Real-Time Camera Tracking: When is High Frame-Rate Best?

Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison

Department of Computing, Imperial College London, UK
{ahanda,rnewcomb,aangeli,ajd}@doc.ic.ac.uk

Abstract. Higher frame-rates promise better tracking of rapid motion, but advanced real-time vision systems rarely exceed the standard 10–60Hz range, arguing that the computation required would be too great. Actually, increasing frame-rate is mitigated by reduced computational cost *per frame* in trackers which take advantage of prediction. Additionally, when we consider the physics of image formation, high frame-rate implies that the upper bound on shutter time is reduced, leading to less motion blur but more noise. So, putting these factors together, how are application-dependent performance requirements of accuracy, robustness and computational cost optimised as frame-rate varies? Using 3D camera tracking as our test problem, and analysing a fundamental dense whole image alignment approach, we open up a route to a systematic investigation via the careful synthesis of photorealistic video using ray-tracing of a detailed 3D scene, experimentally obtained photometric response and noise models, and rapid camera motions. Our multi-frame-rate, multi-resolution, multi-light-level dataset is based on tens of thousands of hours of CPU rendering time. Our experiments lead to quantitative conclusions about frame-rate selection and highlight the crucial role of full consideration of physical image formation in pushing tracking performance.

1 Introduction

High frame-rate footage of bursting balloons or similar phenomena cannot fail to impress in the potential it offers to observe very fast motion. The potential for *real-time tracking* at extreme frame-rates was demonstrated by remarkable early work by researchers at the University of Tokyo (e.g. [1]) who developed custom vision chips operating at 1000Hz. Using essentially trivial algorithms (related to those used in an optical mouse), they tracked balls which were thrown, bounced and shaken, and coupled to slaved control of robotic camera platforms and manipulators. But while there are now commercial cameras available or even installed in commodity mobile devices which can provide video at 100Hz+, advanced real-time visual tracking algorithms rarely exceed 60Hz.

In this paper we take steps towards rigorous procedures for determining when and by how much it is advantageous to increase tracking frame-rate, by experimentally analysing the trade-offs to which this leads. Our first contribution is a precise framework for rendering photorealistic video of agile motion in realistically lit 3D scenes, with multiple, fully controllable settings and full ground

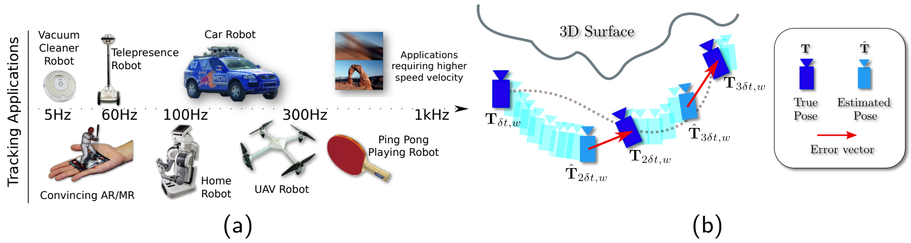


Fig. 1. a) Real-time visual tracking applications, and the frequency of the control signals needed. b) Tracking experiment configuration and evaluation. At each frame of our synthesized sequence, after tracking we record the translational distance between pose estimate $\hat{\mathbf{T}}_{t,w}$ and ground truth $\mathbf{T}_{t,w}$; an average of these distances over the sequence forms our accuracy measure. We then use the ground truth pose as the starting point for the next frame's tracking step.

truth, which enables systematic investigation of tracking performance. Carefully modelling physical image formation and using parameters from a real camera and usage setting, we have generated a substantial photorealistic video dataset by combining over 20000 ray-traced images. Our second contribution is a set of experiments using this data to investigate the limits of camera tracking and how they depend on frame-rate by analysing the performance of a fundamental dense alignment approach, leading to quantitative results on frame-rate optimisation.

1.1 Models, Prediction and Active Processing

Our scope is *real-time model-based tracking*, the problem of obtaining sequential estimates of parameters describing motion in the special case where these are needed 'in the loop' to drive applications in areas such as robotics or HCI (see Figure 1(a)). In particular, our main interest is camera ego-motion tracking through a known, mainly rigid scene. The 3D model of a scene such tracking requires, could come either from prior modelling or an interleaved reconstruction process (e.g. PTAM [2] or DTAM [3]) in a full SLAM system. The fact that in these and other state of the art SLAM systems, reconstruction (often called mapping) runs in a separable, much lower frequency process convinces us that even in SLAM it is sensible to analyse the properties of frame-rate tracking in isolation from the reconstruction process generating the tracked model.

