ugr | Universidad de Granada

Diffusion and Image Correspondences

Jarno Ralli [1]

[1]Department of Computer Architecture and Technology
University of Granada, Spain

2012

Effect of non-linear diffusion upon image

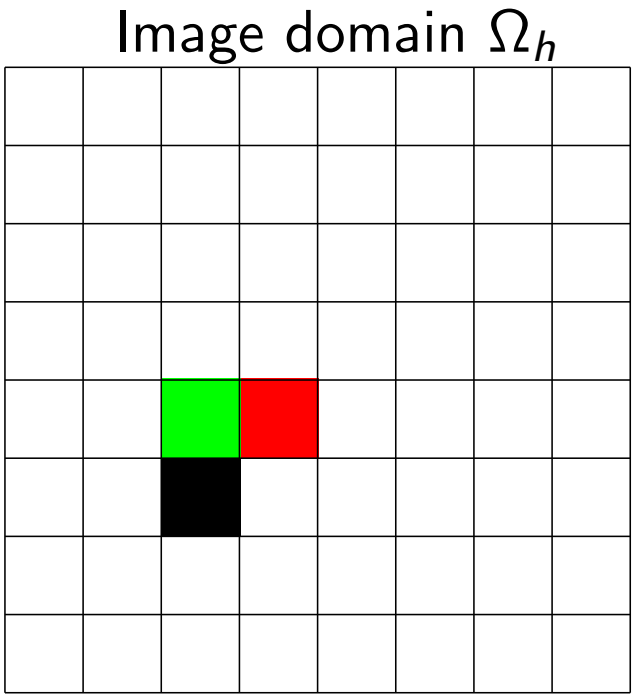
(a) Orig.

(b) $t = 0.5$.

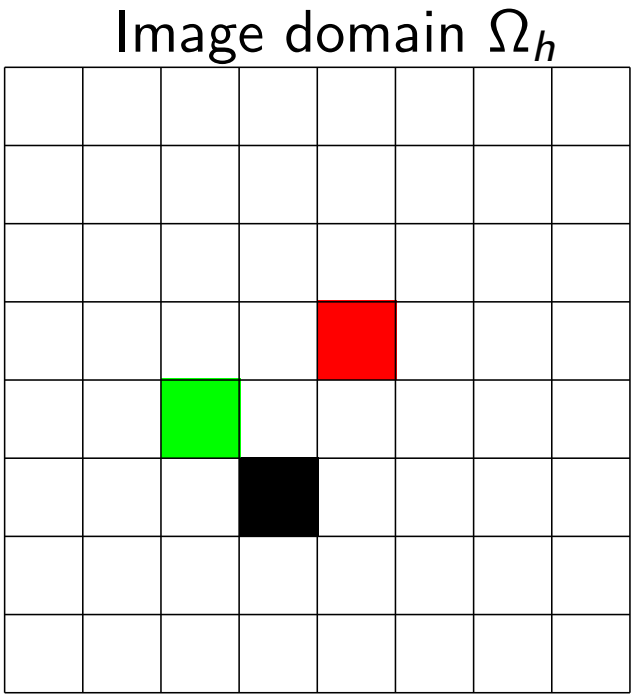
(c) $t = 1.0$.

Artificial time t defines how 'much' the image is diffused. As it can be understood, diffusion 'simplifies' the image.

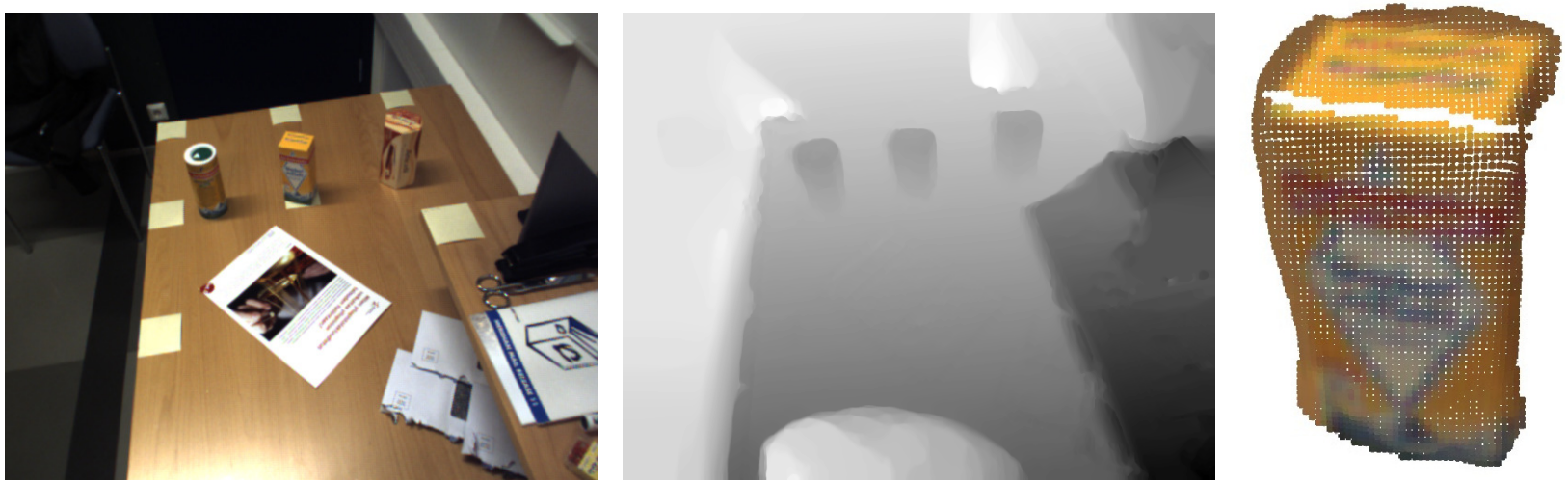
Image Correspondences



(a) First image.



(b) Second image.



(a) Left image. (b) Disparity. (c) 3D.

Figure : A robotics related disparity example. In the case of the disparity map, gray-level codifies the disparity: objects with dark tones are closer to the cameras, while objects with light tones are further away from the cameras.

Image Correspondences

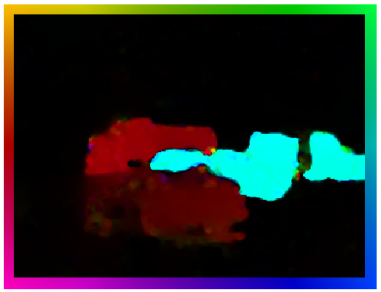
$I(x, y, k, t) : \mathbb{R} \times \mathbb{R} \times [0, \infty) \to \mathbb{R}^3$, where $k = 1 \dots 3$ is the channel (i.e. RGB).

