

Technical Writing: The Anatomy of a Research Paper

Preethi Jyothi
Department of CSE, IIT Bombay



ACM India Grad Cohort 2018
July 6, '18

Disclaimer

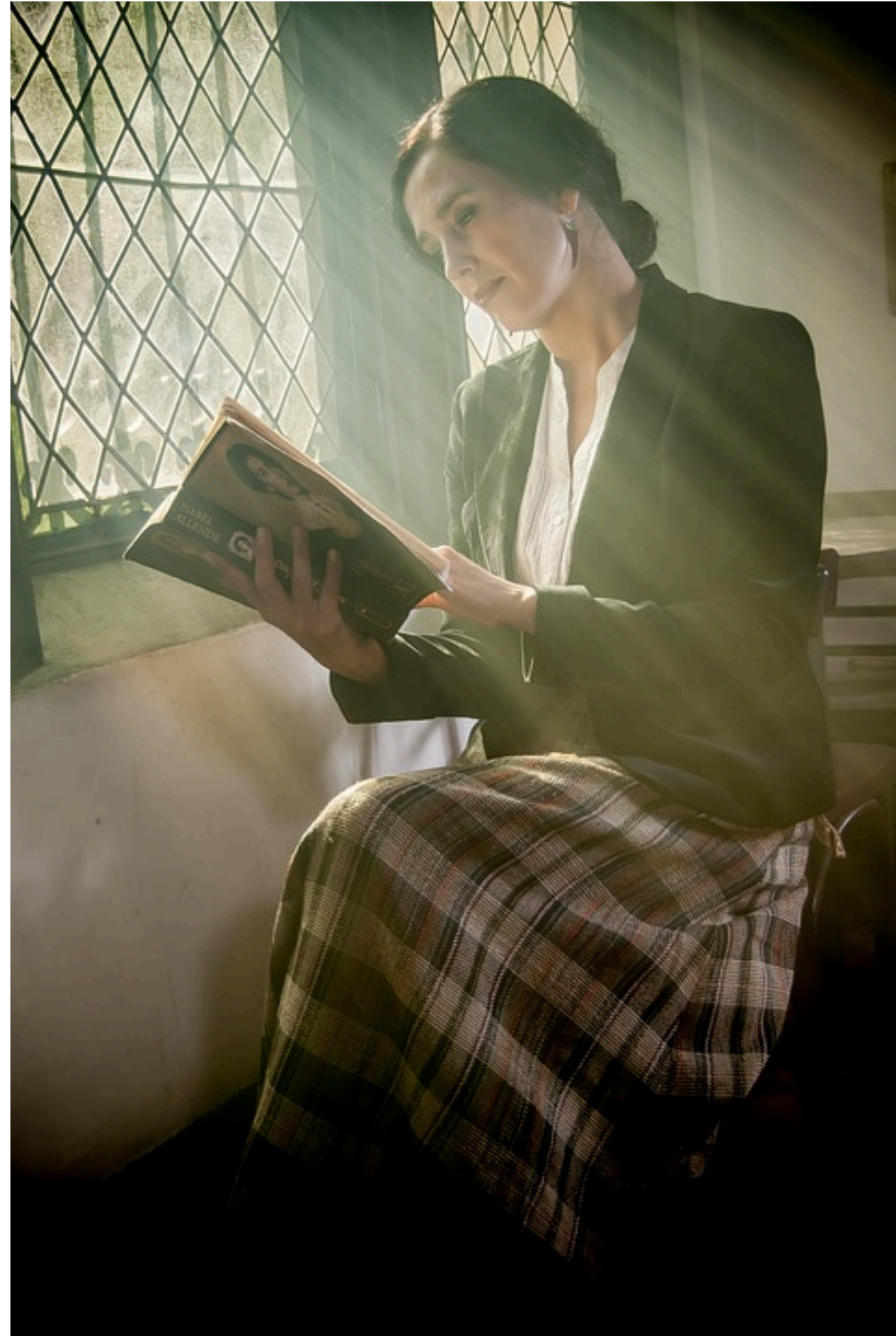
- A talk about technical writing can never be complete
- There are a large number of excellent online resources on this topic
 - ✓ [Knuth89] “Mathematical Writing”, Donald E. Knuth, Tracy Larrabee and Paul M. Roberts, 1989
 - ✓ [Gopen90] “The Science of Scientific Writing”, George Gopen and Judith Swan, American Scientist, 1990
 - [Lisberger11] “From Science to Citation: How to Publish a Successful Scientific Paper”, Stephen Lisberger, 2011
 - ✓ [Mensh17] “Ten Simple Rules for Structuring Papers”, Brett Mensh and Konrad Kording, PLOS Computational Biology, 2017
 - ✓ [Freeman18], “How to Write a Good Research Paper”, Bill Freeman, Good Citizen of CVPR, 2018 (<https://www.cc.gatech.edu/~parikh/citizenofcvpr/>)

Easy to lose sight of

The fundamental purpose of scientific discourse is not the mere presentation of information and thought, but rather its actual communication. It ***does not matter how pleased an author might be to have converted all the right data into sentences and paragraphs***; it matters only ***whether a large majority of the reading audience accurately perceives what the author had in mind***.

George Gopen, Judith Swan

For a reviewer of your paper



it's less like this...



WWW.PHDCOMICS.COM

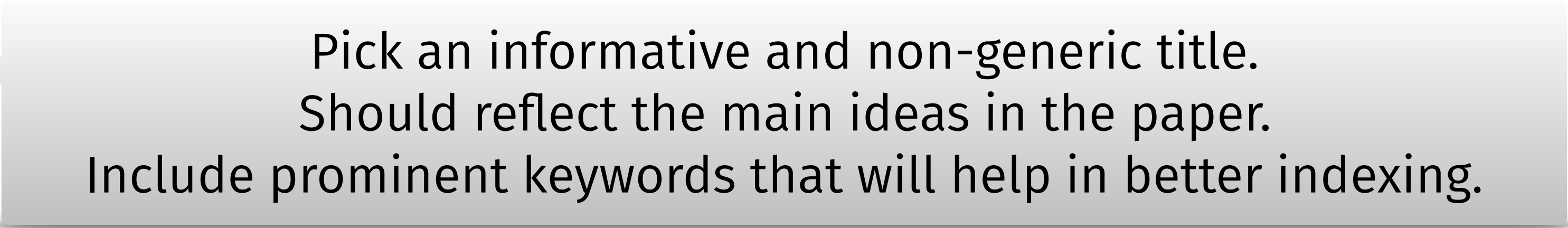
and more like this!

Skeleton of a paper [Mensh17]

- **Context:** State the problem you are tackling. Tell the reader why he/she should care about it.
- **Content:** What is your solution to the problem? Why is it different from previously proposed solutions?
- **Conclusion:** Discuss your approach. Analyse its benefits and weaknesses.

Structure of a typical paper

Varies depending on the subfield of computer science

- Title 

Pick an informative and non-generic title.
Should reflect the main ideas in the paper.
- Abstract

Include prominent keywords that will help in better indexing.
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Convey the entire message of the paper in the abstract.
Maintain the “context, content, conclusion” structure here.

Structure of a typical paper

Varies depending on the subfield of computer science

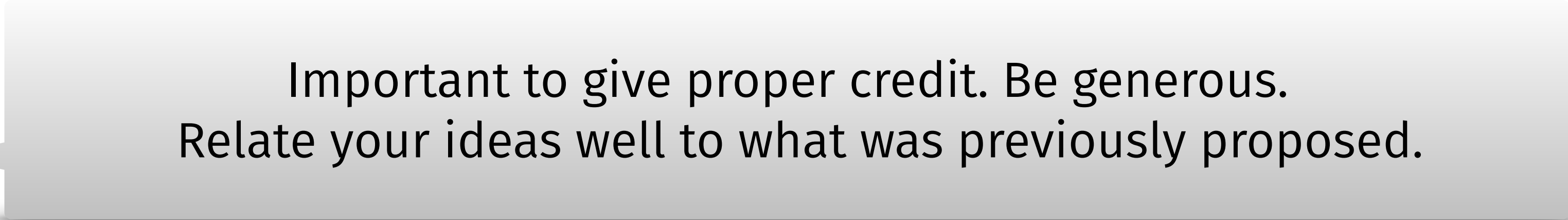
- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Can sell or sink your paper. Spend considerable time on this section!
Explain gaps in the literature and how your paper fills the gap.
Keep the reader interested and clearly summarise your main results.