A model consists of geometry, from abstract point and line features to non-parametric depth maps, meshes or implicit surfaces; and a photometric description to enable image correspondence, either local descriptors or a full RGB texture. For some types of model, a 'tracking as detection' approach of blanket recognition-style processing has been shown to be possible in real-time using very efficient keypoint matching methods such as [4]. However, while such methods have an important role in vision applications for relocalisation after tracking loss, they are not poised to benefit from the main advantage of increasing frame-rate in a system's main tracking loop — that increasingly good *predictions* can be made from one frame to the next, enabling guided, 'active' processing.

Many trackers based on keypoint or edge feature matching have implemented explicit prediction and to achieve real-time efficiency, normally by exhaustive search over probablistically-bounded small windows (e.g. [5]). Dense alignment methods, which do not use features but aim to find correspondence between a textured 3D model and every pixel of an input video frame, benefit from prediction in an automatic manner by requiring fewer iterations to optimise their cost functions when initialised from a better starting point. In camera tracking, modern parallel processing resources are now allowing the accuracy and robustness of dense alignment against whole scene models to surpass that possible with feature-based methods (e.g. [6],[3]); and the performance of multi-view stereo techniques now is so good that we strongly believe that it will soon be standard to assume that dense surface scene models are available as a matter of course in vision problems involving moving cameras (especially with the option of augmenting monocular video with commodity depth camera sensing, e.g. [7]).

1.2 Tracking via Dense Image to Model Alignment

Whole image alignment tracking against a dense model [6] [8] [3] operates by evaluating gradients of a similarity function and descending iteratively towards a minimum as in the original Lucas-Kanade algorithm [9]. In 3D camera tracking, for each live video frame optimal alignment is sought against a textured 3D surface model by minimising a cost function with respect to $\boldsymbol{\psi}$, the 6DOF parameters of general rigid body camera-scene motion $\mathtt{T}_{lr}(\boldsymbol{\psi})$. This cost comprises the sum of squares of the following image brightness difference at pixel $\mathbf{u}$ over all pixels in the live frame for which a valid view of the textured model is available:

$$f_{\mathbf{u}}(\boldsymbol{\psi}) = \mathbf{I}_l\left(\pi\left(\mathtt{K}\mathtt{T}_{lr}(\boldsymbol{\psi})\pi^{-1}\left(\mathbf{u}, \xi_r\left(\mathbf{u}\right)\right)\right)\right) - \mathbf{I}_r\left(\mathbf{u}\right) \ . \tag{1}$$

As in [3], here $\mathbf{I}_l$ denotes the current live video frame, and $\mathbf{I}_r$ the reference image obtained from the model (a projection from the last tracked pose of the textured scene model), with ξ_r the corresponding depth map; perspective projection is denoted by π and the fixed camera instrinsics matrix is $\mathtt{K}$. This cost function is non-convex, but is linearised on the assumption of small inter-frame motion and optimised iteratively by gradient descent. The size of its basin of convergence depends on the scale of the main structures in the image and textured model.

Dense gradient-descent based tracking algorithms implement active processing implicitly, since if run from the starting point of a good prediction they will require fewer iterations to converge to a minimum. Given an increase in framerate, we would expect that from one frame to the next the optimisation will converge faster and with less likelihood of gross failure as inter-frame motion decreases and the linearisation at the heart of Lucas-Kanade tracking becomes increasingly valid. Specifically then, besides the point that we are now seeing increasing practical use of dense tracking methods, we have chosen such a framework within which to perform our experimental study on tracking performance because iterative gradient-based image alignment aims in a direct, pure way at the best possible tracking performance (since it aims to align *all* of the data in an image against the scene model), and makes automatic the alterations in per-frame performance (computation, accuracy and robustness) we expect to see with changing frame-rate. Any feature-based method we might have instead chosen places an abstraction (feature selection and description) between image data and tracking performance which is different for every feature type and would lead us to question whether we were discovering fundamental properties of tracking performance or those of the features used. Feature algorithms have discrete, tuned parameters and thresholds. For example, above some level of motion blur most detectors will find no features at all as strong gradients have been wiped out; but dense tracking will still perform in some quantifiable way (as shown in DTAM's tracking through camera defocus). Further, as we will see in our experiments, there is a complicated interaction between physical image formation blur and noise effects and frame-rate. A dense tracking framework allows an analysis to be made on such degraded images without altering algorithm parameters that might be necessary for feature-based methods.

2 An Experimental Evaluation of Dense 3D Tracking

The kind of question we would like to answer via the analysis in this paper is as follows. In a given tuned tracking system, one day a new processor becomes available with twice the computation rate of the previous model, but at the same cost and power usage. We can surely drop in this new processor to increase tracking performance; but how should the algorithm best be altered? Some possibilities are: I) increase resolution; II) increase frame-rate and algorithm update rate; III) increase another parameter, such as number of optimisation iterations.

