Image Smoothing Using Spatial Iterative Methods Based on Accelerated Iterative Shrinkage

Dabwitso Kasauka†, Hiroshi Tsutsui†, Hiroyuki Okuhata*, Takashi Imagawa†, and Yoshikazu Miyanaga†
†Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan
*Synthesis Corporation
Awajimachi TC Bldg. 4th Floor, Awajimachi 2-6-9, Chuo-ku, Osaka 541-0047, Japan

***Abstract*—In recent years, much research interest has developed in image smoothing techniques. With increasing application in various fields, there is a motivation to explore various modes of algorithm implementation of image smoothing. Recently, edge-aware image smoothing techniques have been developed based on fast Fourier transformation methods. In this paper, we present an alternative implementation for an existing image smoothing algorithm using spatial iterative methods. The motivation of this is to create a performance baseline using spatial iterative methods such as *multigrid (MG)*, *conjugate gradient (CG)*, and *preconditioned conjugate gradient (PCG) methods*, for the purpose that the algorithm can be easily adapted to parallel computing systems. We also determine the competitiveness compared with FFT implementation in terms of computational cost. From experimental results, multigrid preconditioned conjugate gradient (MGCG) method provides superior results both in smoothing quality and computational cost compared to all the spatial iterative methods considered. Furthermore, with relaxed tolerance, it demonstrates lower computational complexity compared with FFT implementation, with similar smoothing results but having minor quality compromise. Hence, MGCG provides a relatively competitive spatial domain alternative to frequency domain solver, FFT. In applications which do not require computation of an exact solution, spatial iterative methods can provide a reasonable computation alternative to FFT implementation as their convergence conditions can easily be altered by the user to fit a specific application, as well as possessing the ease for parallel computing adaptation.**

I. Introduction

In recent years, great emphasis has been placed on the development of least computational costly image smoothing algorithms, possessing edge-aware qualities. Presently, with the advent of high processing power machines, processing time will soon become the least of concern for image smoothing technique developers. On the other hand, massively parallel computing systems have been developed, which has lead to gained interest in spatial iterative methods [1].

An edge-aware/preserving image smoothing algorithm basically limits fine details while maintaining the structural integrity of the overall image i.e. returning the fidelity of the edges. Edge preserving can be achieved via several techniques including energy-minimization, surface fitting, and weighted averaging [2]. According to [3, 4], edge-aware smoothing is a fundamental building block for several applications. As illustrated in [2, 5]–[7], some of these applications are in detail enhancement, high dynamic range (HDR) tone mapping, edge enhancement and extraction, image abstraction and pencil sketching, clip-art compression artifact removal, noise removal, and layer-based contrast manipulation, just to highlight a few. According to [8], smoothing can be combined with segmentation techniques in order to produce superior segmentation results.

Image smoothing involves solving a formulation of a Poisson equation. An FFT solver strives to solve an exact solution, and furthermore can be considered as a black box [1]. However, iterative methods offer more user control flexibility as they can be stopped at any point even before reaching an exact solution, once the pre-specified convergence criteria are achieved. In light of this, even though iterative methods may generally suffer from higher computational cost as compared to FFT solvers, they may find use in applications which may not require the calculation of an exact solution, or favor iterative methods to FFT for ease of parallel computing adaptation based on a specific hardware application.

Having performed a computational cost breakdown of [6] in [9], we also desire to investigate how competitive iterative methods are to the existing FFT implementation. As such, by our work we seek to create a baseline for future works on the comparative pros and cons of iterative method implementation to FFT, in image smoothing.

In this paper, we demonstrate the implementation of image smoothing based on [6] using iterative methods. The algorithm can be represented in two parts: (1) calculation of the shrinkage operation based on [6], (2) implementation of smoothing (solving screened Poisson equation) using iterative methods.

This paper is organized as follows. In Section II, we present the formulation of the problem to be solved iteratively. In Section III, we present the experimental results and analysis. Finally in Section IV, we conclude this paper.