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices



Important to give proper credit. Be generous.
Relate your ideas well to what was previously proposed.

Structure of a typical paper

Varies depending on the subfield of computer science

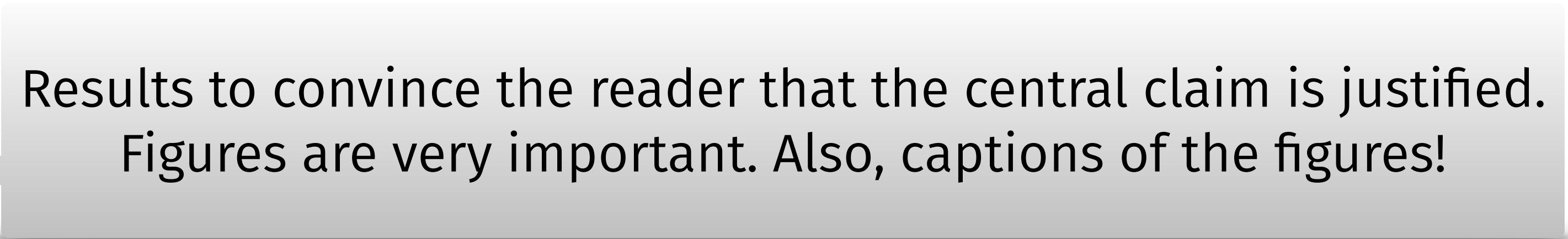
- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices

Organize your main content well. Typically multiple sections.
Ensure logical flow across and within sections.
Figures are a good idea.

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices



Results to convince the reader that the central claim is justified.
Figures are very important. Also, captions of the figures!

Structure of a typical paper

Varies depending on the subfield of computer science

- Title
- Abstract
- Introduction
- Related Work
- Main Ideas/Content
- Experiments/Results
- Discussion/Future Work/Conclusions
- Acknowledgments
- References
- Appendices



Evaluate strengths/weaknesses of your approach.
Preempt reviewers' comments about shortcomings.

Example paper from [Freeman18]

Removing Camera Shake from a Single Photograph

Rob Fergus¹ Barun Singh¹ Aaron Hertzmann² Sam T. Roweis² William T. Freeman¹

¹MIT CSAIL ²University of Toronto



Figure 1: *Left*: An image spoiled by camera shake. *Middle*: result from Photoshop “unsharp mask”. *Right*: result from our algorithm.

Abstract

Camera shake during exposure leads to objectionable image blur and ruins many photographs. Conventional blind deconvolution methods typically assume frequency-domain constraints on images, or overly simplified parametric forms for the motion path during camera shake. Real camera motions can follow convoluted paths, and a spatial domain prior can better maintain visually salient image characteristics. We introduce a method to remove the effects of camera shake from seriously blurred images. The method assumes a uniform camera blur over the image and negligible in-plane camera rotation. In order to estimate the blur from the camera shake, the user must specify an image region without saturation effects. We show results for a variety of digital photographs taken from personal photo collections.

CR Categories: I.4.3 [Image Processing and Computer Vision]: Enhancement, G.3 [Artificial Intelligence]: Learning

Keywords: camera shake, blind image deconvolution, variational learning, natural image statistics

1 Introduction

Camera shake, in which an unsteady camera causes blurry photographs, is a chronic problem for photographers. The explosion of

depth-of-field. A tripod, or other specialized hardware, can eliminate camera shake, but these are bulky and most consumer photographs are taken with a conventional, handheld camera. Users may avoid the use of flash due to the unnatural tonescales that result. In our experience, many of the otherwise favorite photographs of amateur photographers are spoiled by camera shake. A method to remove that motion blur from a captured photograph would be an important asset for digital photography.

Camera shake can be modeled as a blur kernel, describing the camera motion during exposure, convolved with the image intensities. Removing the unknown camera shake is thus a form of blind image deconvolution, which is a problem with a long history in the image and signal processing literature. In the most basic formulation, the problem is underconstrained: there are simply more unknowns (the original image and the blur kernel) than measurements (the observed image). Hence, all practical solutions must make strong prior assumptions about the blur kernel, about the image to be recovered, or both. Traditional signal processing formulations of the problem usually make only very general assumptions in the form of frequency-domain power laws; the resulting algorithms can typically handle only very small blurs and not the complicated blur kernels often associated with camera shake. Furthermore, algorithms exploiting image priors specified in the frequency domain may not preserve important spatial-domain structures such as edges.

This paper introduces a new technique for removing the effects of

Removing Camera Shake from a Single Photograph

Fergus et al. 2006

above assumptions are violated; however, they may be acceptable to consumers in some cases, and a professional designer could touch-up the results. In contrast, the original images are typically unusable, beyond touching-up — in effect our method can help “rescue” shots that would have otherwise been completely lost.

2 Related Work

The task of deblurring an image is image deconvolution; if the blur kernel is not known, then the problem is said to be “blind”. For a survey on the extensive literature in this area, see [Kundur and Hatzinakos 1996]. Existing blind deconvolution methods typically assume that the blur kernel has a simple parametric form, such as a Gaussian or low-frequency Fourier components. However, as illustrated by our examples, the blur kernels induced during camera shake do not have simple forms, and often contain very sharp edges. Similar low-frequency assumptions are typically made for the input image, e.g., applying a quadratic regularization. Such assumptions can prevent high frequencies (such as edges) from appearing in the reconstruction. Caron et al. [2002] assume a power-law distribution on the image frequencies; power-laws are a simple form of natural image statistics that do not preserve local structure. Some methods [Jalobeanu et al. 2002; Neelamani et al. 2004] combine power-laws with wavelet domain constraints but do not work for the complex blur kernels in our examples.

Deconvolution methods have been developed for astronomical images [Gull 1998; Richardson 1972; Tsumuraya et al. 1994; Zarowin 1994], which have statistics quite different from the natural scenes we address in this paper. Performing blind deconvolution in this domain is usually straightforward, as the blurry image of an isolated star reveals the point-spread-function.

Another approach is to assume that there are multiple images available of the same scene [Bascle et al. 1996; Rav-Acha and Peleg 2005]. Hardware approaches include: optically stabilized lenses [Canon Inc. 2006], specially designed CMOS sensors [Liu and Gamal 2001], and hybrid imaging systems [Ben-Ezra and Nayar 2004]. Since we would like our method to work with existing cameras and imagery and to work for as many situations as possible, we do not assume that any such hardware or extra imagery is available.

Recent work in computer vision has shown the usefulness of heavy-tailed natural image priors in a variety of applications, including denoising [Roth and Black 2005], superresolution [Tappen et al. 2003], intrinsic images [Weiss 2001], video matting [Apostoloff and Fitzgibbon 2005], inpainting [Levin et al. 2003], and separating reflections [Levin and Weiss 2004]. Each of these methods is effectively “non-blind”, in that the image formation process (e.g., the blur kernel in superresolution) is assumed to be known in advance.