Clearly, the first question is to define tracking performance (also see Sturm *et al.*[10]). If we consider the distribution over a trajectory of the distance in model parameter space between estimated and ground truth pose, most trackers' performance can be expressed in terms of their average accuracy on frames when they essentially operate correctly, and a 'robustness' measure of how frequently this operation is achieved as opposed to gross failure. When computational cost is also considered, tracking performance is therefore characterised by at least three objectives, the importance of each of which will vary depending on the application. The theoretical best performance of a tracker is not a single point in objective space, but a Pareto Front [11] of possible operating points. With this understanding, the question at the top of this section could be answered by sliding up this front on the computational cost axis, and then trading off between the possible accuracy and robustness improvements this would permit.

In our experiments, we have analysed data where the motions are such that gross tracking failure is rare. Further, a full investigation of the robustness of tracking requires orders of magnitude more data such that meaningful statistics on tracking failure can be obtained. We hope to revisit this issue in the future, because one would guess that one of the main advantages of high frame-rate is improved *per second* robustness. We therefore focus on two measures in our results: accuracy during normal tracking, and computational cost. Our main results are in the form of two-objective plots, with Pareto Fronts suggesting application-dependent operating points a user might choose.

We have used custom photo-realistic video generation as the basis for our experiments, and there are several reasons for this compared to the options for setting up the equivalent real camera experiments. First, the full control we have over parameters (for example continuously variable frame-rate, resolution and shutter control with precisely repeatable motion) would be very challenging to match in a real experimental setting, since real cameras have a discrete range of settings. We have perfect ground truth on both camera motion and scene geometry, factors again subject to experimental error in real conditions. Perhaps most significantly, our experiments have highlighted the extreme importance of proper consideration of scene lighting conditions in evaluating tracking; even in a sophisticated and expensive experimental set-up with motion capture or a robot to track camera motion, and laser scanning to capture scene geometry, controlled lighting and light measurement apparatus would also be needed to ensure repeatability across different camera settings. As the next section details, we have gone to extreme lengths in endeavouring to produce experimental video which is not just as photorealistic as possible, but is based on parameters of a real camera and motion and realistic typical 3D scene. This is highlighted in the samples from our dataset shown in our submitted video. That is of course not to say that we are not interested in pursuing real experiments in future work.

Known scene geometry enables the dense tracking component of the DTAM [3] system for dense monocular SLAM to used as as the tracker within our experiments, a pyramidal, GPGPU implementation of the whole image alignment method explained in Section 1.2. DTAM uses a coarse to fine optimisation which operates on a traditional image pyramid, from low resolution 80 × 60 to full resolution 640 × 480. In our experiments, we use a fixed strategy during optimisation to determine when to make a transition from one pyramid level to the next, switching when the reduction in pose error from one iteration to the next is less than 0.0001cm. Note that this does imply some knowledge of ground truth during tracking which we consider reasonable in our experimental evaluation but a different measure of convergence would be needed in real tracking.

At given camera settings, we multiply the average execution time taken *per frame* for the tracker to converge to an error minimum by the frame-rate being used. This gives a value for the burden the tracker would place on the processor in dimension-free units of computational load — a value of one indicating full occupancy in real-time operation, and greater values indicating the need for multiple processing cards (highly feasible given the massively parallel nature of DTAM's tracking implementation) to be slotted in to achieve that performance.

To quantify accuracy we have used an error measure which is based only on the Euclidean translation distance between the estimated t_{est} and ground truth t_{gt} camera poses averaged over the whole trajectory; see Figure 1(b).

$$e = ||t_{est} - t_{gt}||_2 \ . \tag{2}$$

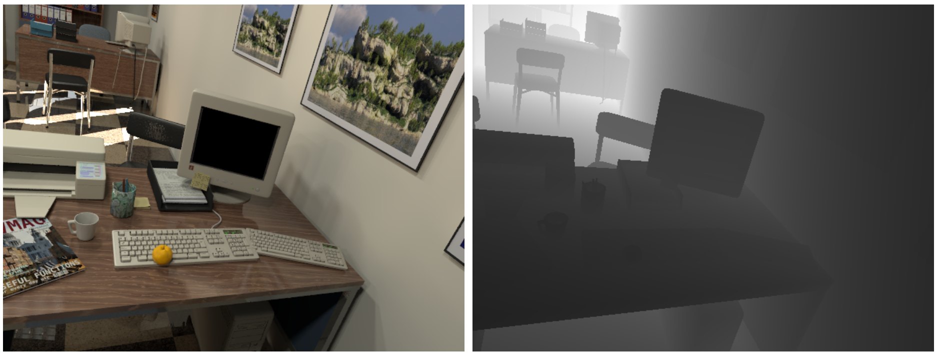


Fig. 2. A 'pure' ray-traced image with no blur or noise effects. Each such image takes 30–60mins to render in two passes on an 8 core Intel i7 3.20GHz machine. The associated planar depth map also generated is shown alongside.

