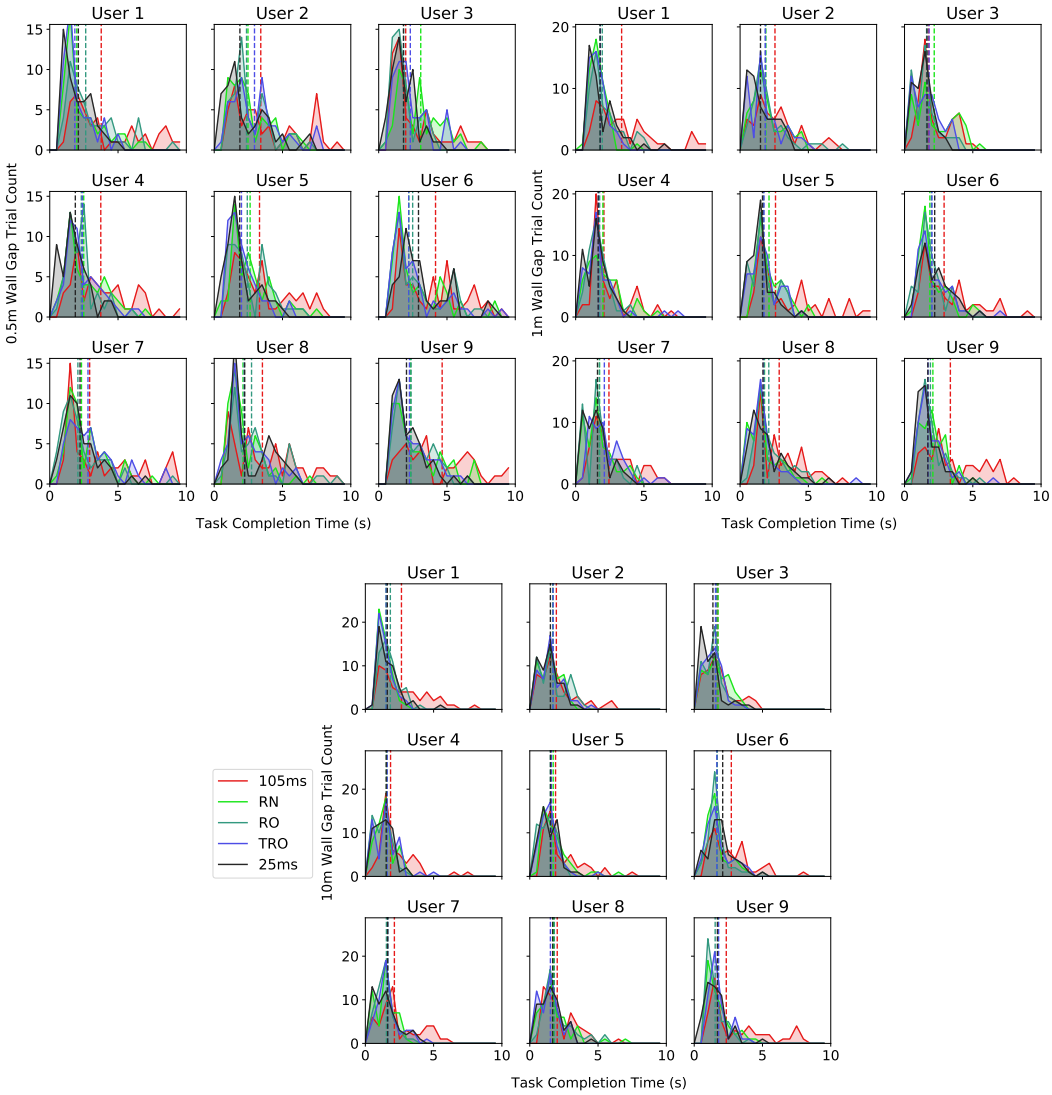

Fig. 3. Task completion time shown per user. Medians are shown as dashed vertical lines.