(a) $I(x, y, k, 0)$

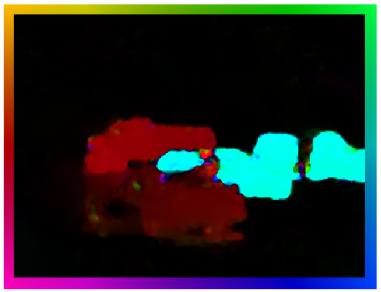
(b) $I(x, y, k, 1)$

(c) $I(x, y, k, 2)$

(d) $I(x, y, k, 3)$



(e) Flow, $0 \dots 1$



(f) Flow, $1 \dots 2$

(g) Flow, $2 \dots 3$

Contents

CONTENTS OF THE PRESENTATION

- Math primer
  - Domains, derivative, partial derivative, gradient etc.
- Diffusion
  - Linear (homogeneous) and non-linear diffusion
  - Divergence, physical interpretation and discretisation example
  - Examples
- Image correspondences
  - Optical-flow constraint (OFC) and formulation
  - Lukas-Kanade model
  - Horn-Schunck and 'robust' HK models
  - Examples

MATH PRIMER

Mathematical Notation 1/2

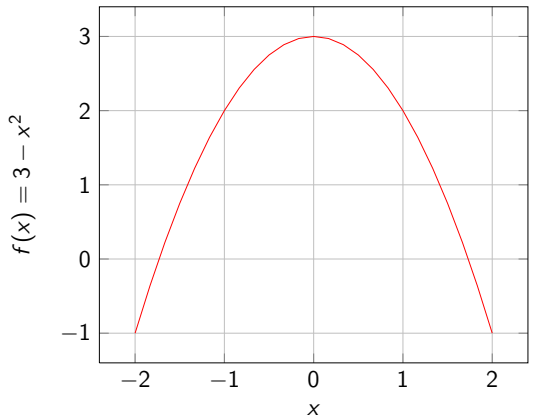
$$\underbrace{NAME}_{\text{name of the mapping}} : \underbrace{DOMAIN}_{\text{input}} \underbrace{\rightarrow}_{\text{maps into}} \underbrace{CODOMAIN}_{\text{output}}$$

- $f : \mathbb{R} \rightarrow \mathbb{R}$, for example $f(x) = 3 - x^2$
- $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, for example $f(x, y) = 3 - x^2 - y^2$
- $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^{\mathbb{N}+}$, for example $I(x, y, k)$ (i.e. image with k channels)
- $f : \mathbb{R} \times \mathbb{R} \times [0, \infty) \rightarrow \mathbb{R}^{\mathbb{N}+}$, for example $I(x, y, k, t)$

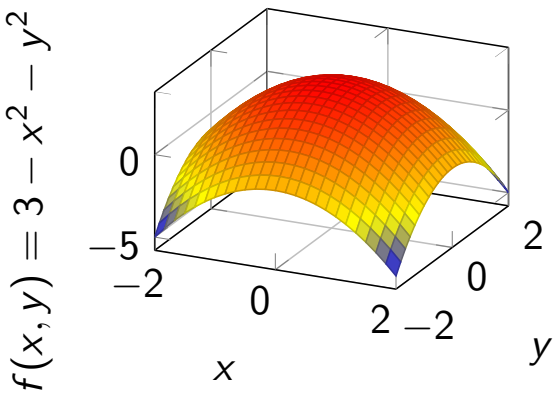
Here $\times$ means Cartesian product (i.e. $X \times Y = \{(x, y) | x \in X,\ y \in Y\}$).

DIFFERENT DOMAINS

Instead of writing $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ we can take a short cut and define the domain as $\Omega \in \mathbb{R} \times \mathbb{R}$ and write it as $f : \Omega \to \mathbb{R}$.



(a) $\Omega \in [-2 \dots 2]$

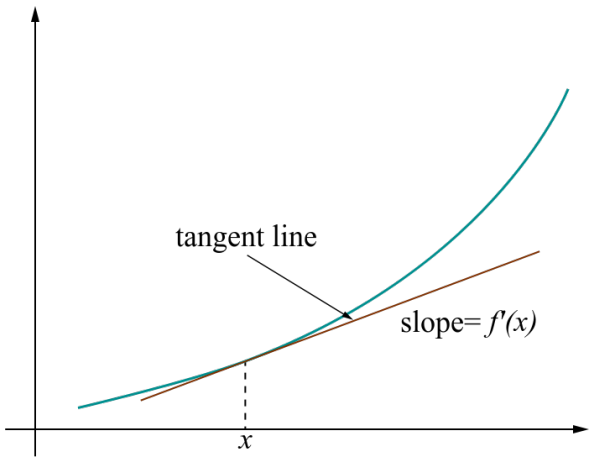


(b) $\Omega \in [-2 \dots 2] \times [-2 \dots 2]$

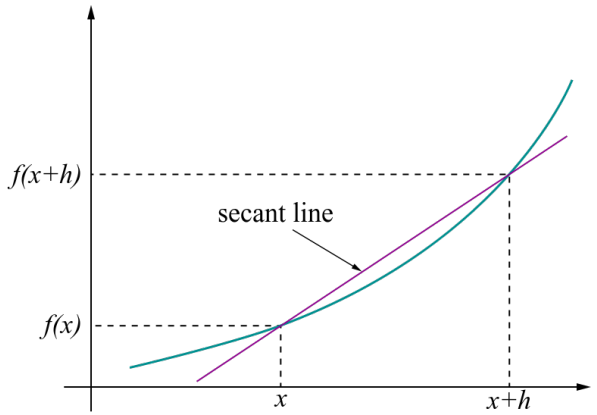
(c) $\Omega \in [1 \dots m] \times [1 \dots n]$

In calculus **derivative** is a measure how much a function changes as its input changes. Let f be a real valued function. Geometrically derivative of f at a point x is **tangent** to the graph of the function at $(x, f(x))$. Formally, the *derivative* of the function f at x is the limit:

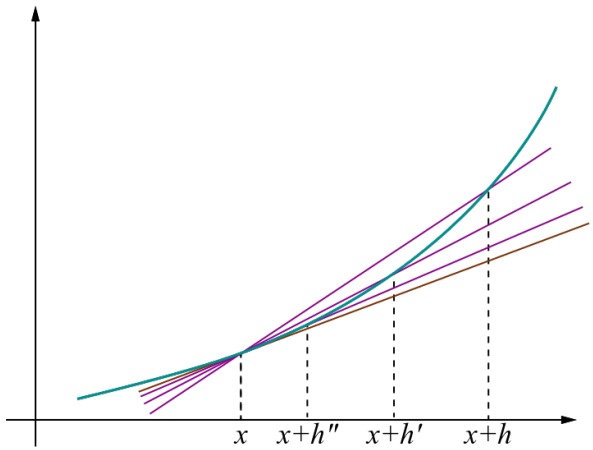
$$f(x)' = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



(a) Tangent



(b) Secant



(c) Limit of secant

NOTATIONS FOR DERIVATIVES

- Leibniz's notation
  - first order: $\frac{dy}{dx}$, $\frac{df}{dx}(x)$, $\frac{d}{dx}f(x)$
  - higher order: $\frac{d^n y}{dx^n}$, $\frac{d^n f}{dx^n}(x)$, $\frac{d^n}{dx^n}f(x)$
- Lagrange's notation
  - first order: f'
  - higher order: f'', f''', $f^{(4)}$
- Newton's notation $\left(y = f(t)\right)$
  - first order: $\dot{y}$ (with respect to time)
  - higher order: $\ddot{y}$ (with respect to time)

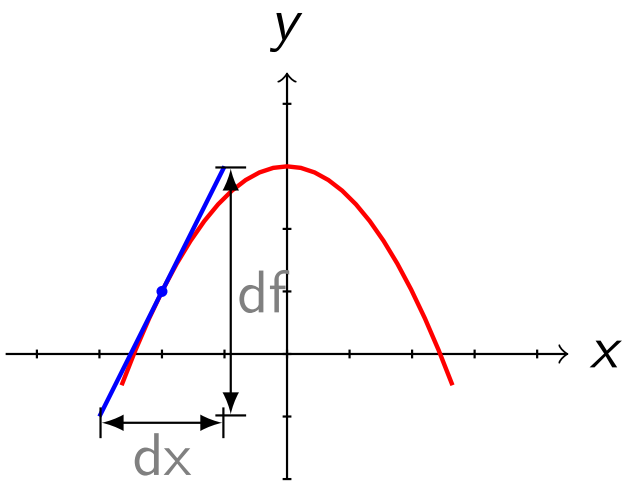


Figure : Leibniz's notation.