3 Generating Synthetic Experimental Video

We have used the open-source ray tracer POV-Ray[1] as our main tool for video synthesis. We augment pure ray-traced images with effects simulating the motion blur and noise induced in physical image formation and combine them to form photorealistic video. POV-Ray has been used previously for ground truth generation [12] with regard to performance evaluation of a feature-based SLAM system, but with very simple non-realistic scenes. We have instead used a publicly available synthetic office scene model [2] ($800 \times 500 \times 250cm^3$) which we render with a simulated camera with standard perspective projection, image size 640×480 pixels and focal length 480 pixels — see Figure 2.

3.1 Experimental Modelling of a Real Camera

In order to set realistic rendering parameters for our synthetic sequences, we have performed experiments with a real high frame-rate camera to determine its camera response function (CRF) and noise characteristics. When a pixel on a camera's sensor chip is illuminated with light of irradiance E, during shutter time Δt it captures energy per unit area $E\Delta t$. This is then turned into a pixel brightness value B. We model this process as in [13]:

$$B = f(E\Delta t + n_s(\Delta t) + n_c) + n_q \ . \tag{3}$$

Here, the CRF f is the essential mapping between irradiance and brightness, monotonic and invertible. n_s is a shot noise random variable with zero mean and variance Var$(n_s(\Delta t))= E\Delta t\sigma_s^2$; and n_c is camera noise with zero mean and variance Var$(n_c)=\sigma_c^2$. We assume that quantisation noise n_q is negligible.

[1] The Persistence of Vision Raytracer, http://www.povray.org.

[2] http://www.ignorancia.org

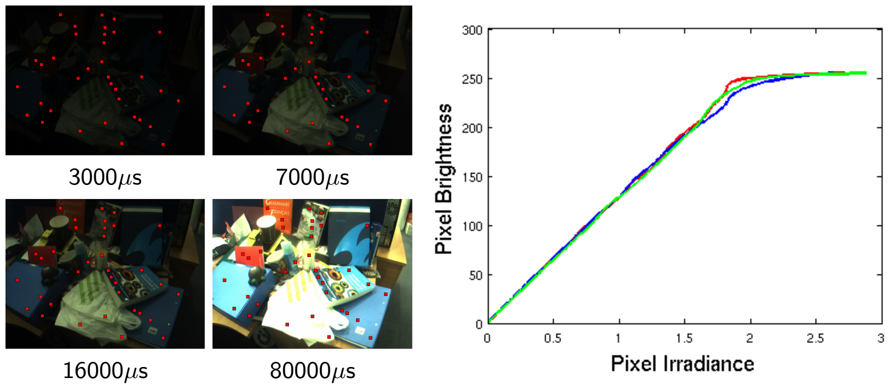


Fig. 3. Left: A selection of images obtained from a real camera with varying shutter time in microseconds. Red rectangles mark pixels manually selected to evenly sample scene irradiance whose brightness values are captured and used as input to CRF estimation. All images taken with zero gain and gamma off. Right: experimentally determined Camera Reponse Function (CRF) of our Basler piA640-210gc camera for each of the R, G and B colour channels with zero gain and gamma switched off using the method of [14]. This camera has a remarkably linear CRF up until saturation; over most of the range image brightness can be taken as proportional to irradiance. Note that the irradiance values determined by this method are up to scale and not absolute.

Determining the Camera Response Function We obtained f^{-1} up to scale for our camera using the chartless calibration method presented in [14], which requires multiple images of a static scene with a range of known shutter times (see the left side of Figure 3), recording the changing brightness values at a number of pixel locations chosen to span the irradiance variation in the scene. Since f is monotonic and invertible we can map from a given brightness value and shutter time via the inverse CRF f^{-1} back into irradiance. For each pixel i at shutter time Δt_j, we take the logarithm of the noise-free version of Equation 7:

$$\log f^{-1}(B_{ij}) = \log E_i + \log \Delta t_j \ . \tag{4}$$

Using measurements of 35 image pixels at 15 different shutter times, we solve for a parameterised form of f^{-1} under an $\mathcal{L}_2$ error norm using a second order smoothness prior. Figure 3 (right) shows the remarkably linear resulting CRF (gamma disabled) for the Basler piA640-210gc camera tested.