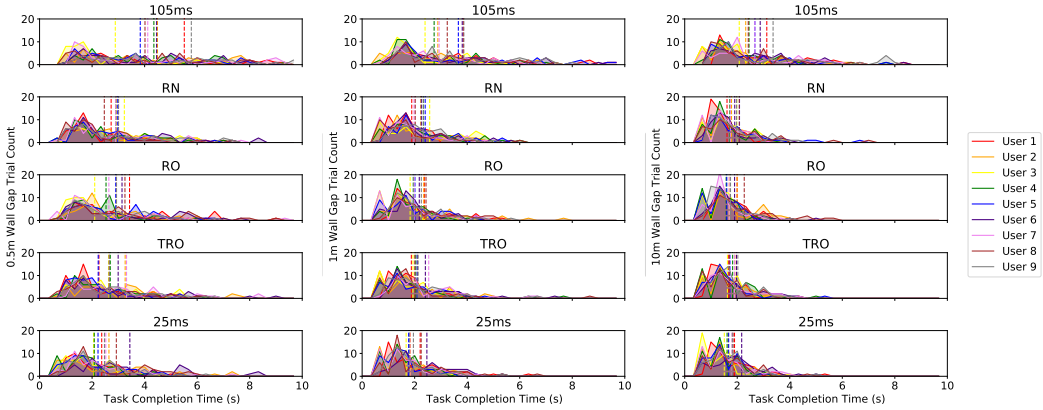

Fig. 4. Performance broken out per condition and wall gap across various users, with distribution means shown as dashed vertical lines.

Table 1. Per user task completion time (s) statistics by condition (per user-condition N = 150 trials).

Subject	Condition	Mean (μ)	Median	Std. Dev. (σ)
User 1	105ms	4.164	3.310	2.805
	RN	2.075	1.790	1.108
	RO	2.597	2.061	1.580
	TRO	1.931	1.725	0.851
	25ms	2.157	1.766	1.052
User 2	105ms	3.231	2.488	2.269
	RN	2.310	1.980	1.239
	RO	2.423	2.020	1.352
	TRO	2.427	1.978	1.411
	25ms	2.048	1.625	1.285
User 3	105ms	2.452	1.806	1.432
	RN	2.594	2.225	1.393
	RO	1.875	1.691	0.833
	TRO	2.063	1.797	1.060
	25ms	1.747	1.633	0.804
User 4	105ms	3.180	2.381	2.075
	RN	2.317	1.938	1.265
	RO	2.019	1.831	0.911
	TRO	2.186	1.781	1.217
	25ms	1.831	1.659	0.806
User 5	105ms	3.389	2.536	2.142
	RN	2.488	2.143	1.319
	RO	2.171	1.805	1.173
	TRO	2.017	1.724	0.952
	25ms	1.891	1.659	0.926
User 6	105ms	3.714	2.915	2.376
	RN	2.388	1.855	1.385
	RO	2.405	1.895	1.395
	TRO	2.457	1.946	1.469
	25ms	2.691	2.316	1.446
User 7	105ms	3.240	2.348	2.096
	RN	2.204	1.814	1.218
	RO	2.105	1.740	1.216
	TRO	2.579	2.111	1.491
	25ms	2.048	1.821	1.093
User 8	105ms	3.225	2.684	1.880
	RN	2.188	1.855	1.223
	RO	2.650	2.184	1.529
	TRO	2.119	1.740	1.214
	25ms	2.337	1.871	1.305
User 9	105ms	4.338	3.433	3.076
	RN	2.437	2.053	1.329
	RO	2.309	1.855	1.286
	TRO	2.269	1.855	1.096
	25ms	2.079	1.863	1.005

Table 2. Per user task completion time (s) statistics by wall separation (per user, condition, and separation N = 50 trials).