Miskin and MacKay [2000] perform blind deconvolution on line art images using a prior on raw pixel intensities. Results are shown for small amounts of synthesized image blur. We apply a similar varia-

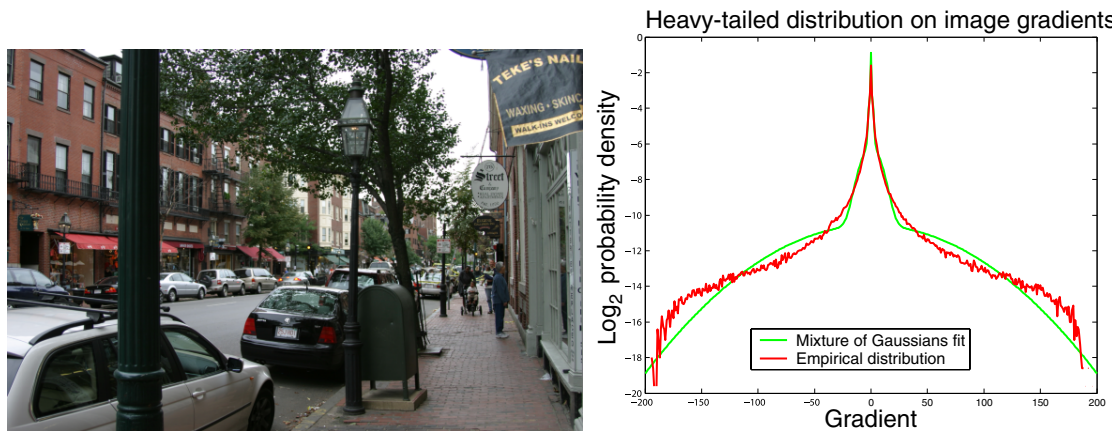


Figure 2: *Left:* A natural scene. *Right:* The distribution of gradient magnitudes within the scene are shown in red. The y-axis has a logarithmic scale to show the heavy tails of the distribution. The mixture of Gaussians approximation used in our experiments is shown in green.

the sensor irradiance. The latent image \mathbf{L} represents the image we would have captured if the camera had remained perfectly still; our goal is to recover \mathbf{L} from \mathbf{B} without specific knowledge of \mathbf{K} .

In order to estimate the latent image from such limited measurements, it is essential to have some notion of which images are *a-priori* more likely. Fortunately, recent research in natural image statistics have shown that, although images of real-world scenes vary greatly in their absolute color distributions, they obey *heavy-tailed* distributions in their gradients [Field 1994]: the distribution of gradients has most of its mass on small values but gives significantly more probability to large values than a Gaussian distribution. This corresponds to the intuition that images often contain large sections of constant intensity or gentle intensity gradient interrupted by occasional large changes at edges or occlusion boundaries. For example, Figure 2 shows a natural image and a histogram of its gradient magnitudes. The distribution shows that the image contains primarily small or zero gradients, but a few gradients have large magnitudes. Recent image processing methods based on heavy-tailed distributions give state-of-the-art results in image denoising [Roth and Black 2005; Simoncelli 2005] and superresolution [Tappen et al. 2003]. In contrast, methods based on Gaussian prior distributions (including methods that use quadratic regularizers) produce overly smooth images.

We represent the distribution over gradient magnitudes with a zero-mean mixture-of-Gaussians model, as illustrated in Figure 2. This representation was chosen because it can provide a good approximation to the empirical distribution, while allowing a tractable estimation procedure for our algorithm.

4 Algorithm

There are two main steps to our approach. First, the blur kernel is estimated from the input image. The estimation process is performed in a coarse-to-fine fashion in order to avoid local minima. Second, using the estimated kernel, we apply a standard deconvolution.

4.1 Estimating the blur kernel

Given the grayscale blurred patch \mathbf{P} , we estimate \mathbf{K} and the latent patch image \mathbf{L}_p by finding the values with highest probability, guided by a prior on the statistics of \mathbf{L} . Since these statistics are based on the image gradients rather than the intensities, we perform the optimization in the gradient domain, using $\nabla \mathbf{L}_p$ and $\nabla \mathbf{P}$, the gradients of \mathbf{L}_p and \mathbf{P} . Because convolution is a linear operation, the patch gradients $\nabla \mathbf{P}$ should be equal to the convolution of the latent gradients and the kernel: $\nabla \mathbf{P} = \nabla \mathbf{L}_p \otimes \mathbf{K}$, plus noise. We assume that this noise is Gaussian with variance σ^2 .

As discussed in the previous section, the prior $p(\nabla \mathbf{L}_p)$ on the latent image gradients is a mixture of C zero-mean Gaussians (with variance v_c and weight π_c for the c -th Gaussian). We use a sparsity prior $p(\mathbf{K})$ for the kernel that encourages zero values in the kernel, and requires all entries to be positive. Specifically, the prior on kernel values is a mixture of D exponential distributions (with scale factors λ_d and weights π_d for the d -th component).

Given the measured image gradients $\nabla \mathbf{P}$, we can write the posterior distribution over the unknowns with Bayes’ Rule:

$$p(\mathbf{K}, \nabla \mathbf{L}_p | \nabla \mathbf{P}) \propto p(\nabla \mathbf{P} | \mathbf{K}, \nabla \mathbf{L}_p) p(\nabla \mathbf{L}_p) p(\mathbf{K}) \quad (2)$$

$$= \prod_i \mathcal{N}(\nabla \mathbf{P}(i) | (\mathbf{K} \otimes \nabla \mathbf{L}_p(i)), \sigma^2) \quad (3)$$

$$\prod_i \sum_{c=1}^C \pi_c \mathcal{N}(\nabla \mathbf{L}_p(i) | 0, v_c) \prod_j \sum_{d=1}^D \pi_d \mathcal{E}(\mathbf{K}_j | \lambda_d)$$

where i indexes over image pixels and j indexes over blur kernel elements. \mathcal{N} and \mathcal{E} denote Gaussian and Exponential distributions respectively. For tractability, we assume that the gradients in $\nabla \mathbf{P}$ are independent of each other, as are the elements in $\nabla \mathbf{L}_p$ and \mathbf{K} .

A straightforward approach to deconvolution is to solve for the maximum a-posteriori (MAP) solution, which finds the kernel \mathbf{K} and latent image gradients $\nabla \mathbf{L}$ that maximizes $p(\mathbf{K}, \nabla \mathbf{L}_p | \nabla \mathbf{P})$. This is equivalent to solving a regularized-least squares problem that attempts to fit the data while also minimizing small gradients. We tried this (using conjugate gradient search) but found that the algorithm failed. One interpretation is that the MAP objective function attempts to minimize all gradients (even large ones), whereas we expect natural images to have some large gradients. Consequently, the algorithm yields a two-tone image, since virtually all the gradients are zero. If we reduce the noise variance (thus increasing the weight on the data-fitting term), then the algorithm yields a delta-function for \mathbf{K} , which exactly fits the blurred image, but without any deblurring. Additionally, we find the MAP objective function to be very susceptible to poor local minima.

Instead, our approach is to approximate the full posterior distribution $p(\mathbf{K}, \nabla \mathbf{L}_p | \nabla \mathbf{P})$, and then compute the kernel \mathbf{K} with max-

Following Miskin and MacKay [2000], we also treat the noise variance σ^2 as an unknown during the estimation process, thus freeing the user from tuning this parameter. This allows the noise variance to vary during estimation: the data-fitting constraint is loose early in the process, becoming tighter as better, low-noise solutions are found. We place a prior on σ^2 , in the form of a Gamma distribution on the inverse variance, having hyper-parameters a, b : $p(\sigma^2 | a, b) = \Gamma(\sigma^{-2} | a, b)$. The variational posterior of σ^2 is $q(\sigma^{-2})$, another Gamma distribution.

The variational algorithm minimizes a cost function representing the distance between the approximating distribution and the true posterior, measured as: $KL(q(\mathbf{K}, \nabla \mathbf{L}_p, \sigma^{-2}) || p(\mathbf{K}, \nabla \mathbf{L}_p | \nabla \mathbf{P}))$. The independence assumptions in the variational posterior allows the cost function C_{KL} to be factored:

$$\langle \log \frac{q(\nabla \mathbf{L}_p)}{p(\nabla \mathbf{L}_p)} \rangle_{q(\nabla \mathbf{L}_p)} + \langle \log \frac{q(\mathbf{K})}{p(\mathbf{K})} \rangle_{q(\mathbf{K})} + \langle \log \frac{q(\sigma^{-2})}{p(\sigma^{-2})} \rangle_{q(\sigma^{-2})} \quad (4)$$

where $\langle \cdot \rangle_{q(\theta)}$ denotes the expectation with respect to $q(\theta)^2$. For brevity, the dependence on $\nabla \mathbf{P}$ is omitted from this equation.