Noise Level Function Calibration To obtain the noise level function (NLF), multiple images of same static scene were taken at the same range of shutter times. For the chosen pixel locations, the mean and standard deviation brightness were recorded at each shutter setting (see Figure 4), separately for each colour channel. The lower envelope of this scatter plot is used to obtain the observed NLF, parameterised by σ_s and σ_c. To do this we define a function very similar to that used in [13] to model the likelihood of obtaining the measured data given

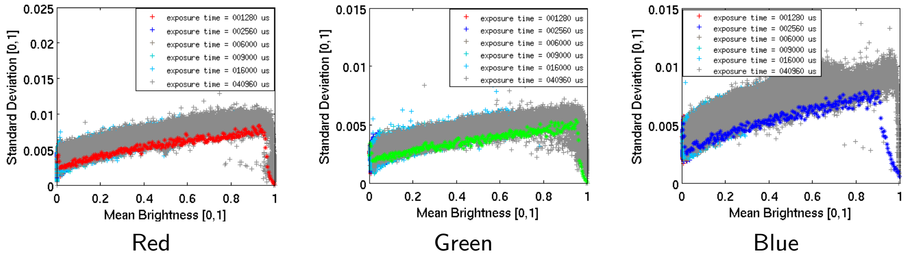


Fig. 4. Data and results for experimental Noise Level Function (NLF) calibration for each channel of our real camera shown as scatter plots. Overlaid on each of the plots is the brightness-dependent NLF computed using the minimisation technique formulated in section 3.1. Note that the green channel has significantly lower noise than red and blue due to there being twice as many green pixels in our camera's Bayer pattern.

the theoretical standard deviation $\tau(B_n)$ predicted by Equation 7:

$$\tau(B_n) = \left(\frac{\partial f(I)}{\partial I}\right)\sqrt{E\varDelta t\sigma_s^2 + \sigma_c^2}\ , \tag{5}$$

where E is the irradiance which maps to the brightness level B_n. This can be easily obtained using the inverse CRF obtained previously. We used the Matlab optimisation toolbox and standard functions `fmincon` (with constraints $\sigma_s \geq 0$ and $\sigma_c \geq 0$) and `fminunc` to obtain the optimal values. Optimisation results are overlaid on the observed NLFs from the images for each channel in Figure 4.

3.2 Inserting a Realistic Camera Trajectory

Going from still images to video requires a camera trajectory for the simulated camera to follow through the scene. After initially experimenting with various methods for sythesizing trajectories and finding these unsatisfactory, we decided to capture a trajectory from a real camera tracking experiment. Using DTAM [3] we tracked an extreme hand-held shaky motion which was at the limits of DTAM's state of the art tracking capability with a 30Hz camera. These poses are then transformed using a single similarity transform into the POV-Ray frame of reference to similarly render the synthetic scene. To obtain images for any frame-rate, the poses were interpolated, using cubic interpolation for translation and slerp for rotations. At our lowest experimental frame-rate of 20Hz, we see a maximum observed horizontal frame-to-frame motion of 260 pixels, nearly half of the whole 640 pixel wide image, indicating the rapidity of the motion.

3.3 Rendering Photorealistic Synthetic Sequences

We have followed previous research such as [15] in combining the 'perfect' rendered images from POV-Ray with our physical camera model to generate photorealistic video sequences. Since we do not have absolute scale for the irradiance

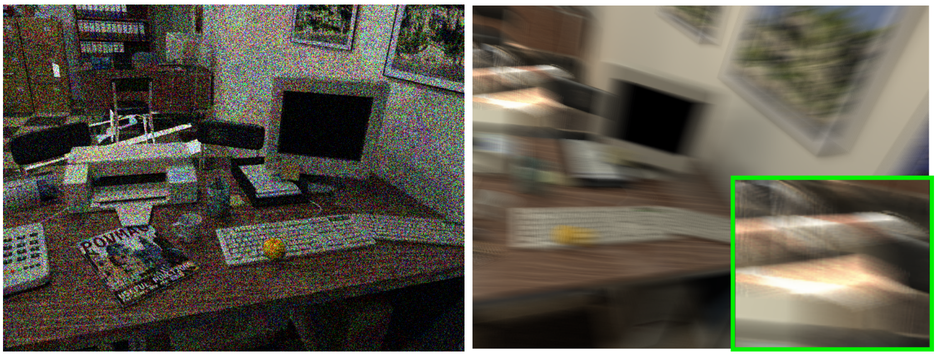


Fig. 5. Synthetic photorealistic images at shutter timings set to half of the maximum at a given frame-rate. Left: at 100Hz, images have very little blur but are dark and noisy. The image shown has brightness values rescaled for viewing purposes only. Right: at 20Hz, motion blur dominates. The cutout highlights our correct handling of image saturation by averaging for blur in irradiance space.

values obtained from the CRF, we cannot define scene illumination in absolute terms and instead define a new constant α representing overall scene brightness:

$$B = (\alpha E)\Delta t \ . \tag{6}$$

In our experiments we have values $\alpha \in \{1, 10, 40\}$, with increasing scene illumination leading to better image SNR. The reference E for each pixel is set to the base pixel value obtained from POV-Ray.