II. Problem Formulation

Given that a smoothing problem can be represented by two procedures [6],

$$v^{(k+1)} \leftarrow \underset{v}{\operatorname{argmin}}\, \psi(v) + \frac{\beta}{2}\left\|\nabla u^{(k)} - v\right\|_2^2, \quad (1)$$

$$u^{(k+1)} \leftarrow \underset{u}{\operatorname{argmin}}\, \lambda\left\|u - g\right\|_2^2 + \beta\left\|\nabla u - v^{(k+1)}\right\|_2^2, \quad (2)$$

where g is input image, u is the desired smoothed output, ∇u is the gradient of u, $\psi(v)$ is a model function, and v is a

gradient field. Equation 1 corresponds to a shrinkage operation while Eq. 2 corresponds to the screened Poisson equation [10].

We are interested in solving the screened Poisson equation using iterative methods. Sparse iterative linear solvers are used to solve linear problems of the form of $Ax = b$. Hence, it is first required to formulate the screened Poisson equation into the form of $Ax = b$.

The minimization problem expressed by the screened Poisson equation can be likened to minimizing a quadratic function, with the solution u occurring at minimum point (zero gradient location). Henceforth, for simplicity of manipulation, we can represent the screened Poisson equation as:

$$L = \lambda \left\| u - g \right\|_2^2 + \beta \left\| \nabla u - v \right\|_2^2 . \quad (3)$$

According to [10], the equation above can be simplified into

$$L = \lambda \left(u - g \right)^2 + \beta \left((u_x - v^x)^2 + (u_y - v^y)^2 \right)^2 . \quad (4)$$

Deriving the derivative of the equation above, based on the satisfaction of the Euler-Lagrange equation, and equating to zero yields,

$$\lambda u - \beta(u_{xx} + u_{yy}) = \lambda g - \beta(v_x^x + v_y^y), \quad (5)$$

which can be further simplified into,

$$(\lambda I - \beta \nabla^2) u = -\beta \nabla . v + \lambda g, \quad (6)$$

where ∇^2 is the discrete Laplacian and I is an identity matrix.

Note the equation above is basically a representation of the form of $Ax = b$, with:

$$A = \lambda I - \beta \nabla^2, \quad (7)$$

$$x = u, \quad (8)$$

$$b = -\beta \nabla . v + \lambda g. \quad (9)$$

In order to solve Eq. 6 using iterative methods, it is required to only have one unknown variable, which in this case is the desired smoothed output u. Note that λ and β are user defined scalar smoothing parameters, ∇^2 is a known parameter (simply modeled after a 5 point discrete Laplacian kernel), and g is the known input image. In order to solve $Ax = b$, b is required to be a known parameter. Hence to have it so, the gradient field v in Eq. 6 has to be calculated first by Eq. 1 before we can proceed with iterative method implementation. This is calculated based on accelerated shrinkage operation presented in [6]. In summary, our proposed algorithm is given in Alg. 1.

As we are dealing with spatial iterative methods, the parameter common to all is ε. This parameter is a convergence condition which determines the accuracy of the method.

III. Results and Analysis

All experiments shown in this section were performed on Intel Core i7 CPU @3.4 MHz, and the implementation was done in MATLAB R2013a. We solved Eq. 6 using multigrid method and conjugate gradient method (with their preconditioned variants) in the following subsections, as they present superior processing attributes when dealing with very large number of unknowns as opposed to other iterative methods such as Jacobi or Gauss-Seidel methods. Refer to Alg. 1, in order to clearly identify where these iterative methods were applied.

Algorithm 1 Proposed Smoothing Implementation

Data: Input image g, parameters λ, β and convergence stopping criteria ε.
Result: Smoothed image u.
Step 1: Calculate gradient field v, use it to compute $b = -\beta \nabla . v + \lambda g$ and store.
Step 2: Calculate $A = \lambda I - \beta \nabla^2$ and store it.
Step 3:
while (!convergence) **do**
  Solve $Ax = b$ using iterative method.
end while
Solution $u = x$