The cost function is then minimized as follows. The means of the distributions $q(\mathbf{K})$ and $q(\nabla \mathbf{L}_p)$ are set to the initial values of \mathbf{K} and $\nabla \mathbf{L}_p$ and the variance of the distributions set high, reflecting the lack of certainty in the initial estimate. The parameters of the distributions are then updated alternately by coordinate descent; one is updated by marginalizing out over the other whilst incorporating the model priors. Updates are performed by computing closed-form optimal parameter updates, and performing line-search in the direction of these updated values (see Appendix A for details). The updates are repeated until the change in C_{KL} becomes negligible. The mean of the marginal distribution $\langle \mathbf{K} \rangle_{q(\mathbf{K})}$ is then taken as the final value for \mathbf{K} . Our implementation adapts the source code provided online by Miskin and MacKay [2000a].

In the formulation outlined above, we have neglected the possibility of saturated pixels in the image, an awkward non-linearity which violates our model. Since dealing with them explicitly is complicated, we prefer to simply mask out saturated regions of the image during the inference procedure, so that no use is made of them.

For the variational framework, $C = D = 4$ components were used in the priors on \mathbf{K} and $\nabla \mathbf{L}_p$. The parameters of the prior on the latent image gradients π_c, v_c were estimated from a single street scene image, shown in Figure 2, using EM. Since the image statistics vary across scale, each scale level had its own set of prior parameters. This prior was used for all experiments. The parameters for the prior on the blur kernel elements were estimated from a small set of low-noise kernels inferred from real images.

4.1.1 Multi-scale approach

Removing Camera Shake from a Single Photograph

Fergus et al. 2006

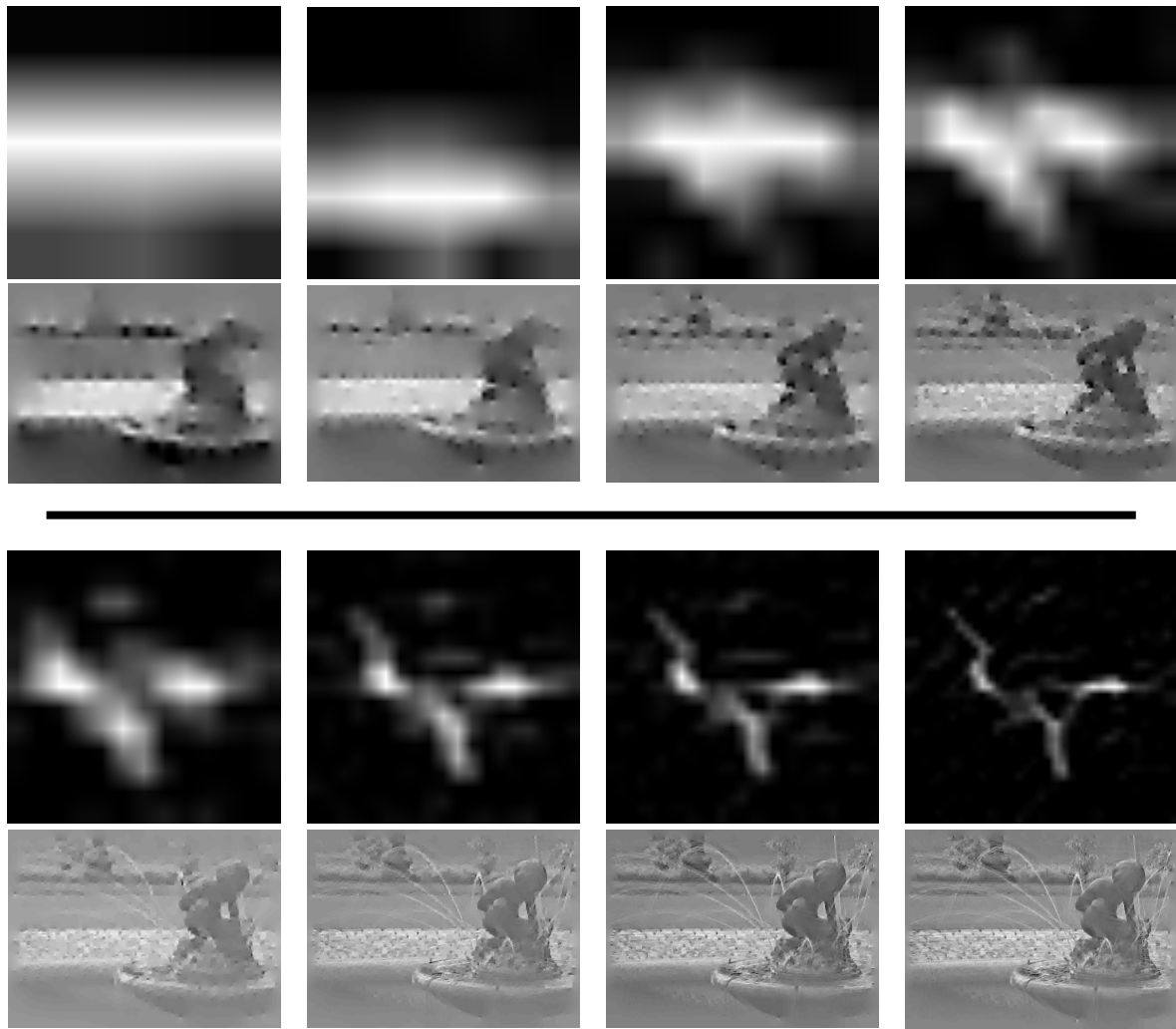


Figure 3: The multi-scale inference scheme operating on the fountain image in Figure 1. *1st & 3rd rows*: The estimated blur kernel at each scale level. *2nd & 4th rows*: Estimated image patch at each scale. The intensity image was reconstructed from the gradients used in the inference using Poisson image reconstruction. The Poisson reconstructions are shown for reference only; the final reconstruction is found using the Richardson-Lucy algorithm with the final estimated blur kernel.

4.1.2 User supervision

Although it would seem more natural to run the multi-scale inference scheme using the full gradient image $\nabla \mathbf{L}$, in practice we found the algorithm performed better if a smaller patch, rich in edge structure, was manually selected. The manual selection allows the user to avoid large areas of saturation or uniformity, which can be disruptive or uninformative to the algorithm. Examples of user-selected patches are shown in Section 5. Additionally, the algorithm runs much faster on a small patch than on the entire image.

An additional parameter is that of the maximum size of the blur kernel. The size of the blur encountered in images varies widely, from a few pixels up to hundreds. Small blurs are hard to resolve if the algorithm is initialized with a very large kernel. Conversely, large blurs will be cropped if too small a kernel is used. Hence, for operation under all conditions, the approximate size of the kernel is a required input from the user. By examining any blur artifact in the image, the size of the kernel is easily deduced.

synthetic test examples, our real images exhibit a range of non-linearities not present in synthetic cases, such as non-Gaussian noise, saturated pixels, residual non-linearities in tonescale and estimation errors in the kernel. Disappointingly, when run on our images, most methods produced unacceptable levels of artifacts.

We also used our variational inference scheme on the gradients of the whole image $\nabla \mathbf{B}$, while holding \mathbf{K} fixed. The intensity image was then formed via Poisson image reconstruction [Weiss 2001]. Aside from being slow, the inability to model the non-linearities mentioned above resulted in reconstructions no better than other approaches.