To account for camera motion we model motion blur. For a given shutter time Δt we compute the average of multiple POV-Ray frames i along the camera trajectory during that period followed by application of the noisy CRF:

$$B_{avg} = f\left(\frac{\alpha\Delta t}{N}\sum_{i=1}^{N} E_i + n_s + n_c\right) \ . \tag{7}$$

Finally, we quantise the obtained brightness function into an 8 bit per channel colour image. Figure 5 gives examples for simlar motion at 20 and 100Hz.

4 Tracking Analysis and Results

4.1 Experiments Assuming Perfect Lighting

We used the above framework to synthesize video at 10 different frame-rates in the range 20–200Hz using a five second rapid camera motion. To push frame-rate further we also synthesized 400 and 800Hz sequences. Although we only present results from this sequence and concentrate on clarity of interpretation, we have

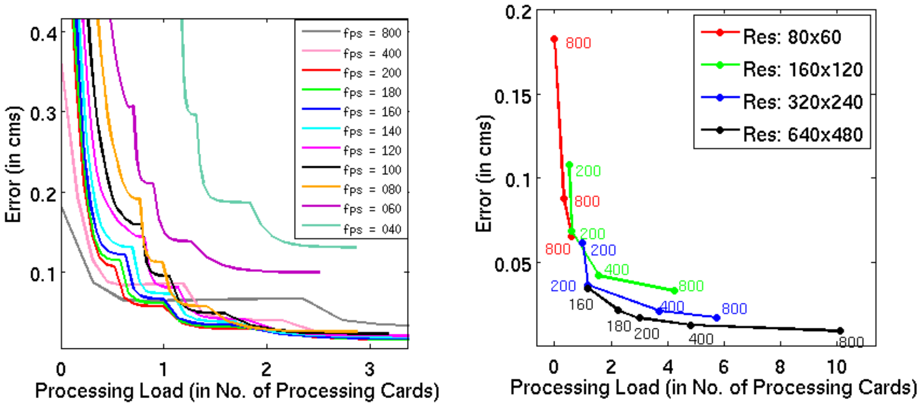


Fig. 6. Left: error plots for different frame-rates as a function of available computational budget under perfect lighting conditions. Points of high curvature denote the switch from one pyramid level to the next. Right: Pareto front for mininum error/minimum processing load performance, highlighting with numbers the frame-rates that are optimal for each available budget.

other sections of data with different scenes and motions where we have found very similar results in terms of tracking performance as a function of frame-rate.

Our first set of experiments assumes that whatever the frame-rate it is possible to assume blur- and noise-free images from the camera, and uses 'pure' ray-traced images as in Figure 2. This assumption is most closely represented in reality in an *extremely* well-lit scene, with shutter time set to a very low constant value at all frame-rates while still capturing enough light for good SNR.

Figure 6 shows the clean results we obtain in this case. In Figure 6(a) we plot how for each frame-rate setting, the average tracking error (our measure of accuracy) is reduced as more computation is thrown at the pyramidal LK optimisation per frame, permitting more iterations. Remember that the unit of computation we use is processor occupancy for real-time operation, obtained by multiplying computation time per frame by frame-rate. The sudden gradient changes in each curve are when the pyramidal optimisation switches from one level to the next — from low to high image resolution.

At the bottom of these overlapping curves is the interesting region of possible operating points where we can obtain the smallest average error as a function of processing load for our tracker, forming a Pareto Front. We display this more clearly in Figure 6(b), where we have rescaled the graph to focus on the interesting regions where crossovers occur, and used different lines coloured according to the maximum pyramid level resolution needed for that result (i.e. when we talk about resolution, we mean pyramidal optimisation using all resolutions of that value and below), and frame-rates are indicated as numbers. The behaviour of the Pareto Front is that generally very high frame-rates (200+) are optimal, with gradual interacting switching of maximum resolution and frame-rate as the processing budget increases and smaller errors can be obtained.

4.2 Experiments with Realistic Lighting Settings

We now extend our experiments to incorporate the shutter time-dependent noise and blur artifacts modelled in Section 3.1 that will affect most real world lighting conditions. We present a set of results for various global lighting levels.