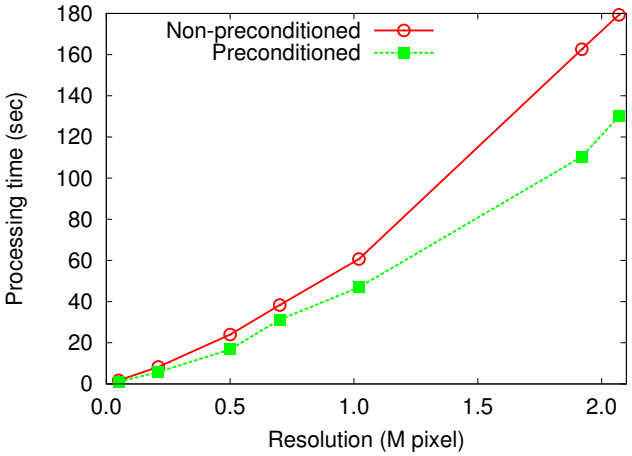

Fig. 1: Result of algebraic multigrid method (AMG). Resolution vs processing time.

A. Algebraic Multigrid

Algebraic multigrid method (AMG) is a general form of multigrid, which solely derives required information from the coefficient matrix A, as opposed to geometric multigrid. We present experimental results based on v-cycle implementation for both non-preconditioned and preconditioned algebraic multigrid.

To solve Eq. 6, various parameter tweaks were applied. These were variation of convergence stopping criterion referred as *tolerance*, optimization of *grid number* and application of a preconditioner. Smoothing parameters β and λ were set experimentally to 1 and 0.02, respectively.

Figure 1 shows a result obtained with a tolerance of $\varepsilon = 10^{-6}$, for various input image resolutions for both preconditioned and non-preconditioned AMG. By introducing a preconditioner, which in this case was CG method, it can be observed that the convergence time is improved from 179 sec to 130 sec for a full HD image. This is due to the fact that introducing a preconditioner basically improves the condition number of the coefficient matrix A in Eq. 6, thereby

TABLE I: Preconditioned AMG coarsening level vs processing time (tolerance: $\varepsilon = 10^{-6}$, image size: 256×256).

Coarsest	Number of grids	Processing time (sec)
128×128	2	1.78
64×64	3	1.26
32×32	4	0.76
16×16	5	0.69
8×8	6	0.66
4×4	7	0.66
2×2	8	0.66

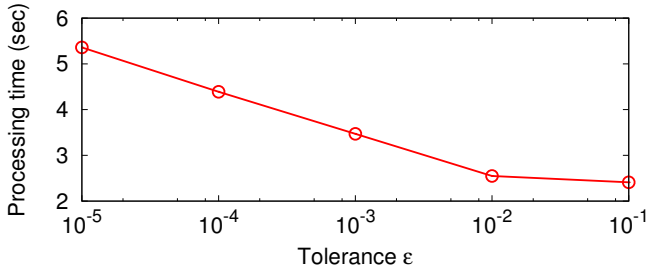

Fig. 2: AMG: Tolerance vs processing time.

causing the eigenvalues to cluster around 1 and hence resulting in a relatively well-conditioned problem. A well-conditioned problem converges faster than an ill-conditioned problem.

As we are dealing with a multigrid method, for further optimization we ran test to determine the optimal number of grids necessary to ensure efficient and fast processing. Our test image was a 256×256 RGB image. Table I illustrates the optimal number of grids and the associated processing time. From this, we notice that only 6 grids are necessary with an optimal processing time of 0.66 sec. For various resolutions, the required number of grids was generally the same with minor deviation, with the general optimal number determined to be 6.

In most image smoothing applications, in which minor details are not a priority but the fidelity of the edges, we can relax the tolerance condition in a bid to further optimize this iterative method. Figure 2 depict the effect of tolerance on convergence rate. With $\lambda = 0.02$, it was noted that over-relaxed tolerance levels of 10^{-2} and 10^{-1} impacted the smoothing quality of higher resolution input images (i.e. from about 1.7 megapixels upwards) more than it did for lower resolution images. By changing the value of λ to 0.01, the smoothing quality in the case for high resolution images was greatly improved at such relaxed tolerance values.