Subject	Cond	0.5m		1m		10m	
		Med	$\mu \pm \sigma$	Med	$\mu \pm \sigma$	Med	$\mu \pm \sigma$
User 1	105ms	3.770	5.510 ± 3.571	3.367	3.858 ± 2.230	2.645	3.125 ± 1.701
	RN	1.978	2.725 ± 1.507	1.800	1.888 ± 0.695	1.527	1.611 ± 0.504
	RO	2.633	3.426 ± 2.193	1.949	2.370 ± 1.075	1.832	1.996 ± 0.652
	TRO	1.840	2.261 ± 1.106	1.807	1.873 ± 0.639	1.519	1.659 ± 0.596
	25ms	2.094	2.366 ± 1.045	1.790	2.209 ± 1.152	1.610	1.896 ± 0.888
User 2	105ms	3.409	4.489 ± 2.910	2.555	2.882 ± 1.607	1.946	2.323 ± 1.365
	RN	2.471	2.915 ± 1.505	1.885	2.233 ± 1.102	1.679	1.782 ± 0.689
	RO	2.365	2.926 ± 1.448	1.865	2.346 ± 1.423	1.692	1.996 ± 0.961
	TRO	2.949	3.259 ± 1.703	1.840	2.114 ± 1.155	1.708	1.907 ± 0.821
	25ms	1.880	2.642 ± 1.744	1.494	1.840 ± 0.914	1.510	1.663 ± 0.731
User 3	105ms	1.978	2.883 ± 1.805	1.773	2.396 ± 1.184	1.715	2.076 ± 1.078
	RN	3.079	3.227 ± 1.545	2.175	2.572 ± 1.409	1.747	1.984 ± 0.819
	RO	1.813	2.100 ± 0.926	1.616	1.828 ± 0.863	1.641	1.696 ± 0.629
	TRO	2.314	2.627 ± 1.299	1.797	1.935 ± 0.885	1.559	1.627 ± 0.613
	25ms	1.821	2.058 ± 0.905	1.649	1.675 ± 0.633	1.370	1.507 ± 0.748
User 4	105ms	3.737	4.352 ± 2.616	2.071	2.741 ± 1.461	1.847	2.448 ± 1.356
	RN	2.488	2.902 ± 1.470	1.995	2.333 ± 1.220	1.552	1.716 ± 0.667
	RO	2.308	2.530 ± 1.058	1.757	1.947 ± 0.776	1.592	1.579 ± 0.555
	TRO	2.389	2.691 ± 1.269	1.675	2.043 ± 1.303	1.593	1.825 ± 0.856
	25ms	1.873	2.082 ± 0.991	1.651	1.797 ± 0.723	1.528	1.614 ± 0.575
User 5	105ms	3.317	3.834 ± 2.064	2.595	3.663 ± 2.329	1.889	2.672 ± 1.816
	RN	2.627	2.977 ± 1.509	2.119	2.401 ± 1.082	1.716	2.086 ± 1.166
	RO	2.421	2.901 ± 1.439	1.716	2.010 ± 0.936	1.510	1.602 ± 0.545
	TRO	1.995	2.232 ± 1.040	1.749	2.097 ± 0.956	1.585	1.721 ± 0.762
	25ms	1.864	2.226 ± 1.173	1.641	1.773 ± 0.799	1.518	1.674 ± 0.619
User 6	105ms	4.162	4.457 ± 2.568	2.899	3.812 ± 2.598	2.718	2.874 ± 1.524
	RN	2.200	3.005 ± 1.820	1.855	2.253 ± 1.088	1.675	1.905 ± 0.795
	RO	2.487	3.139 ± 1.867	1.970	2.176 ± 0.903	1.691	1.899 ± 0.833
	TRO	2.200	2.992 ± 1.886	2.011	2.407 ± 1.255	1.658	1.971 ± 0.901
	25ms	2.915	3.438 ± 1.849	2.209	2.465 ± 1.052	2.086	2.170 ± 0.931
User 7	105ms	2.923	4.119 ± 2.728	2.439	2.931 ± 1.530	2.127	2.671 ± 1.486
	RN	2.323	2.897 ± 1.527	1.748	1.978 ± 0.993	1.633	1.737 ± 0.618
	RO	2.061	2.638 ± 1.597	1.740	1.980 ± 1.008	1.543	1.695 ± 0.633
	TRO	2.800	3.311 ± 1.788	2.101	2.535 ± 1.294	1.643	1.892 ± 0.889
	25ms	2.217	2.521 ± 1.317	1.599	1.823 ± 0.853	1.632	1.801 ± 0.885
User 8	105ms	3.529	4.014 ± 2.287	2.864	3.241 ± 1.605	2.011	2.419 ± 1.236
	RN	2.109	2.465 ± 1.293	1.789	2.021 ± 1.101	1.732	2.078 ± 1.220
	RO	2.733	3.251 ± 1.943	2.102	2.432 ± 1.149	1.798	2.267 ± 1.167
	TRO	2.200	2.655 ± 1.370	1.740	2.005 ± 1.239	1.510	1.699 ± 0.727
	25ms	2.232	2.923 ± 1.506	1.674	2.248 ± 1.276	1.675	1.841 ± 0.785
User 9	105ms	4.646	5.778 ± 4.071	3.357	3.873 ± 1.955	2.348	3.365 ± 2.182
	RN	2.331	3.007 ± 1.744	2.077	2.328 ± 0.935	1.756	1.975 ± 0.913
	RO	2.372	2.930 ± 1.683	1.896	2.253 ± 0.998	1.534	1.743 ± 0.649
	TRO	2.251	2.654 ± 1.314	1.724	2.131 ± 1.057	1.790	2.023 ± 0.729
	25ms	2.035	2.472 ± 1.177	1.708	1.952 ± 0.882	1.699	1.813 ± 0.791

Table 3. Per user task completion time (s) statistics by weapon type (per user, condition, and weapon N = 75 trials).

