

# An Algorithm for Minimizing the Mumford-Shah Functional

Michael Bauer

Technische Universität München  
Department of Informatics  
Computer Vision Group

October 5, 2015

# Outline

- 1 Related Work
- 2 Primal-Dual Algorithm
- 3 Demo

# Outline

- 1 Related Work
- 2 Primal-Dual Algorithm
- 3 Demo

## Related work and further references

- T. Pock and D. Cremers and H. Bischof and A. Chambolle, An Algorithm for Minimizing the Piecewise Smooth Mumford-Shah Functional, ICCV, 2009.
- <https://www.github.com/BauerMichael>
- <http://www.mik-e.com>
- Tuesday, 20th October, 10 a.m., Department of Mathematics, University of Regensburg

# Outline

- 1 Related Work
- 2 Primal-Dual Algorithm
- 3 Demo

# Primal-Dual Algorithm

## Algorithm

Choose  $(x^0, y^0) \in C \times K$  and let  $\bar{x}^0 = x^0$ . We choose  $\tau, \sigma > 0$ .  
Then, we let for each  $n \geq 0$

$$\begin{cases} y^{n+1} = \Pi_K(y^n + \sigma A \bar{x}^n) \\ x^{n+1} = \Pi_C(x^n - \tau A^* y^{n+1}) \\ \bar{x}^{n+1} = 2x^{n+1} - x^n. \end{cases}$$

## The Projection onto $C$

$$C = \{x \in X : x(i, j, k) \in [0, 1], x(i, j, 1) = 1, x(i, j, M) = 0\} \subseteq X.$$

### Algorithm (Clipping)

$$x^{n+1} = \min\{1, \max\{0, x^n\}\}.$$

## The Projection onto $K$

$$K = \left\{ y = (y^1, y^2, y^3)^T \in Y : \right.$$

$$y^3(i, j, k) \geq \frac{y^1(i, j, k)^2 + y^2(i, j, k)^2}{4} - \lambda \left( \frac{k}{M} - f(i, j) \right)^2, \quad (1)$$

$$\left. \left| \sum_{k_1 \leq k \leq k_2} (y^1(i, j, k), y^2(i, j, k))^T \right| \leq \nu \right\} \quad (2)$$



## Boyle-Dykstra Algorithm

### Algorithm ([pami11])

Choose  $u_i^k, v_i^k$  and initialize  $u_p^0 = u_{cur}$  and  $v_i^0 = 0$  for all  $i = 1, 2, \dots, p$ .

$$\begin{aligned}u_0^k &= u_p^{k-1}, \\u_i^k &= \Pi_i(u_{i-1}^k - v_i^{k-1}), \quad i = 1, 2, \dots, p, \\v_i^k &= u_i^i - (u_{i-1}^k - v_i^{k-1}), \quad i = 1, 2, \dots, p.\end{aligned}$$

# Minimization with Lagrange-Multipliers

## Algorithm

Choose  $(x^0, y^0, \lambda^0, p^0) \in C \times K_p \times \mathbb{R}^{2 \times N \times N \times M} \times \mathbb{R}^{2 \times N \times N \times M}$  and let  $\bar{x}^0 = x^0, \bar{\lambda}^0 = \lambda^0$ . We choose

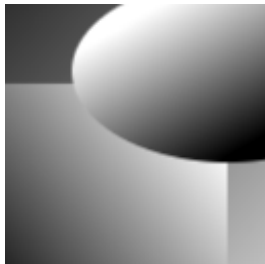
$\tau_x = \frac{1}{6}, \tau_\lambda = \frac{1}{2+k_2-k_1}, \sigma_y = \frac{1}{3+M}, \sigma_p = 1$ . Then, we let for each  $n \geq 0$