PARTIAL DERIVATIVES

So far we have seen how to calculate how much a function of one input variable (e.g. $f(x)$) changes with respect as its input changes. A **partial derivative** tell us how a multi-variable function (e.g. $f(x, y)$) changes with respect to **one of the variables** while the **rest are kept constant**. The partial derivative with respect to x can be noted by: f'_x, f_x, $\partial_x f$ or $\frac{\partial}{\partial x}$.
For example:

$$\begin{cases} \frac{\partial}{\partial x} 3 - x^2 - y^2 = -2x \\ \frac{\partial}{\partial y} 3 - x^2 - y^2 = -2y \end{cases}$$

GRADIENT

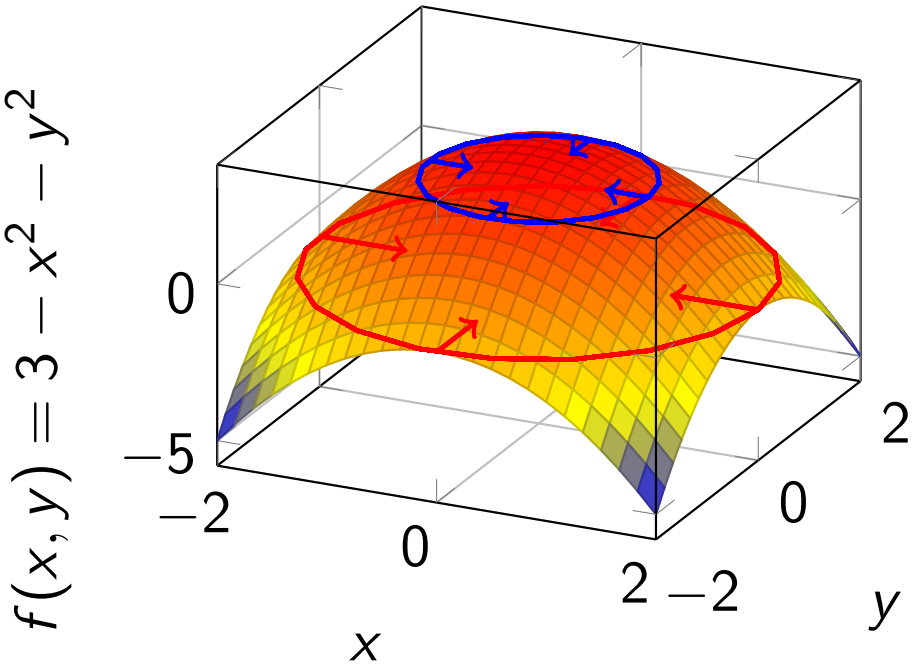
- **Gradient** of a **scalar field** is a **vector field** that points in the direction of the greatest rate of increase of the scalar field.
- Gradient as operator: $\nabla := \left[\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y}\right]$
- $|\vec{x}|$ means the Euclidean length of a vector $\vec{x}$
- Magnitude of rate of change: $|\nabla(f)| = \sqrt{f_x^2 + f_y^2}$

SCALAR AND VECTOR FIELDS

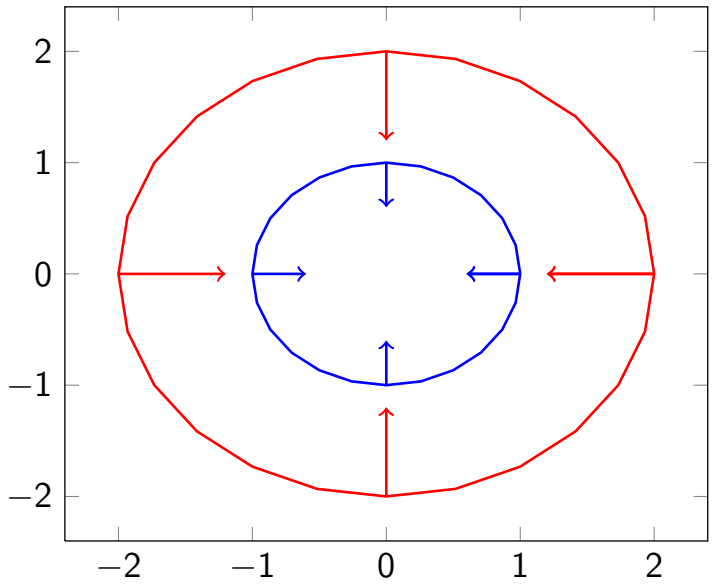
- Function $f(x, y)$ is a scalar field since it maps $\Omega := \mathbb{R} \times \mathbb{R}$ to a single value $z = f(x, y)$
- Gradient of this scalar maps two values (f_x and f_y) for every point $z = f(x, y)$ and, therefore, it is called a vector field

Calculus

Gradient, $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$, points in the direction of the greatest rate of increase.



(a) 3D plot of ∇f



(b) Seen from above

So, how can we calculate/approximate derivatives in images? Typically we use so called 'finite differences', which can be derived from the formal definition of the derivative seen before:

$$f(x)' = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Since in our approximation h has a value that is considerably bigger than 0, we have error in the approximation of the derivative.

1. First order forward difference is given by:

$$D_x^+ f(x,y) = f_x^+(x,y) = \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} \tag{1}$$

2. First order backward difference is given by:

$$D_x^- f(x,y) = f_x^-(x,y) = \frac{f(x,y) - f(x-\Delta x, y)}{\Delta x} \tag{2}$$

3. First order central difference is given by:

$$D_x^0 f(x,y) = f_x^0(x,y) = \frac{f(x+\frac{1}{2}\Delta x, y) - f(x-\frac{1}{2}\Delta x, y)}{\Delta x} \tag{3}$$

4. Second order central difference is given by:

$$DD_x^0 f(x,y) = f_{xx}^0(x,y) = \frac{f(x+\Delta x, y) - 2f(x,y) + f(x-\Delta x, y)}{\Delta x^2} \tag{4}$$

Discretisation

We consider the image to be a continuous function/mapping with $I(x, y, k) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{K+}$, where the domain of the image is $\Omega \subset \mathbb{R} \times \mathbb{R}$ and K defines the number of channels. The kind of images that we are dealing with are, in fact, discretised versions that we receive from the imaging devices, such as digital- or thermal cameras. We define a discretisation grid as:

$$G_h := \{(x, y) \mid x = x_i = ih_x,\ y = y_j = jh_y\ ;\ i, j \in \mathbb{Z}\}$$

where $h = (h_x, h_y)$ is a discretisation parameter. With the discretisation grid, the domain of the discretised images can be defined as $\Omega_h = \Omega \cap G_h$. Instead of $I(x, y) = I(ih_x, jh_y)$, we typically use $I_{i,j}$ when pointing to the pixels.