Subject	Cond	Discrete		Continuous	
		Med	$\mu \pm \sigma$	Med	$\mu \pm \sigma$
User 1	105ms	2.777	3.634 ± 2.673	3.795	4.695 ± 2.833
	RN	1.379	1.683 ± 1.009	2.118	2.466 ± 1.063
	RO	1.776	2.177 ± 1.057	2.348	3.017 ± 1.877
	TRO	1.331	1.578 ± 0.743	2.020	2.285 ± 0.804
	25ms	1.330	1.760 ± 0.980	2.398	2.554 ± 0.970
User 2	105ms	1.759	2.532 ± 2.123	3.334	3.931 ± 2.193
	RN	1.463	1.723 ± 0.906	2.496	2.897 ± 1.249
	RO	1.643	1.981 ± 1.220	2.463	2.865 ± 1.334
	TRO	1.527	1.817 ± 1.045	2.611	3.036 ± 1.464
	25ms	1.084	1.494 ± 1.072	2.200	2.603 ± 1.240
User 3	105ms	1.559	1.850 ± 1.102	2.610	3.053 ± 1.471
	RN	1.675	2.111 ± 1.335	2.902	3.078 ± 1.276
	RO	1.346	1.451 ± 0.633	1.986	2.299 ± 0.793
	TRO	1.362	1.661 ± 0.906	2.216	2.465 ± 1.051
	25ms	1.050	1.310 ± 0.603	1.904	2.183 ± 0.740
User 4	105ms	1.954	2.821 ± 2.148	3.038	3.540 ± 1.935
	RN	1.551	2.082 ± 1.311	2.184	2.552 ± 1.170
	RO	1.577	1.770 ± 0.933	1.970	2.267 ± 0.817
	TRO	1.577	2.062 ± 1.422	1.987	2.311 ± 0.955
	25ms	1.478	1.467 ± 0.667	2.004	2.195 ± 0.767
User 5	105ms	2.118	3.116 ± 2.290	3.136	3.663 ± 1.946
	RN	1.757	2.232 ± 1.360	2.348	2.744 ± 1.224
	RO	1.494	1.829 ± 1.166	2.134	2.513 ± 1.075
	TRO	1.510	1.753 ± 0.990	2.119	2.281 ± 0.833
	25ms	1.232	1.511 ± 0.715	1.970	2.271 ± 0.956
User 6	105ms	2.890	3.616 ± 2.552	3.235	3.812 ± 2.181
	RN	1.642	1.976 ± 1.110	2.300	2.799 ± 1.506
	RO	1.658	1.997 ± 1.291	2.200	2.812 ± 1.377
	TRO	1.724	2.175 ± 1.620	2.413	2.738 ± 1.237
	25ms	2.217	2.450 ± 1.327	2.430	2.931 ± 1.519
User 7	105ms	2.037	2.872 ± 2.136	3.021	3.609 ± 1.989
	RN	1.510	1.786 ± 1.168	2.199	2.621 ± 1.121
	RO	1.479	1.782 ± 1.067	1.970	2.427 ± 1.270
	TRO	1.659	2.328 ± 1.659	2.495	2.831 ± 1.252
	25ms	1.412	1.727 ± 1.074	2.085	2.369 ± 1.014
User 8	105ms	2.216	2.724 ± 1.612	3.578	3.725 ± 1.992
	RN	1.445	1.808 ± 1.229	2.183	2.568 ± 1.093
	RO	1.740	2.252 ± 1.425	2.544	3.048 ± 1.525
	TRO	1.461	1.751 ± 1.235	2.200	2.488 ± 1.071
	25ms	1.609	2.064 ± 1.319	2.085	2.610 ± 1.232
User 9	105ms	2.381	3.519 ± 2.589	4.089	5.158 ± 3.299
	RN	1.757	2.149 ± 1.320	2.331	2.724 ± 1.274
	RO	1.559	2.083 ± 1.250	2.167	2.535 ± 1.281
	TRO	1.690	1.978 ± 1.069	2.364	2.561 ± 1.043
	25ms	1.576	1.863 ± 1.090	1.986	2.294 ± 0.860

Table 4. Per user task completion time (s) statistics by target type (per user-condition N in column headers).