We have used the same main frame-rate range of 20–200Hz and the same five second motion used with perfect images. We needed to make a choice about shutter time for each frame-rate, and while we have made some initial experiments (not presented here) on optimisation of shutter time, in these results we choose always half of the inverse of each frame-rate. To generate each synthesized video frame, we rendered multiple ray-traces for blur averaging from interpolated camera poses always 1.25ms part over the shutter time chosen; so to generate the 20Hz sequence with 25ms shutter time we needed to render $5 \times 20 \times 20 = 2000$ ray-traced frames. In fact this number is the same for every frame-rate (higher frame-rate countered by a reduced number of frames averaged for blurring), leading the the total of 20000 renders needed (with some duplicates).

An important detail is that dense matching is improved by pre-blurring the reprojected model template to match the level expected by the shutter time used so we implemented this; and the depth map is also be blurred by the same amount. We conducted some initial characterisation experiments to confirm aspects of the correct functioning of our photorealistic synthetic framework including this template blurring (Figure 7).

4.3 High Lighting

Figure 8 (a) shows the Pareto Front for α=40, where the scene light is high but not perfect. Images have good SNR, but high frame-rate images are darker; the 200Hz image is nearly 5 times darker than the corresponding image for perfect lighting and as frame-rate is pushed the images get darker and noisier. For clarity, we have omitted 400Hz and 800Hz in the graph. We observe that 200Hz at low resolution remains the best choice for low budgets, the image gradient information still strong enough to guide matching. And again a few iterations at 200Hz are enough because the baseline is short enough to aid accurate matching.

A further increase in budget reveals that 160Hz becomes the best choice for a higher resolution i.e. 320×240. This is where the error plots for frame-rates cross and better results are obtained by increasing resolution rather than frame-rate. As the processing load is further increased higher frame-rates are preferred, and the pattern repeats at 640×480. The plots however suggest a later transition to this highest resolution compared to the perfect sequence results in Figure 6 (b). The highest resolutions clearly have more benefit when SNR is high.

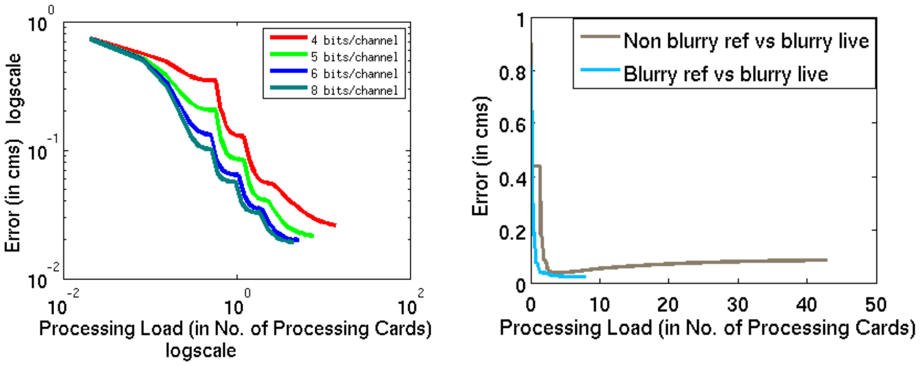


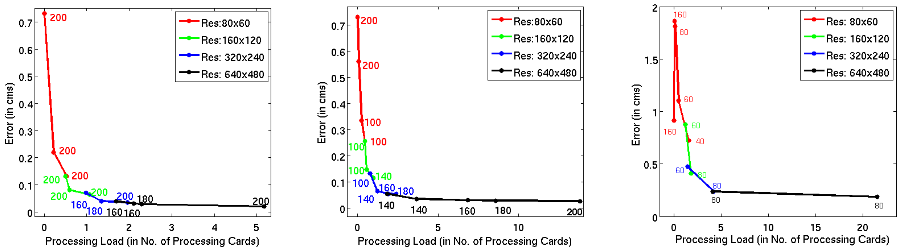
Fig. 7. Characterisation experiments to confirm our photorealistic synthetic video. Left: For a single motion and 200Hz frame-rate, we varied only shutter time to confirm that reduced light indeed leads to lower SNR and worse tracking performance. Here SNR is directly quantified in bits per colour channel. Right: An experiment explicitly showing that quality of tracking improves if a deliberately blurred prediction is matched against the blurry live image.

4.4 Moderate Lighting

In Figure 8(b) we reduce α to 10, representative of a typical indoor lighting setting. 200Hz is still the best choice at very low processing load when predictions are very strong and only one or two alignment iterations are sufficient. A slight increase in processing load sees the best choice of frame-rate shifting towards 100Hz even without resolution change, in contrast to both perfect lighting and high lighting conditions where working with very low processing load demands a high frame-rate. In moderate lighting, it is better to do more iterations at a lower frame-rate and take advantage of improved SNR because noise is too great at very high frame-rates to make multiple iterations worthwhile. When we do shift to 160×120 resolution, we see that the best frame-rate has shifted to 100–140Hz compared to 200Hz in high lighting conditions. Higher resolutions, 320×240 and 640×480, follow similar trends.