As \mathbf{L} typically is large, speed considerations make simple methods attractive. Consequently, we reconstruct the latent color image \mathbf{L} with the Richardson-Lucy (RL) algorithm [Richardson 1972; Lucy 1974]. While the RL performed comparably to the other methods evaluated, it has the advantage of taking only a few minutes, even on large images (other, more complex methods, took hours or days). RL is a non-blind deconvolution algorithm that iteratively maximizes the likelihood function of a Poisson statistics image noise model. One benefit of this over more direct methods is that it gives only non-negative output values. We use Matlab’s implementation of the algorithm to estimate \mathbf{L} , given \mathbf{K} , treating each color channel independently. We used 10 RL iterations, although for large blur kernels, more may be needed. Before running RL, we clean up \mathbf{K} by applying a dynamic threshold, based on the maximum intensity value within the kernel, which sets all elements below a certain value to zero, so reducing the kernel noise. The output of RL was then gamma-corrected using $\gamma = 2.2$ and its intensity histogram matched to that of \mathbf{B} (using Matlab’s `histeq` function), resulting in \mathbf{L} . See pseudo-code in Appendix A for details.

5 Experiments

We performed an experiment to check that blurry images are mainly due to camera translation as opposed to other motions, such as in-plane rotation. To this end, we asked 8 people to photograph a whiteboard³ which had small black dots placed in each corner whilst using a shutter speed of 1 second. Figure 4 shows dots extracted from a random sampling of images taken by different people. The dots in each corner reveal the blur kernel local to that portion of the image. The blur patterns are very similar, showing that our assumptions of spatially invariant blur with little in plane rotation are valid.

We apply our algorithm to a number of real images with varying degrees of blur and saturation. All the photos came from personal photo collections, with the exception of the fountain and cafe images which were taken with a high-end DSLR using long exposures

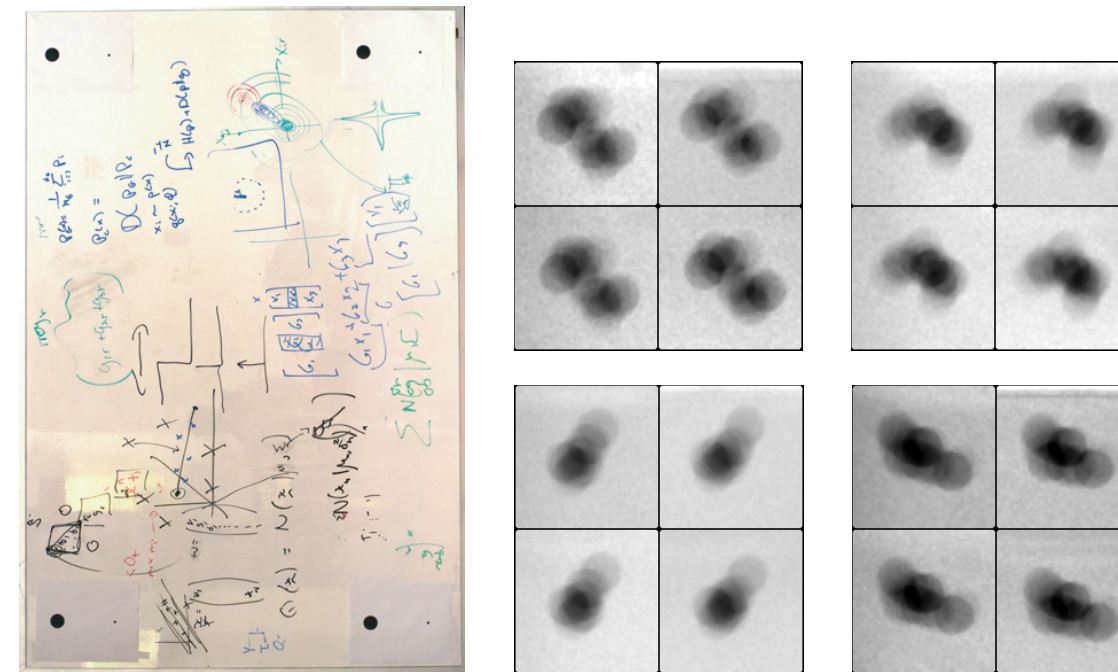


Figure 4: *Left*: The whiteboard test scene with dots in each corner. *Right*: Dots from the corners of images taken by different people. Within each image, the dot trajectories are very similar suggesting that image blur is well modeled as a spatially invariant convolution.



Figure 6: *Top*: A scene with complex motions. While the motion of the camera is small, the child is both translating and, in the case of the arm, rotating. *Bottom*: Output of our algorithm. The face and shirt are sharp but the arm remains blurred, its motion not modeled by our algorithm.

As demonstrated in Figure 8, the true blur kernel is occasionally revealed in the image by the trajectory of a point light source transformed by the blur. This gives us an opportunity to compare the inferred blur kernel with the true one. Figure 10 shows four such image structures, along with the inferred kernels from the respective images.

We also compared our algorithm against existing blind deconvolution algorithms, running Matlab’s `deconvblind` routine, which provides implementations of the methods of Biggs and Andrews

Removing Camera Shake from a Single Photograph

Fergus et al. 2006



Figure 7: *Top*: A scene with a large blur. *Bottom*: Output of our algorithm. See Figure 8 for a closeup view.