DIFFUSION

LINEAR IMAGE DIFFUSION EQUATION

$$\boxed{I_t = DIV\left(\nabla I\right)}$$

where $\nabla I = \left[\frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y}\right]$, which is the same as $\nabla I = \partial_x I\vec{i} + \partial_x I\vec{j}$. This equation tells us how each and every 'pixel position' evolves with respect to an artificial time t. Linear diffusion is as called homogeneous diffusion.

Here, physical interpretation of the *divergence* operator is that of diffusion.

NON-LINEAR IMAGE DIFFUSION EQUATION
(Perona&Malik, 1990)

$$\boxed{I_t = DIV\left(g(x,y,t)\nabla I\right)}$$

where $\nabla I = \left[\frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y}\right]$, which is the same as $\nabla I = \partial_x I\vec{i} + \partial_x I\vec{j}$. This equation tells us how each and every 'pixel position' evolves with respect to an artificial time t.

Here we have a function $g(x,y,t)$ that controls the diffusion between the pixels as we shall see later on.

Linear- vs. non-linear diffusion

(a) Non-linear

(b) Linear

Linear (homogeneous) diffusion $I_t = DIV\left(\nabla I\right)$, stopping the evolution at time T, is equivalent to convoluting the image by:

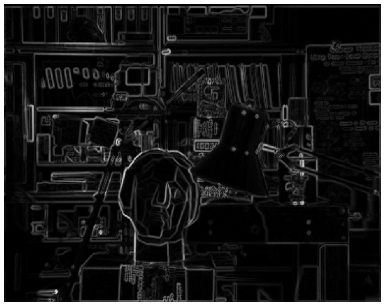
$$I(x, y, T) = \left(K_{2\sqrt{T}} * I\right)(x, y)$$

where $K_\rho(x, y) = \frac{1}{2\pi\rho^2} exp\left(-\frac{x^2 + y^2}{2\rho^2}\right)$ is a Gaussian kernel and $*$ denotes convolution.

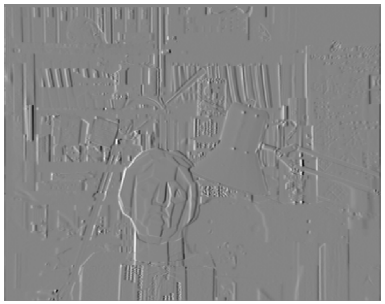
It other words, linear diffusion is equivalent with convoluting the image with a Gaussian kernel, meaning that the image borders also get 'diffused', whereas in the case of non-linear diffusion the object borders remain 'crisp'.

Function of the $g(|\nabla I|)$ is to stop diffusion at borders (i.e. where $|\nabla I|$ obtains a 'big' value).

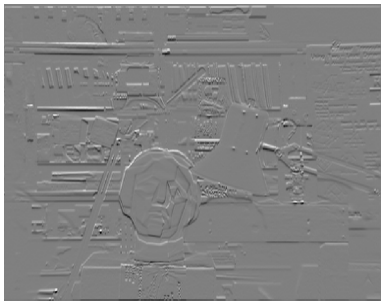
(a) Image.



(b) $|\nabla I|$



(c) I_x

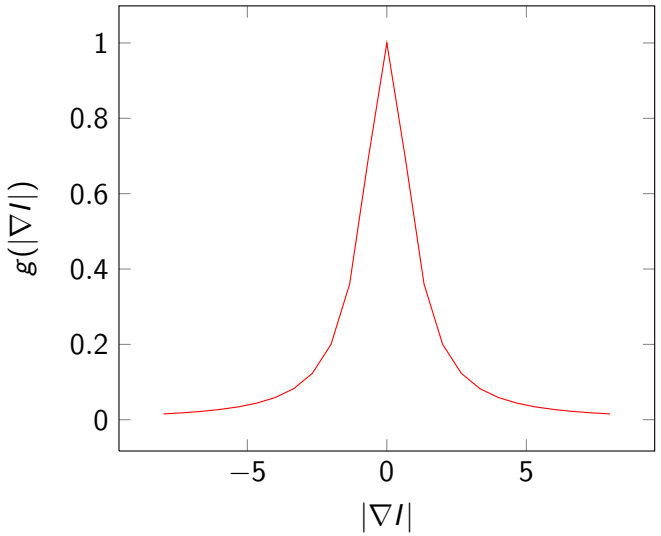


(d) I_y

Therefore, as $|\nabla I| \to \inf$, then $g(|\nabla I|) \to 0$. Once such function is:

$$g(|\nabla I|) = \frac{1}{1 + \left(\frac{|\nabla I|}{\lambda}\right)^2}$$

, where λ is a parameter that controls shape of the function and it is used for defining what strength (i.e. magnitude) of gradient is considered to be a border of an object.



In order for use to use the diffusion equation in a computer, we need to discretise it first. In order to calculate a new value of I at $t+1$ we use a backward Euler (semi-implicit) method, and obtain the following:

$$\frac{I^{t+1} - I^t}{\tau} = DIV\left(g^t \nabla I^{t+1}\right) \tag{5}$$

where τ is size of the time step, subscripts t and $t+1$ refer to the time (i.e. not exponent) and $g^t = g(x, y, t)$. In other words, the idea is to calculate new values of I at $t+1$ (i.e. I^{t+1}) using Equation (5). However, we still have to discretise the DIV operator before we have a model that we can use.

Mathematically, for a differentiable vector function $F = U\vec{i} + V\vec{j}$, divergence operator is defined as:

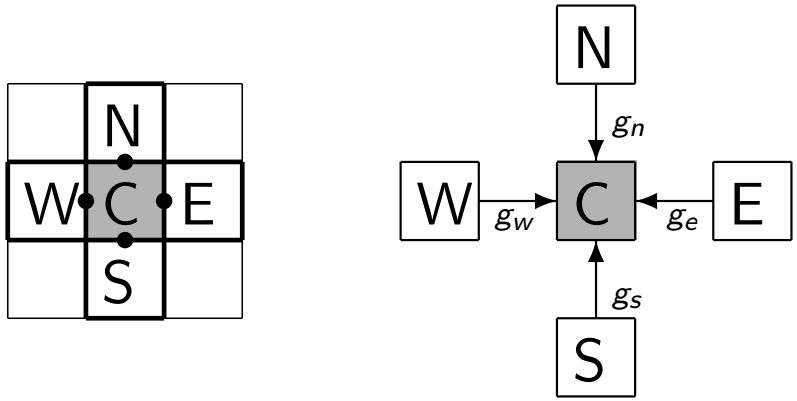
$$\boxed{DIV\left(F\right) = \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y}}$$

In other words, divergence is a sum of partial derivatives of a differentiable vector function. Therefore, in our case, we have the following.

$$\boxed{\begin{aligned} DIV\left(\nabla I\right) &= \frac{\partial}{\partial x}\left(I_x\right) + \frac{\partial}{\partial y}\left(I_y\right) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \Delta I \\ DIV\left(g(x,y,t)\nabla I\right) &= \frac{\partial}{\partial x}\left(g(x,y,t)I_x\right) + \frac{\partial}{\partial y}\left(g(x,y,t)I_y\right) \end{aligned}}$$

Diffusion

Here the physical interpretation of divergence is diffusion:



When we are calculating a new value for the central pixel (C), the values of the surrounding pixels are taken into account. Weights $g_{\{w,n,e,s\}}$ (based on the stopping function $g(x, y, t)$ define how much the surrounding pixels influence.