4.5 Low Lighting

Figure 8 (c) shows similar curves for scene illumination $\alpha = 1$, where images are now extremely dark. Our main observation here is that even at substantial processing load the Pareto Front does not feature frame-rates beyond 80Hz. These images have such low SNR that at high frame-rate essentially all tracking information has been destroyed. The overall quality of tracking results here is much lower (the error curve is much higher up the scale) as we would expect.



(a) α=40 (b) α=10 (c) α=1

Fig. 8. Pareto Fronts for lighting levels (a) $\alpha = 40$, (b) $\alpha = 10$ and (c) $\alpha = 1$. Numbers on the curves show the frame-rates that can be used to achieve the desired error with a given computational budget.

5 Conclusions

Our experiments give an acute insight into the trade-offs involved in high frame-rate tracking. With perfect lighting and essentially infinite SNR, the highest accuracy is achieved using a combination of high framerate and high resolution, with limits only set by the available computational budget. But using a realistic camera model, there is an optimal famerate for given lighting levels due to the tradeoff between SNR and motion blur. Therefore, frame-rate cannot be arbitrarily pushed up even if the budget allows because of image degradation. Decreasing lighting in the scene shifts the optimal frame-rates to slightly lower values that have higher SNR and somewhat more motion blur for all resolutions, but overall increasing resolution results in quicker improvement in accuracy than increasing frame-rate. Hasinoff *et al.*[16] [17] also worked on time-bound analysis but only for SNR image quality assessment with either static cameras or simple planar motions.

Our dataset, rendering scripts used to generate it and other materials are available from `http://www.doc.ic.ac.uk/~ahanda/HighFrameRateTracking/`. We hope this will further motivate both applied and theoretical investigations of high frame-rate tracking relevant to the design of practical vision systems. Our dataset may also be useful for analysis of many other 3D vision problems.

Acknowledgements

This research was supported by ERC Starting Grant 210346. We thank Steven Lovegrove, Hauke Strasdat, Margarita Chli and Thomas Pock for discussions.

References

1. Nakabo, Y., Ishikawa, M., Toyoda, H., Mizuno, S.: 1ms column parallel vision system and its application of high speed target tracking. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2000) 1

2. Klein, G., Murray, D.W.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR). (2007) 2
3. Newcombe, R.A., Lovegrove, S., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: Proceedings of the International Conference on Computer Vision (ICCV). (2011) 2, 3, 5, 8
4. Calonder, M., Lepetit, V., Konolige, K., Mihelich, P., Bowman, J., Fua, P.: Compact signatures for high-speed interest point description and matching. In: Proceedings of the International Conference on Computer Vision (ICCV). (2009) 2
5. Chli, M., Davison, A.J.: Active Matching. In: Proceedings of the European Conference on Computer Vision (ECCV). (2008) 3
6. Comport, A.I., Malis, E., Rives, P.: Accurate quadri-focal tracking for robust 3D visual odometry. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2007) 3
7. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR). (2011) 3
8. Meilland, M., Comport, A.I., Rives, P.: A spherical robot-centered representation for urban navigation. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). (2010) 3
9. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). (1981) 3
10. Sturm, J., Magnenat, S., Engelhard, N., Pomerleau, F., Colas, F., Burgard, W., Cremers, D., Siegwart, R.: Towards a benchmark for RGB-D SLAM evaluation. In: Proceedings of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems (RSS). (2011) 4
11. Calonder, M., Lepetit, V., Fua, P.: Pareto-optimal dictionaries for signatures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2010) 4
12. Funke, J., Pietzsch, T.: A framework for evaluating visual SLAM. In: Proceedings of the British Machine Vision Conference (BMVC). (2009) 6
13. Liu, C., Freeman, W.T., Szeliski, R., Kang, S.B.: Noise estimation from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2006) 6, 8
14. Debevec, P., Malik, J.: Recovering high dynamic range radiance maps from photographs. In: ACM Transactions on Graphics (SIGGRAPH). (1997) 7
15. Klein, G., Murray, D.W.: Simulating low-cost cameras for augmented reality compositing. IEEE Transactions on Visualization and Computer Graphics (VGC) **16** (2010) 369–380 8
16. Hasinoff, S.W., Kutulakos, K.N., Durand, F., Freeman, W.T.: Time-constrained photography. In: Proceedings of the International Conference on Computer Vision (ICCV). (2009) 13
17. Hasinoff, S.W., Durand, F., Freeman, W.T.: Noise-optimal capture for high dynamic range photography. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2010) 13