$$\left\{ \begin{array}{l} y^{n+1} = \Pi_{K_p}(y^n + \sigma_y(A\bar{x}^n + \tilde{y})) \\ p_{k_1, k_2}^{n+1} = \Pi_{\|\cdot\|_2 \leq \nu}(p_{k_1, k_2}^n + \sigma_p \bar{\lambda}_{k_1, k_2}^n) \\ x^{n+1} = \Pi_C(x^n - \tau_x A^* y^{n+1}) \\ \lambda_{k_1, k_2}^{n+1} = \lambda_{k_1, k_2}^n - \tau_\lambda (p_{k_1, k_2}^{n+1} - \sum_{k_1 \leq k \leq k_2} (y_1(i, j, k), y_2(i, j, k))^T) \\ \bar{x}^{n+1} = 2x^{n+1} - x^n \\ \bar{\lambda}_{k_1, k_2}^{n+1} = 2\lambda_{k_1, k_2}^{n+1} - \lambda_{k_1, k_2}^n \end{array} \right.$$

# Outline

- 1 Related Work
- 2 Primal-Dual Algorithm
- 3 Demo

## Synthetic Image (Size: 128 x 128)



*Synthetic Image*  
Size: 128 x 128  
grayscale

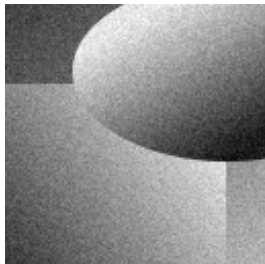
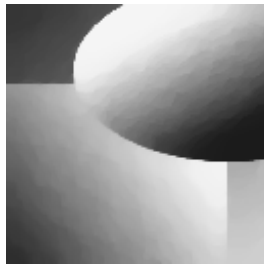


*Boyle-Dysktra*  
Level: 16  
Memory Used: 1.33 GB  
Runtime: 4505 s  
Iterations: 1000



*Lagrange-Multipliers*  
Level: 16  
Memory Used: 0.155 GB  
Runtime: 10.39 s  
Iterations: 1000

## Synthetic Image With Gaussian Noise (Size: 128 x 128)

*Noisy Image**Size: 128 x 128**grayscale**Boyle-Dysktra**Level: 16**Memory Used: 1.33 GB**Runtime: 4495 s**Iterations: 1000**Lagrange-Multipliers**Level: 16**Memory Used: 0.155 GB**Runtime: 10.47 s**Iterations: 1000*



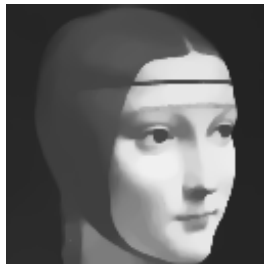
## La dama con l'ermellino (Size: 128 x 128)



*La dama Image*  
Size: 128 x 128  
grayscale



*Boyle-Dysktra*  
Level: 16  
Memory Used: 1.33 GB  
Runtime: 4495 s  
Iterations: 1000

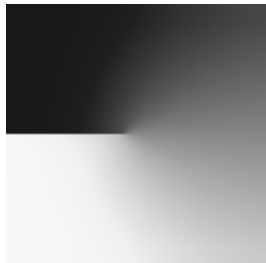


*Lagrange-Multipliers*  
Level: 16  
Memory Used: 0.155 GB  
Runtime: 10.42 s  
Iterations: 1000

## Crack Tip Inpainting (Size: 128 x 128)



*Crack Tip Problem*  
Size: 128 x 128  
grayscale



*Boyle-Dysktra*  
Level: 16  
Memory Used: 1.33 GB  
Runtime: 4501 s  
Iterations: 1000



*Lagrange-Multipliers*  
Level: 16  
Memory Used: 0.155 GB  
Runtime: 10.49 s  
Iterations: 1000

## Special Thanks To

- Prof. Dr. Daniel Cremers
- Evgeny Strelakovskiy
- Thomas Moellenhoff



## Bibliography I

- [iccv09] T. Pock and D. Cremers and H. Bischof and A. Chambolle, An Algorithm for Minimizing the Piecewise Smooth Mumford-Shah Functional, iccv, 2009.
- [pami11] D. Cremers and K. Kolev Multiview Stereo and Silhouette Consistency via Convex Functionals over Convex Domains, pami, 2011.