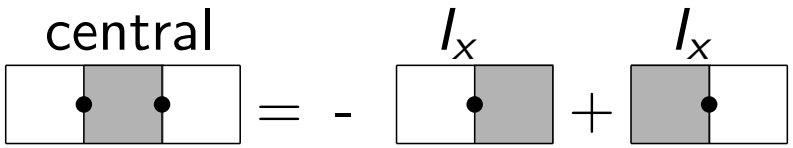
In order to discretise the divergence terms, first we apply the central difference and then forward- and backward differences. Since the distance between the pixels is one ($\Delta x = \Delta y = 1$), we have the following:

$$
\begin{aligned}
D_x^+ f(x,y) &= f_x^+(x,y) = f(x+1,y) - f(x,y) \\
D_x^- f(x,y) &= f_x^-(x,y) = f(x,y) - f(x-1,y) \\
D_x^0 f(x,y) &= f_x^0(x,y) = f(x+\frac{1}{2},y) - f(x-\frac{1}{2},y)
\end{aligned}
$$

Here we show how the term $\frac{\partial}{\partial x}(I_x)$ can be discretised. As it was mentioned earlier, first we discretise the operator $\frac{\partial}{\partial x}$ using a central difference (D_x^0) and obtain the following:

$$\boxed{\frac{\partial}{\partial x}(I_x) \approx I_x(x+\frac{1}{2},y) - I_x(x-\frac{1}{2},y)}$$

where $I_x(x+\frac{1}{2},y)$ means that we have to approximate the derivative I_x at position $(x+\frac{1}{2},y)$. Visually this would be equivalent to:



In order to approximate the derivatives $I_x(x+\frac{1}{2},y)$ and $I_x(x-\frac{1}{2},y)$, we can use the forward- and backward finite differences (i.e. D_x^+ and D_x^-) and, therefore, we have:

$$\boxed{\begin{aligned} I_x(x+\frac{1}{2},y) &\approx I(x+1,y)-I(x,y) \\ I_x(x-\frac{1}{2},y) &\approx I(x,y)-I(x-1,y) \end{aligned}}$$

Discretising $\dfrac{\partial}{\partial y}$ is essentially done in the same way.

Discretisation of $DIV(\nabla I)$:

$$\begin{aligned}\frac{\partial}{\partial x}\Big(I_x\Big)(x,y)+\frac{\partial}{\partial y}\Big(I_y\Big)(x,y) =& \Big(I_x\Big)(x+0.5,y)-\Big(I_x\Big)(x-0.5,y)\\ &+\Big(I_y\Big)(x,y+0.5)-\Big(I_y\Big)(x,y-0.5)\\ =& I(x+1,y)-I(x,y)\\ &+I(x-1,y)-I(x,y)\\ &+I(x,y+1)-I(x,y)\\ &+I(x,y-1)-I(x,y)\\ =& \nabla_E I+\nabla_W I+\nabla_S I+\nabla_N I\end{aligned}$$

For more information, see `http://www.jarnoralli.fi/joomla/images/pdf/diffusion_and_aos.pdf`

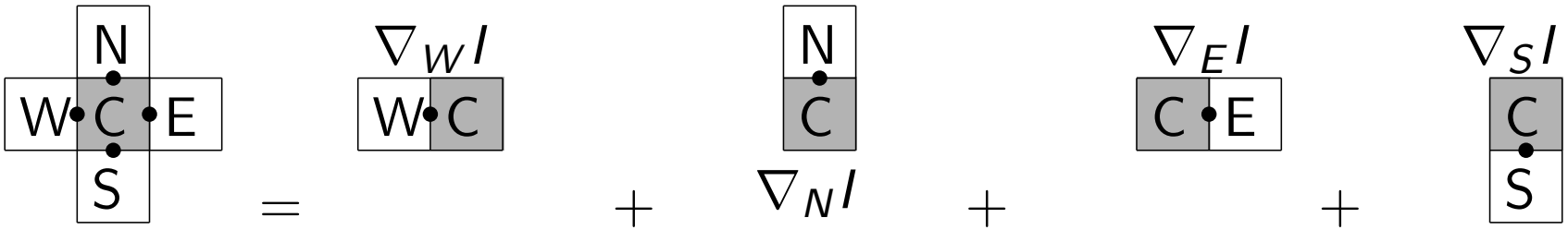
Discretisation of $DIV\left(g(x,y,t)\nabla I\right)$:

$$\begin{aligned}
\frac{\partial}{\partial x}\Big(gI_x\Big)(x,y)+\frac{\partial}{\partial y}\Big(gI_y\Big)(x,y) =& \Big(gI_x\Big)(x+0.5,y)-\Big(gI_x\Big)(x-0.5,y)\\
&+\Big(gI_y\Big)(x,y+0.5)-\Big(gI_y\Big)(x,y-0.5)\\
=& g(x+0.5,y)\Big(I(x+1,y)-I(x,y)\Big)\\
&+g(x-0.5,y)\Big(I(x-1,y)-I(x,y)\Big)\\
&+g(x,y+0.5)\Big(I(x,y+1)-I(x,y)\Big)\\
&+g(x,y-0.5)\Big(I(x,y-1)-I(x,y)\Big)\\
=& g_E\nabla_E I+g_W\nabla_W I+g_S\nabla_S I+g_N\nabla_N I
\end{aligned}$$

For more information, see `http://www.jarnoralli.fi/joomla/images/pdf/diffusion_and_aos.pdf`

Diffusion

Due to heavy use of indices, the discretisation shown in the above slides can be better understood visually:



where:

$$g_W := g(x - 0.5, y),\ \nabla_W I := (I(x - 1, y) - I(x, y))$$
$$g_N := g(x, y - 0.5),\ \nabla_N I := (I(x, y - 1) - I(x, y))$$
$$g_E := g(x + 0.5, y),\ \nabla_E I := (I(x + 1, y) - I(x, y))$$
$$g_S := g(x, y - 0.5),\ \nabla_S I := (I(x, y + 1) - I(x, y))$$

The dots indicate where the $g_{\{W,N,E,S\}}$ are calculated.

The only thing left to do is to plug in the corresponding discretisation in Equation(5), re-order the terms, and now we have a discretised version that we can solved in a computer:

$$\begin{aligned} I_{i,j}^{t+1}\left(1+\tau(g_N^t+g_S^t+g_W^t+g_E^t)\right) = & I_{i,j}^t \\ & + \tau g_N^t\left(I_{i-1,j}^{t+1}\right) \\ & + \tau g_S^t\left(I_{i+1,j}^{t+1}\right) \\ & + \tau g_W^t\left(I_{i,j-1}^{t+1}\right) \\ & + \tau g_E^t\left(I_{i,j+1}^{t+1}\right) \end{aligned} \tag{6}$$

Basically, Equation (6) tells us how I evolves in the case of a single pixel. Therefore, for each and every pixel in the image, we have a similar equation. Another complication is that when solving for I^{t+1}, from Equation (6), the term I^{t+1} appears on both sides of $=$.