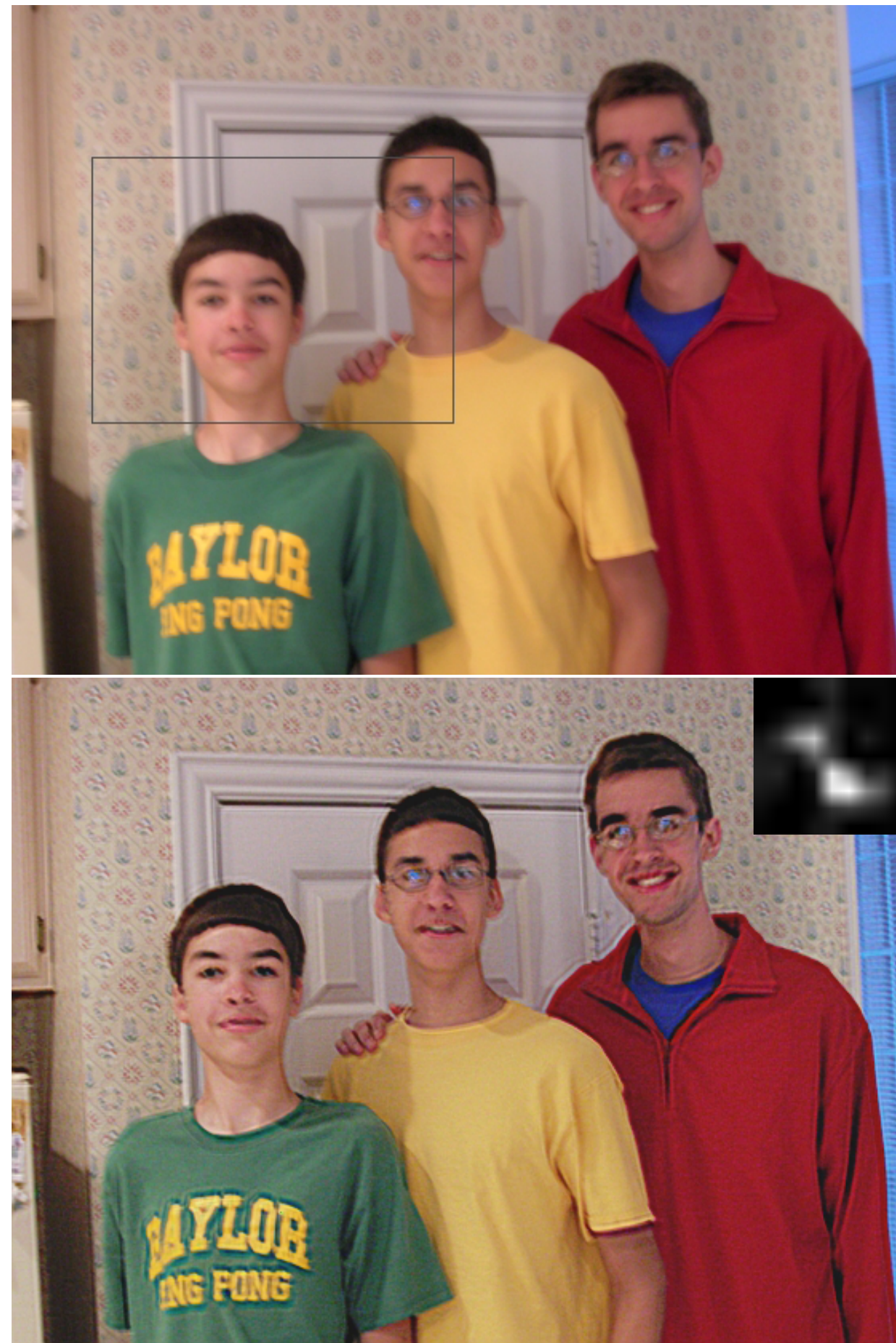
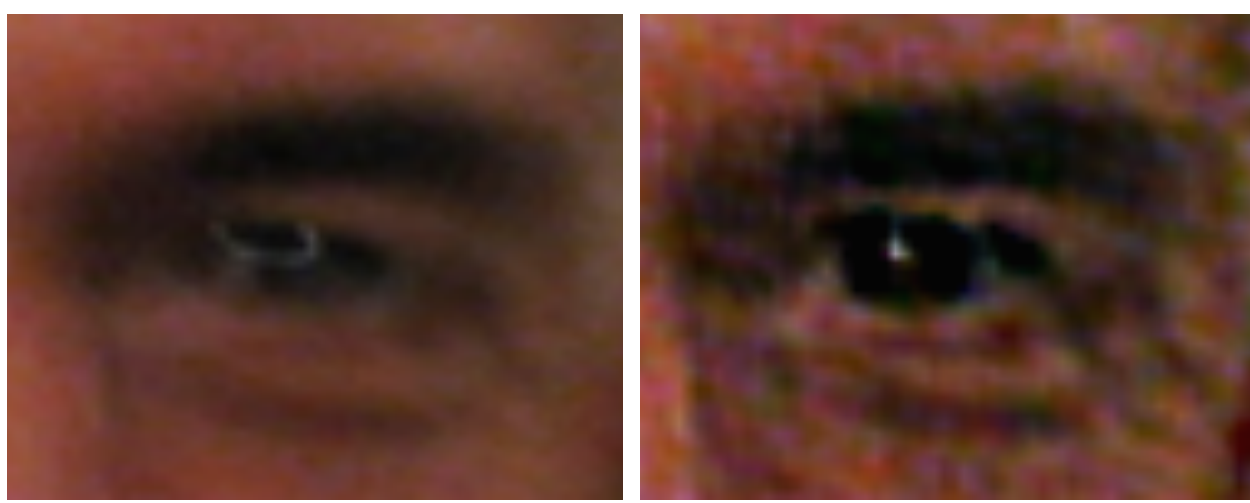


Figure 9: *Top*: A blurry photograph of three brothers. *Bottom*: Output of our algorithm. The fine detail of the wallpaper is now visible.

6 Discussion

We have introduced a method for removing camera shake effects from photographs. This problem appears highly underconstrained at first. However, we have shown that by applying natural image priors and advanced statistical techniques, plausible results can be obtained.

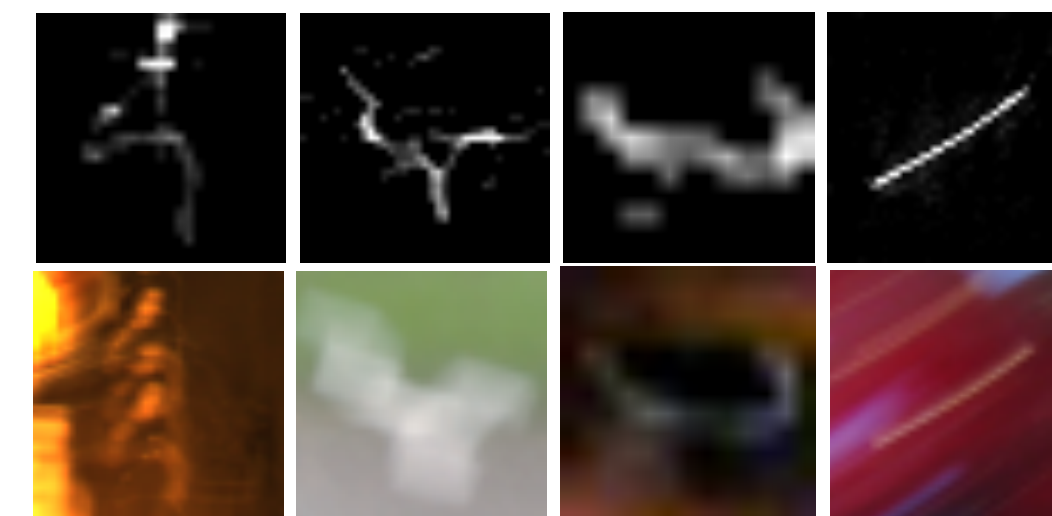


Figure 10: *Top row*: Inferred blur kernels from four real images (the cafe, fountain and family scenes plus another image not shown). *Bottom row*: Patches extracted from these scenes where the true kernel has been revealed. In the cafe image, two lights give a dual image of the kernel. In the fountain scene, a white square is transformed by the blur kernel. The final two images have specularities transformed by the camera motion, revealing the true kernel.