TABLE II: Optimized AMG for smoothing application

Resolution (Mpixel)	Non-optimized (sec)	Optimized (sec)
0.05	1.8	0.45
0.21	9.0	1.62
0.50	22.5	4.02
0.70	39.6	5.55
1.02	64.8	8.22
1.92	142.3	15.88
2.07	185.7	17.18

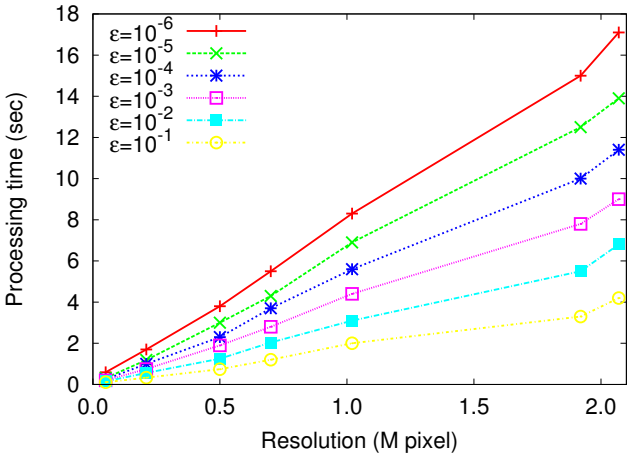

Fig. 3: Result of conjugate gradient method (CG). Resolution vs time.

Finally by a combined optimization of the three parameters described previously within this subsection, we obtained results given in Table II. Clearly, convergence rate is greatly improved, achieving smoothing within 17.18 sec as opposed to the initial 185.7 sec when considering a color full HD image (1920×1080). Hence, convergence within 17.18 sec is what is the general computation expectation when we implement image smoothing using multigrid method.

B. Conjugate Gradient Method

Conjugate gradient method (CG) is a Krylov space iterative method. There are many variants, some of which are application specific. We performed image smoothing (solving Eq. 6) using conjugate gradient method (CG), incomplete Cholesky factorization preconditioned conjugate gradient (ICCG), and multigrid preconditioned conjugate gradient method (MGCG). The smoothing parameters were set to $\lambda = 0.02$ and $\beta = 1$.

Figure 3 shows the performance of non-preconditioned conjugate gradient method for various resolutions and tolerance values. As can be noted from this result, computational performance of conjugate gradient method even under strict tolerance criterion is superior to that of multigrid method. For 1920×1080 resolution image smoothing, a tolerance of $\varepsilon = 10^{-3}$ was sufficient to obtain a reasonably good smoothing quality result within 9 sec. Further we introduced preconditioning by applying incomplete Cholesky factorization (IC) and multigrid (MG), independently. It should be noted that the complexity of computing and applying a preconditioner is vital in the selection of a preconditioner as this also affects the overall processing time. Hence, applying a highly robust but very complex preconditioner would defeat the purpose of preconditioning to reduce the overall computation time.

Figure 4 shows the performance when IC factorization of coefficient matrix A (in Eq. 6) is used as a preconditioner. As expected of preconditioning, there was a computation time improvement compared to the non-preconditioned conjugate gradient method resulting in time reduction from 17.1 to

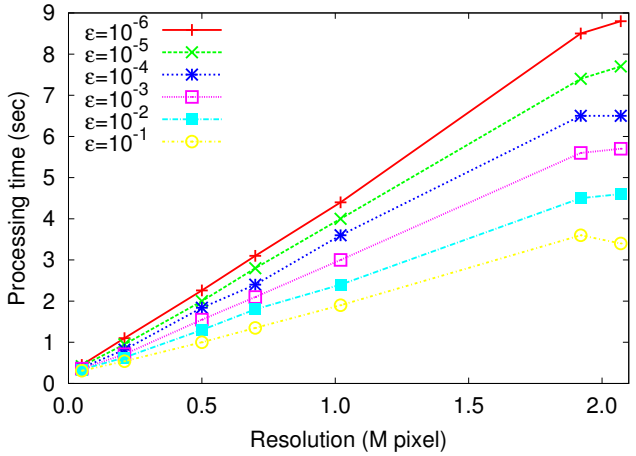

Fig. 4: Result of incomplete Cholesky factorization preconditioned conjugate gradient method (ICCG). Resolution vs processing time.