What we do is write the system as a system of equations of the form $Ax = b$ (matrix-vector format), where one solves for x (in our case I^{t+1}) . In this case the system matrix A contains the information about the interactions (i.e. diffusion coming from the DIV term) between the pixels.

MATRIX-VECTOR FORMAT

If the domain of the discretised image is $\Omega_h : [1, m] \times [1, n]$ (discrete image with m columns and n rows), the system matrix A is defined on $[m \cdot n] \times [m \cdot n]$ (here $\cdot$ denotes multiplication). Now, we can write the Euler forward, semi-implicit formulation in a vector/matrix format as follows:

$$\frac{\mathbf{I}^{t+1} - \mathbf{I}^{t}}{\tau} = A\left(\mathbf{I}^{t}\right)\mathbf{I}^{t+1} \tag{7}$$

where $\mathbf{I} := (I)_{\mathcal{I}}$ with $\mathcal{I} = [1 \dots N]$ and $N = mn$ (i.e. all the pixels in the image).

In order to simplify the notation, we write A instead of $A\left((\mathbf{I}_k)^t\right)$. Id refers to the identity matrix. Therefore, we have:

$$\boxed{\mathbf{I}^{t+1} = \mathbf{I}^t + \tau A \mathbf{I}^{t+1}} \tag{8}$$

From which $\mathbf{I}^{t+1}$ can be solved as follows:

$$\boxed{\mathbf{I}^{t+1} = \left(Id - \tau A\right)^{-1} \mathbf{I}^t} \tag{9}$$

Typically we don't directly calculate the inverse of the $\left(Id - \tau A\right)$, but we use iterative numerical methods, such as Gauss-Seidel method, to solve for $\mathbf{I}^{t+1}$.

Diffusion

Effects of non-linear diffusion upon image

(a) Orig. (b) Diffused.

IMAGE CORRESPONDENCES

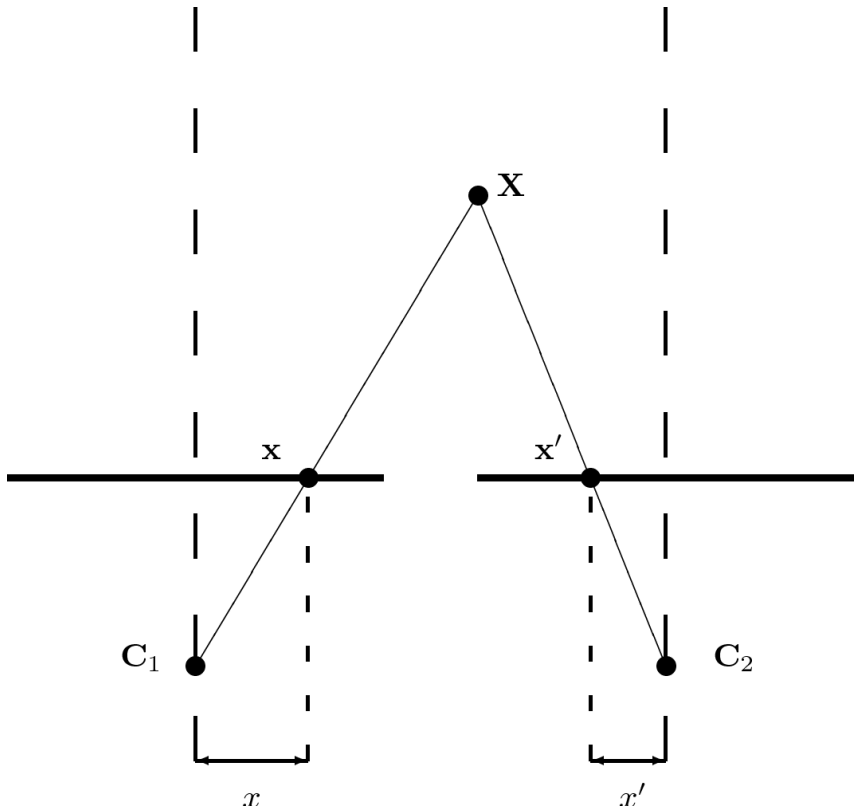
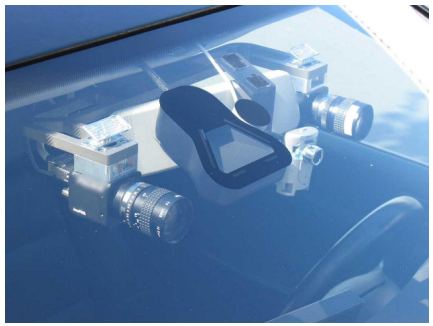
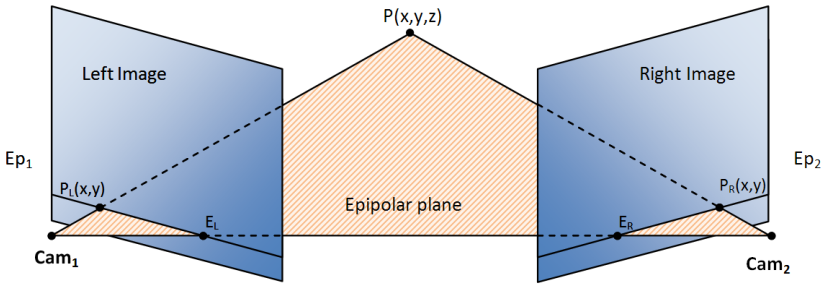


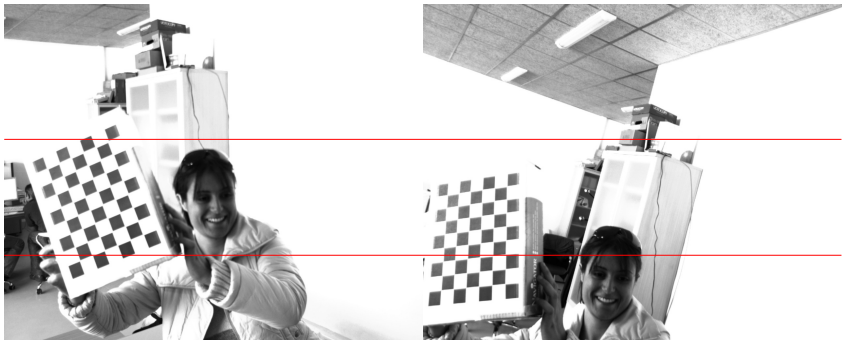
Figure : $\mathbf{C}_1$ and $\mathbf{C}_2$ are the camera centres of the respective cameras. $\mathbf{X}$ is the point of interest in 3D world coordinates, while $\mathbf{x}$ and $\mathbf{x}'$ are the images of the same on the respective camera planes. Disparity is $d = x - x'$.