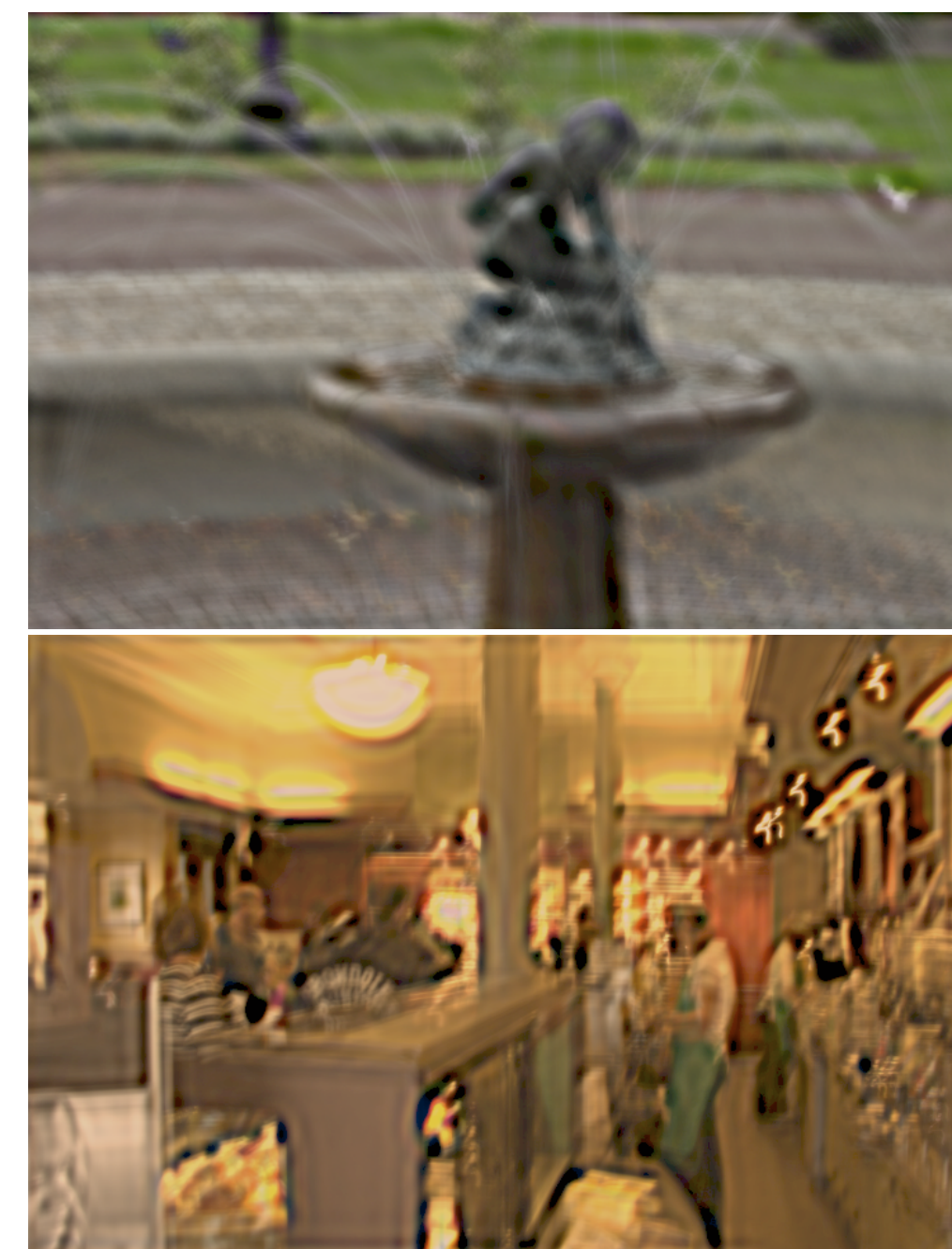


Figure 11: Baseline experiments, using Matlab's blind deconvolution algorithm deconvblind on the fountain image (top) and cafe



Figure 12: *Top*: A blurred scene with significant saturation. The long thin region selected by the user has limited saturation. *Bottom*: output of our algorithm. Note the double exposure type blur kernel



Removing Camera Shake from a Single Photograph

Fergus et al. 2006

Acknowledgements

We are indebted to Antonio Torralba, Don Geman and Fredo Durand for their insights and suggestions. We are most grateful to James Miskin and David MacKay, for making their code available online. We would like to thank the following people for supplying us with blurred images for the paper: Omar Khan, Reinhard Klette, Michael Lewicki, Pietro Perona and Elizabeth Van Ruitenbeek. Funding for the project was provided by NSERC, NGA NEGI-1582-04-0004 and the Shell Group.

References

- APOSTOLOFF, N., AND FITZGIBBON, A. 2005. Bayesian video matting using learnt image priors. In *Conf. on Computer Vision and Pattern Recognition*, 407–414.
- BASCLE, B., BLAKE, A., AND ZISSERMAN, A. 1996. Motion Deblurring and Super-resolution from an Image Sequence. In *ECCV (2)*, 573–582.
- BEN-EZRA, M., AND NAYAR, S. K. 2004. Motion-Based Motion Deblurring. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 26, 6, 689–698.
- BIGGS, D., AND ANDREWS, M. 1997. Acceleration of iterative image restoration algorithms. *Applied Optics* 36, 8, 1766–1775.
- CANON INC., 2006. What is optical image stabilizer? <http://www.canon.com/bctv/faq/optis.html>.
- CARON, J., NAMAZI, N., AND ROLLINS, C. 2002. Noniterative blind data restoration by use of an extracted filter function. *Applied Optics* 41, 32 (November), 68–84.
- FIELD, D. 1994. What is the goal of sensory coding? *Neural Computation* 6, 559–601.
- GEMAN, D., AND REYNOLDS, G. 1992. Constrained restoration and the recovery of discontinuities. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14, 3, 367–383.
- GULL, S. 1998. Bayesian inductive inference and maximum entropy. In *Maximum Entropy and Bayesian Methods*, J. Skilling, Ed. Kluwer, 54–71.
- JALOBEANU, A., BLANC-FRAUD, L., AND ZERUBIA, J. 2002. Estimation of blur and noise parameters in remote sensing. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing*.
- JANSSON, P. A. 1997. *Deconvolution of Images and Spectra*. Academic Press.
- JORDAN, M., GHAHRAMANI, Z., JAAKKOLA, T., AND SAUL, L. 1999. An introduction to variational methods for graphical models. In *Machine Learning*, vol. 37, 183–233.
- KUNDUR, D., AND HATZINAKOS, D. 1996. Blind image deconvolution. *IEEE Signal Processing Magazine* 13, 3 (May), 43–64.
- LEVIN, A., AND WEISS, Y. 2004. User Assisted Separation of Reflections from a Single Image Using a Sparsity Prior. In *ICCV*, vol. 1, 602–613.
- LEVIN, A., ZOMET, A., AND WEISS, Y. 2003. Learning How to Inpaint from Global

Appendix A

Here we give pseudo code for the algorithm, Image Deblur. This calls the inference routine, Inference, adapted from Miskin and MacKay [2000a; 2000]. For brevity, only the key steps are detailed. Matlab notation is used. The Matlab functions `imresize`, `edgetaper` and `deconvlucy` are used with their standard syntax.

Algorithm 1 Image Deblur

Require: Blurry image \mathbf{B} ; selected sub-window \mathbf{P} ; maximum blur size ϕ ; overall blur direction o ($= 0$ for horiz., $= 1$ for vert.); parameters for prior on $\nabla\mathbf{L}$: $\theta_L = \{\pi_c^s, \nu_c^s\}$; parameters for prior on \mathbf{K} : $\theta_K = \{\pi_d, \lambda_d\}$.
Convert \mathbf{P} to grayscale.
Inverse gamma correct \mathbf{P} (default $\gamma = 2.2$).
 $\nabla\mathbf{P}_x = \mathbf{P} \otimes [1, -1]$. % Compute gradients in x
 $\nabla\mathbf{P}_y = \mathbf{P} \otimes [1, -1]^T$. % Compute gradients in y
 $\nabla\mathbf{P} = [\nabla\mathbf{P}_x, \nabla\mathbf{P}_y]$. % Concatenate gradients
 $S = \lceil -2 \log_2 (3/\phi) \rceil$. % # of scales, starting with 3×3 kernel
for $s = 1$ to S **do** % Loop over scales, starting at coarsest
 $\nabla\mathbf{P}^s = \text{imresize}(\nabla\mathbf{P}, (\frac{1}{\sqrt{2}})^{S-s}, \text{'bilinear'})$. % Rescale gradients
 if ($s==1$) **then** % Initial kernel and gradients
 $\mathbf{K}^s = [0, 0, 0; 1, 1, 1; 0, 0, 0]/3$. If ($o == 1$), $\mathbf{K}^s = (\mathbf{K}^s)^T$.
 $[\mathbf{K}^s, \nabla\mathbf{L}_p^s] = \text{Inference}(\nabla\mathbf{P}^s, \mathbf{K}^s, \nabla\mathbf{P}^s, \theta_K^s, \theta_L^s)$, keeping \mathbf{K}^s fixed.
 else % Upsample estimates from previous scale
 $\nabla\mathbf{L}_p^s = \text{imresize}(\nabla\mathbf{L}_p^{s-1}, \sqrt{2}, \text{'bilinear'})$.
 $\mathbf{K}^s = \text{imresize}(\mathbf{K}^{s-1}, \sqrt{2}, \text{'bilinear'})$.
 end if
 $[\mathbf{K}^s, \nabla\mathbf{L}_p^s] = \text{Inference}(\nabla\mathbf{P}^s, \mathbf{K}^s, \nabla\mathbf{L}_p^s, \theta_K^s, \theta_L^s)$. % Run inference
end for
Set elements of \mathbf{K}^S that are less than $\max(\mathbf{K}^S)/15$ to zero. % Threshold kernel
 $\mathbf{B} = \text{edgetaper}(\mathbf{B}, \mathbf{K}^S)$. % Reduce edge ringing
 $\mathbf{L} = \text{deconvlucy}(\mathbf{B}, \mathbf{K}^S, 10)$. % Run RL for 10 iterations
Gamma correct \mathbf{L} (default $\gamma = 2.2$).
Histogram match \mathbf{L} to \mathbf{B} using `histeq`.
Output: \mathbf{L}, \mathbf{K}^S .