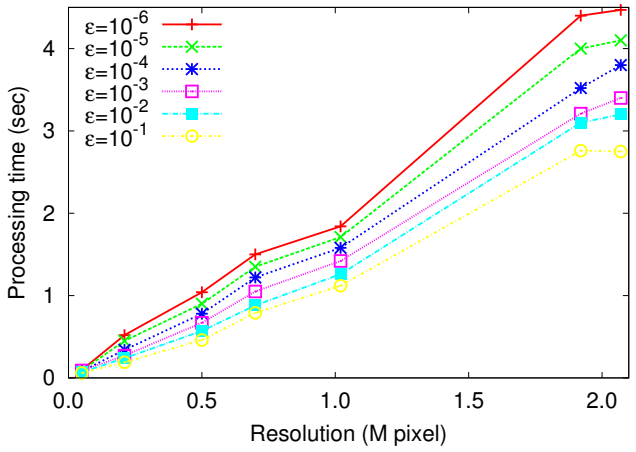

Fig. 5: Result of multigrid preconditioned conjugate gradient method (MGCG). Resolution vs processing time.

8.8 secs for a full HD image with $\varepsilon = 10^{-6}$. In this case of ICCG, computing an incomplete Cholesky preconditioner proved to be reasonable both in terms of computation and storage (intrinsically). This is so due to the fact that this preconditioner is very sparse (as coefficient matrix A is sparse), while also being easier to apply than A as the computation of its inverse is easier. From Fig. 4, it is shown that IC factorization preconditioning effectively improved the overall processing performance of CG method.

In the case of applying multigrid as a preconditioner, the computation performance results for various image resolutions and at different tolerance are shown in Fig. 5. Smoothing convergence occurred within 4.4 sec for a full HD image with $\varepsilon = 10^{-6}$. In MGCG, multigrid method as a preconditioner basically aids the removal of low frequency errors ("smooth error") on the problem grid (actual problem size) by means of relatively converting them into high frequency errors ("rough error") on a coarse grid, which can then be easily removed. Hence, multigrid preconditioner relaxes the problem being solved iteratively by CG, thereby leading to faster convergence rate. Applying multigrid as a preconditioner is computationally reasonable as the error removal computation is performed on a smaller (coarse grid) problem size.

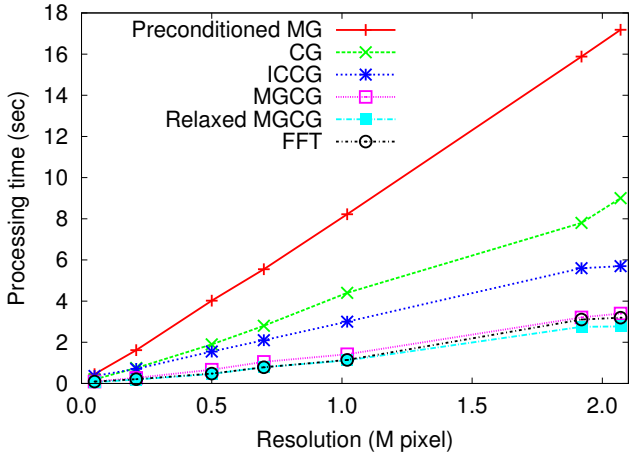

Fig. 6: Performance comparison of iterative methods considered in this paper as well as the existing FFT implementation.

From this, MGCG provided the best computational performance results compared with the other spatial iterative solvers considered in this paper. Generally, concerning spatial iterative methods investigated in this paper, it is noted that the trade-off between processing time and output quality is greatly influenced by the value of ε. Hence, a strict ε (e.g. $\varepsilon = 10^{-6}$) provide better smoothing quality but at a higher processing time compared with more relaxed ε (e.g. $\varepsilon = 10^{-2}$).

C. Performance Comparison

Even though our main objective is to present a spatial iterative solver approach to an existing image smoothing algorithm thereby also providing a performance baseline, it is also necessary to compare the overall performance for such iterative methods to the existing FFT solver approach.