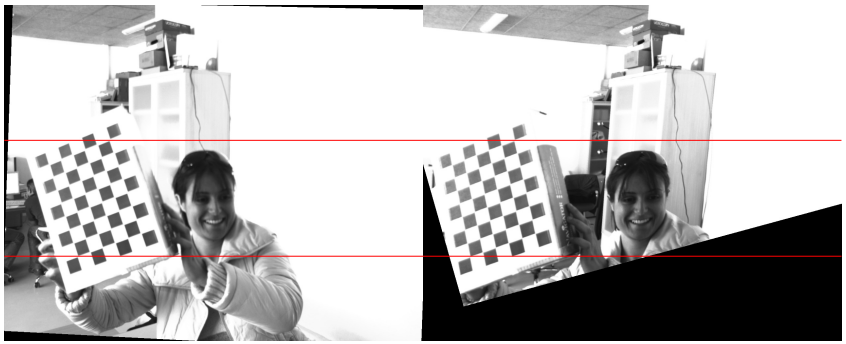
(a) Stereo-cameras.



(b) Epipolar geometry.

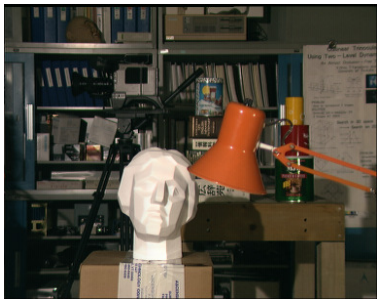


(c) Not rectified. Image information does not match on horizontal lines.

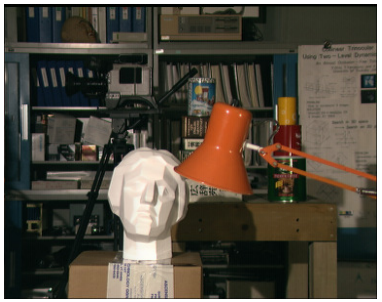


(d) Rectified.

Correspondences



(a) Left.



(b) Right.

(c) Left-right.

(d) Disparity.

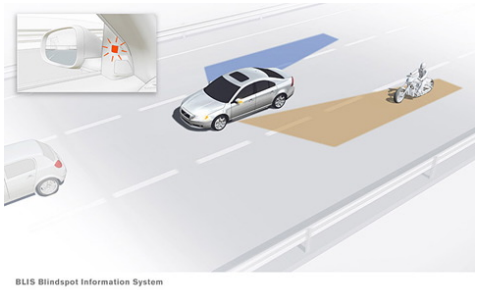
Figure : Intensity level codifies depth of the objects in the disparity image.

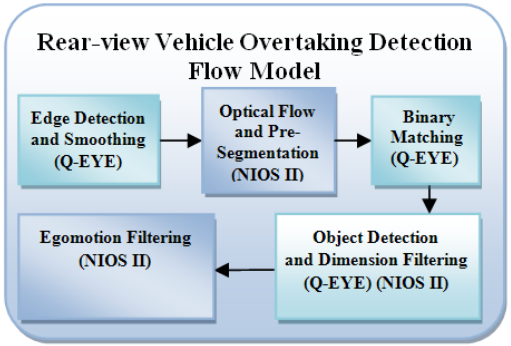
Correspondences

(a) Left image. (b) Disparity. (c) 3D.

Figure : A robotics related disparity example. Based on the 3D 'image', the system detects the object and the decides upon how it can be manipulated etc.

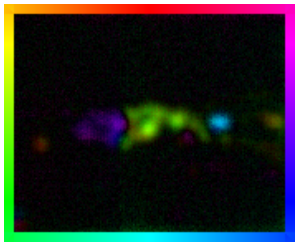
Correspondences





(c) System scheme.

(d) Image.

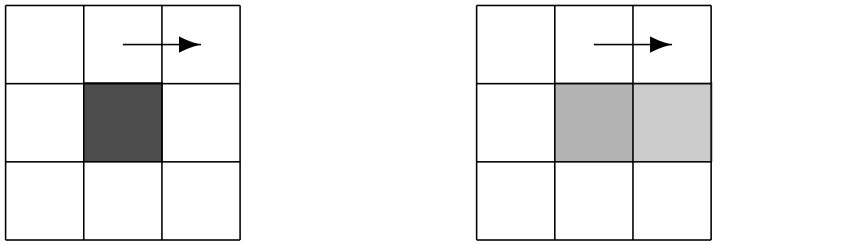


(e) Flow.

(f) Dir.

P. Guzmán, J. Díaz, **J. Ralli**, R. Agís, and E. Ros, 'Low-cost Sensor to Detect Overtaking Based on Optical-flow', Machine Vision and Applications.

We assume that any observed change in pixel intensity level is due to movement $(x-u, y-v)$ in the image plane as time changes from t to $t+1$.



(a) I(x,y,t) (b) I(x+u,y+v,t+1)

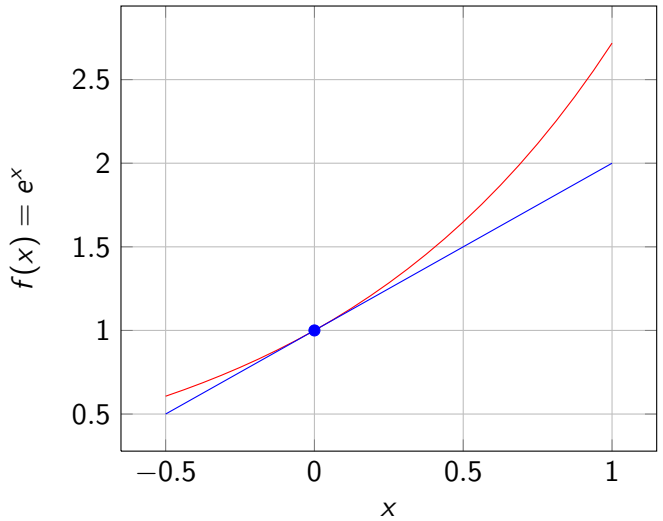
Mathematically this constancy assumption can be formulated as:

$$\begin{aligned} \text{op-flow: } & I(x,y,t) = I(x-u, y-v, t+1) \\ \text{stereo: } & I_L(x,y) = I_R(x-d, y) \end{aligned} \tag{10}$$

Correspondences

In order for to use the constancy assumption, we need to linearise it. To this end we use Taylor expansion: if a real-value function f is differentiable at the point a then it has a linear approximation at the point a. This means that there exists a function h_1 such that:

$$f(x) = f(a) + f'(a)(x-a) + h_1(x)(x-a), \quad \lim_{x \to a} h_1(x) = 0$$



Correspondences

The gray-level constancy seen previously reads: The gray-level constancy is as given by equation (11), while first order (linear) approximation is given by equation (12). By plugging (12) into (11) we obtain (13):

$$I(x,\, y,\, t) = I(x+u,\, y+v,\, t+1) \tag{11}$$

$$I(x+u,\, y+v,\, t+1) = I(x,\, y,\, t) + I_x u + I_y v + I_t \tag{12}$$

$$\boxed{I_x u + I_y v + I_t = 0} \tag{13}$$

Equation (13) is called the Optical-Flow Constraint (OFC).