Subject	Cond	Predictable (N=60 trials)		Unpredictable (N=90 trials)	
		Med	$\mu \pm \sigma$	Med	$\mu \pm \sigma$
User 1	105ms	1.972	2.248 ± 0.879	4.871	5.442 ± 2.919
	RN	1.470	1.559 ± 0.410	2.078	2.419 ± 1.279
	RO	1.577	1.757 ± 0.459	2.620	3.157 ± 1.799
	TRO	1.511	1.473 ± 0.360	2.111	2.237 ± 0.942
	25ms	1.454	1.544 ± 0.498	2.448	2.566 ± 1.124
User 2	105ms	1.816	2.097 ± 0.992	3.409	3.987 ± 2.548
	RN	1.560	1.662 ± 0.534	2.558	2.742 ± 1.379
	RO	1.642	1.649 ± 0.442	2.685	2.939 ± 1.501
	TRO	1.691	1.893 ± 0.794	2.398	2.782 ± 1.606
	25ms	1.413	1.471 ± 0.538	2.085	2.433 ± 1.479
User 3	105ms	1.535	1.615 ± 0.486	2.437	3.009 ± 1.576
	RN	1.667	1.837 ± 0.722	2.982	3.099 ± 1.499
	RO	1.518	1.514 ± 0.414	1.986	2.115 ± 0.948
	TRO	1.477	1.562 ± 0.551	2.240	2.397 ± 1.180
	25ms	1.395	1.417 ± 0.450	1.831	1.967 ± 0.906
User 4	105ms	1.781	1.951 ± 0.633	3.450	4.000 ± 2.287
	RN	1.510	1.636 ± 0.500	2.570	2.771 ± 1.408
	RO	1.542	1.553 ± 0.413	2.127	2.329 ± 1.015
	TRO	1.585	1.548 ± 0.483	2.447	2.612 ± 1.365
	25ms	1.478	1.429 ± 0.381	2.086	2.099 ± 0.898
User 5	105ms	1.691	1.896 ± 0.629	4.031	4.385 ± 2.215
	RN	1.675	1.744 ± 0.498	2.718	2.984 ± 1.455
	RO	1.526	1.586 ± 0.529	2.298	2.561 ± 1.313
	TRO	1.535	1.537 ± 0.469	2.208	2.336 ± 1.053
	25ms	1.470	1.464 ± 0.414	2.037	2.175 ± 1.054
User 6	105ms	1.855	2.062 ± 0.839	4.410	4.815 ± 2.430
	RN	1.510	1.592 ± 0.472	2.430	2.918 ± 1.532
	RO	1.625	1.651 ± 0.494	2.396	2.907 ± 1.565
	TRO	1.625	1.709 ± 0.548	2.545	2.955 ± 1.665
	25ms	1.740	1.911 ± 0.716	2.817	3.211 ± 1.571
User 7	105ms	1.716	1.847 ± 0.574	3.950	4.169 ± 2.224
	RN	1.535	1.601 ± 0.452	2.340	2.606 ± 1.390
	RO	1.502	1.508 ± 0.423	2.135	2.502 ± 1.396
	TRO	1.617	1.714 ± 0.536	3.005	3.157 ± 1.638
	25ms	1.453	1.491 ± 0.503	2.264	2.420 ± 1.215
User 8	105ms	1.625	1.918 ± 0.897	3.808	4.095 ± 1.860
	RN	1.510	1.566 ± 0.515	2.305	2.603 ± 1.374
	RO	1.633	1.745 ± 0.594	2.774	3.254 ± 1.659
	TRO	1.625	1.643 ± 0.664	2.200	2.437 ± 1.382
	25ms	1.510	1.552 ± 0.534	2.429	2.861 ± 1.401
User 9	105ms	2.077	2.387 ± 1.059	5.255	5.639 ± 3.286
	RN	1.560	1.621 ± 0.465	2.750	2.980 ± 1.435
	RO	1.477	1.621 ± 0.520	2.421	2.767 ± 1.431
	TRO	1.568	1.653 ± 0.597	2.446	2.680 ± 1.158
	25ms	1.436	1.512 ± 0.386	2.126	2.457 ± 1.108