Figure 6 illustrates the computation cost of both iterative and FFT solvers. For the spatial iterative solvers, the outlined results were taken at tolerance of $\varepsilon = 10^{-3}$, yielding acceptably good smoothing quality results. It can be noted that FFT and MGCG at the above specified tolerance (ε does not apply to FFT) were very close in computational performance, with FFT slightly performing faster than MGCG under the specified tolerance. Furthermore, by setting $\lambda = 0.01$ and relaxing the convergence tolerance, MGCG converged within 3.2 sec at a tolerance of $\varepsilon = 10^{-2}$, with FFT converging within 3.1 sec. Further tolerance relaxing leads to MGCG converging within 2.8 sec, while FFT still in 3.1 sec. Figure 7 (a), (b), and (c) shows the input image, and smoothing results using relaxed MGCG and FFT solver, respectively. We evaluated the quality difference between the two methods using PSNR, yielding 48.29 dB in the case of Fig. 7. By this, we can loosely say that

(a) Input.

(b) Relaxed MGCG smoothed output.

(c) FFT smoothed output.

Fig. 7: Image smoothing retults. The input image is obtained from `publicdomainpictures.net` (image 8363).

MGCG performed faster while maintaining reasonably good smoothing quality performance compared to the existing FFT solver implementation.

IV. Conclusion

In conclusion, it has been successfully demonstrated that image smoothing can be implemented using spatial iterative methods, while also comparing the performance of the iterative methods considered in this paper with each other as well as with the existing FFT implementation. Spatial iterative methods have been proven that they can also serve as good solvers in image smoothing applications, especially in cases where relaxed convergence conditions are sufficient. We have also provided a performance comparison for iterative methods, clearly depicting that MGCG provides the best computational performance results among all the other spatial iterative methods considered in this paper for the purpose of image smoothing. When compared to FFT, under moderate convergence conditions, MGCG performed slightly slower within 3.4 sec compared to FFT which converged within 3.1 sec. Nonetheless, by relaxing the convergence criteria for MGCG, it converged within 2.8 sec. Hence, the iterative method provided lower computational complexity with some quality compromise. In addition, iterative solvers provide users greater degree of flexibility, as various components which affect processing time can be tweaked to achieve an application specific desired outcome.

References

[1] T. Guillet and R. Teyssier, "A simple multigrid scheme for solving the poisson equation with arbitrary domain boundaries," *J. Comput. Phys.*, vol. 230, no. 12, pp. 4756–4771, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2011.02.044

[2] Z. He, Y. Zhang, R. Su, and S. Fan, "A novel image smoothing algorithm based on variational decomposition," in *Proceedings of International Conference on Genetic and Evolutionary Computing*, Dec. 2010, pp. 181–185. [Online]. Available: http://dx.doi.org/10.1109/ICGEC.2010.52

[3] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," in *Proceedings of SIGGRAPH 2006*. New York, NY, USA: ACM, 2006, pp. 646–653. [Online]. Available: http://doi.acm.org/10.1145/1179352.1141936

[4] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 22:1–22:10, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531328

[5] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 174:1–174:12, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024208

[6] H. Badri, H. Yahia, and D. Aboutajdine, "Fast multi-scale detail decomposition via accelerated iterative shrinkage," in *SIGGRAPH Asia 2013 Technical Briefs*, ser. SA '13. New York, NY, USA: ACM, 2013, pp. 33:1–33:4. [Online]. Available: http://doi.acm.org/10.1145/2542355.2542397

[7] G. Tanaka, N. Suetake, and E. Uchino, "Image enhancement based on nonlinear smoothing and sharpening for noisy images," *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, vol. 14, no. 2, pp. 200–207, 2010.

[8] D. G. Lowe, "Organization of smooth image curves at multiple scales," *International Journal of Computer Vision*, vol. 3, no. 2, pp. 119–130, 1989.

[9] D. Kasauka, H. Tsutsui, H. Okuhata, and Y. Miyanaga, "Computational cost analysis and implementation of accelerated iterative shrinkage smoothing," in *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, Dec 2014, pp. 1–4.

[10] P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick, "Fourier analysis of the 2D screened Poisson equation for gradient domain problems," in *Proceedings of European Conference on Computer Vision: Part II*, Oct. 2008, pp. 114–128. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88688-4_9