Algorithm 2 Inference (simplified from Miskin and MacKay [2000])

Require: Observed blurry gradients $\nabla\mathbf{P}$; initial blur kernel \mathbf{K} ; initial latent gradients $\nabla\mathbf{L}_p$; kernel prior parameters θ_K ; latent gradient prior θ_L .
% Initialize $q(\mathbf{K})$, $q(\nabla\mathbf{L}_p)$ and $q(\sigma^{-2})$
For all $i = 1, \dots, N$:
 $\mathbf{K}^i = \mathbf{K} + \text{randn}(\text{size}(\mathbf{K})) \cdot \sqrt{\lambda_d}$
 $\nabla\mathbf{L}_p^i = \nabla\mathbf{L}_p + \text{randn}(\text{size}(\nabla\mathbf{L}_p)) \cdot \sqrt{\pi_d}$
 $\sigma^2 = 1 + \text{randn}() \cdot 10^4$

Other Writing Tips (I)

- Good resource on mathematical writing:
“Mathematical Writing”, Donald E. Knuth, Tracy Larrabee and Paul M. Roberts, 1989
- Several specific stylistic (mathematical/prose) elements listed
 - Use consistent notation for the same thing when it appears in several places.
 - Don’t just list a sequence of formulas. Tie concepts together with a running commentary.
 - Don’t be too generous with using notation. Sometimes prose is better.

Other Writing Tips (II)

- Avoid zig-zagging
 - Minimize the number of subject changes. Otherwise, it can be distracting to the reader.
 - Only the central idea(s) should appear multiple times in the paper.
- Bulleted lists are your friend, but don't go overboard with their use!
- Use examples to illustrate your model/algorithm, especially if it has many moving parts.

Useful tools to write a paper in CS

- **LaTeX**: Powerful document processor. Superior aesthetics in print. Very good at typesetting equations.
- Good beginner's tutorial: <http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf>
- Online, real-time collaborative LaTeX tools like overleaf.com and sharelatex.com are popular
- Compile list of references using a reference management software like **BibTeX**.

LaTeX tips

- Pay attention to typesetting
 - E.g., make sure mathematical symbols are typeset in mathematical fonts even within text

Let n be an integer
 - Use macros to make your formulas easier to write (and rewrite) consistently
- ```
\newcommand{\G}{\ensuremath{\mathcal{G}}\xspace}
```

  
...  
**Let  $\mathcal{G}$  be a class**

<sup>2</sup>Let  $\mathcal{G}$  be a class
- Use appropriate packages
- A good online resource: <https://en.wikibooks.org/wiki/LaTeX>

# Reviewing Process

---

- Anonymous peer review
- In conferences: Reviewing is either blind (only reviewers are anonymous) or double-blind (both authors and reviewers are anonymous)
- 70-80% of papers are rejected at top-tier conferences
  - A poorly written paper almost never makes the cut

# Where should I submit my work?

---

- Important to submit to the right venue. Consult with your advisor.
- You should have read dozens of papers submitted to a venue to understand it better.
  - What does the audience expect?
  - What are the conventions adopted here?
  - What is their process of selection?

# Deciding the author list

---

- Who are the authors? Everyone who made a significant contribution.
- Author order:
  - Some communities (TCS) use alphabetical order
  - Some communities use descending order of contribution

# Rewriting: Lather, Rinse, Repeat

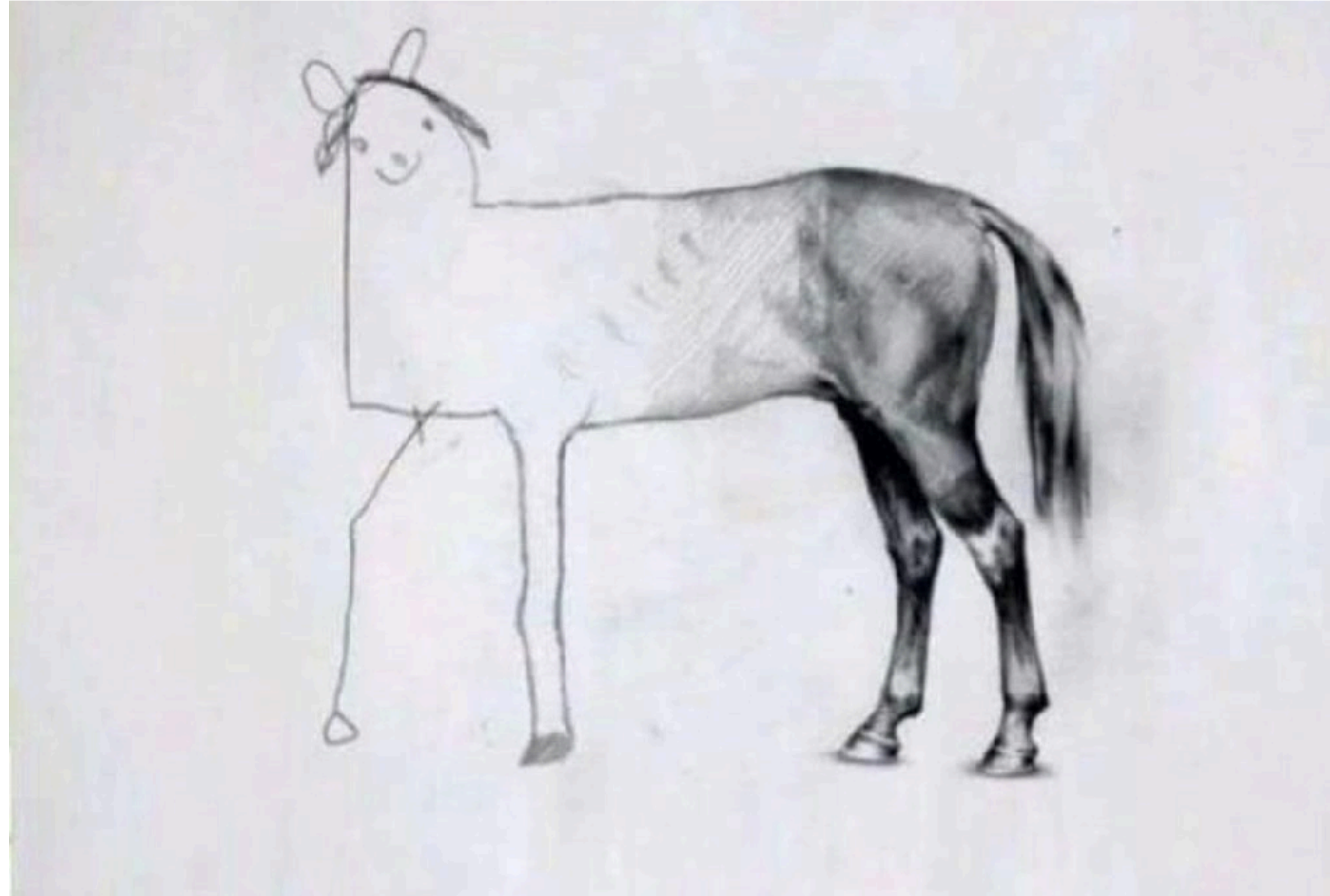
---

- Writing is an iterative process
  - Don't get too attached to your words. Rewriting typically produces better text.
  - To allow time for rewrites, start writing your paper early!
- Spend time “debugging” your paper. Use a spell-checker, if needed. Otherwise, very tedious for the reviewer.



# Unfortunately, deadlines loom...

---



- Tip: Spend more time on the important parts of the paper (introduction, overall organisation, key technical aspects)

# Final Takeaways

---

- The reader is the boss. Always keep your reader in mind when structuring your paper.
- Spend time planning your paper. Organisation is key. Let good work not sink due to poor writing.
- Rewriting is good writing. Your words are very likely to improve over multiple iterations.
- The old adage “practice makes perfect” holds value when it comes to technical writing.