In the Lucas-Kanade (1981) method the optical-flow vector is assumed to be 'constant' in a certain neighbourhood. Neighbourhood is typically defined by convolving with a Gaussian K_ρ having a standard deviation ρ. This allows estimating the optical-flow, at each point, by minimising the energy function:

$$E(u,\, v) = \frac{1}{2} K_\rho * \left((I_x u + I_y v + I_t)^2 \right)$$

A minimum $(u,\, v)$ of E satisfies $\partial_u E = 0$ and $\partial_v E = 0$, therefore:

$$\underbrace{\begin{bmatrix} K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & K_\rho * I_y^2 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -K_\rho * I_x I_t \\ -K_\rho * I_x I_t \end{bmatrix} \tag{14}$$

where $*$ denotes convolution.

In order for Equation (14) to have a solution, determinant of the system matrix A must be nonzero:

$$det(A) = (K_\rho * I_x^2)(K_\rho * I_y^2) - (K_\rho * I_x I_y)(K_\rho * I_x I_y) \tag{15}$$

Determinant is nonzero if and only if both $K_\rho * I_x$ and $K_\rho * I_y$ are nonzero. In other words, we can solve for optical-flow only if both the derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are nonzero!

Correspondences

1. Pros
   1. Simple, easy to understand
   2. Fast
   3. Sparse, approximations only where information is available
2. Cons
   1. Sparse, obtained (u, v) might not cover the whole image
   2. Local solution
   3. Basic model is not very robust
   4. Can only detect small displacements

The Horn-Schunck (1981) model includes a smoothness term that penalises changes in the optical flow field:

$$E(u,v) = \int_\Omega \left(\underbrace{(I_t - I_x u - I_y v)^2}_{\text{data term}} + \alpha \underbrace{\left(|\nabla u|^2 + |\nabla v|^2\right)}_{\text{smoothness term}} \right) \mathbf{dx} \qquad (16)$$

where Ω is the image domain, i.e. this model is *global* in a sense that we seek (u, v) that minimises the energy E for the whole image. The Tikhonov regulariser penalises deviations in the optical-flow field.

A necesary condition for a minimum (or maximum) of (16) is that the related Euler-Lagrange equations equal to zero:

$$\begin{aligned} (I_t - I_x u - I_y v)\, I_x + DIV(\nabla u) = 0 \\ (I_t - I_x u - I_y v)\, I_y + DIV(\nabla v) = 0 \end{aligned} \tag{17}$$

Discresation of Equation (17) leads to a sparse linear system of equations that can be solved using iterative methods, such as Gauss-Seidel, Successive-Over-Relaxation etc.

'Because a differentiable functional is stationary at its local maxima and minima, the **Euler–Lagrange equation is useful in seeking the function that minimizes (or maximizes) it**. *This is analogous to calculus:* **a differentiable function attains its local extrema where its derivatives are zero**' (Wikipedia)

$$S(f) = \int_a^b L(x, f(x), f'(x)) \tag{18}$$

where $f(x)$ is a function of a real variable x and L is the functional. The corresponding Euler-Lagrange equation is given by:

$$\frac{\partial}{\partial f} L - \frac{d}{dx} \frac{\partial}{\partial f'} L = 0 \tag{19}$$

If we solve the coupled PDE (17) in elliptic, semi-implicit form, after discretisation we have:

$$\begin{bmatrix} (I_{i,j})_x^2 + 4\alpha & (I_{i,j})_y (I_{i,j})_x \\ (I_{i,j})_y^2 + 4\alpha & (I_{i,j})_y (I_{i,j})_x \end{bmatrix} \begin{bmatrix} u_{i,j}^{l+1} \\ v_{i,j}^{l+1} \end{bmatrix} = \begin{bmatrix} (I_{i,j})_t (I_{i,j})_x + \alpha(u_{i-1,j}^l + u_{i+1,j}^l + u_{i,j-1}^l + u_{i+1,j}^l) \\ (I_{i,j})_t (I_{i,j})_y + \alpha(v_{i-1,j}^l + v_{i+1,j}^l + v_{i,j-1}^l + v_{i+1,j}^l) \end{bmatrix} \tag{20}$$

When solving for the whole image, we end up with a linear system of equations of the form $Ax = b$, where A is a sparse matrix. As was mentioned earlier, this can be solved using iterative methods, such as Gauss-Seidel, Successive-Over-Relaxation etc.

Correspondences

1. Pros
   1. Global solution, in theory more robust
   2. Linear, therefore easy and quick to solve
   3. Smoothness of the solution
2. Cons
   1. Only works with small displacements
   2. Global solution, propagates even erroneous solutions
   3. Quadratic error function emphasises occlusions etc.
   4. Tikhonov regulariser propagates solution accross object boudaries

Correspondences

In order for the HS model to take into account 'outliers' (i.e. model errors), we can embed both the data- and the smoothness term in a robust error function:

$$E(u,v) = \int_\Omega \left(\underbrace{\Psi\Big((I_t - I_x u - I_y v)^2 \Big)}_{\text{data term}} + \alpha \underbrace{\Psi\Big(|\nabla u|^2 + |\nabla v|^2 \Big)}_{\text{smoothness term}} \right) \mathbf{dx} \quad (21)$$

where $\Psi(s^2)$ is a robust error function, e.g. $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$.

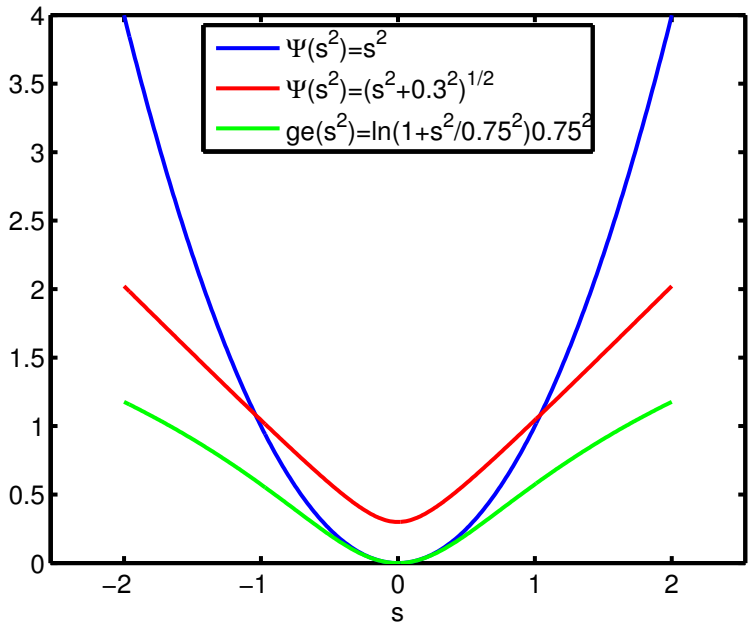
Again, a necesary condition for a minimum (or maximum) of (21) is that the related Euler-Lagrange equations equal to zero:

$$
\begin{aligned}
E_D &= I_t - I_x u - I_y v \\
E_S &= |\nabla u|^2 + |\nabla v|^2 \\
\Psi'\Big(E_D\Big)\big(I_t - I_x u - I_y v\big) I_x - \alpha DIV\Big(\Psi'(E_S)\nabla u\Big) &= 0 \\
\Psi'\Big(E_D\Big)\big(I_t - I_x u - I_y v\big) I_y - \alpha DIV\Big(\Psi'(E_S)\nabla v\Big) &= 0
\end{aligned}
\tag{22}
$$

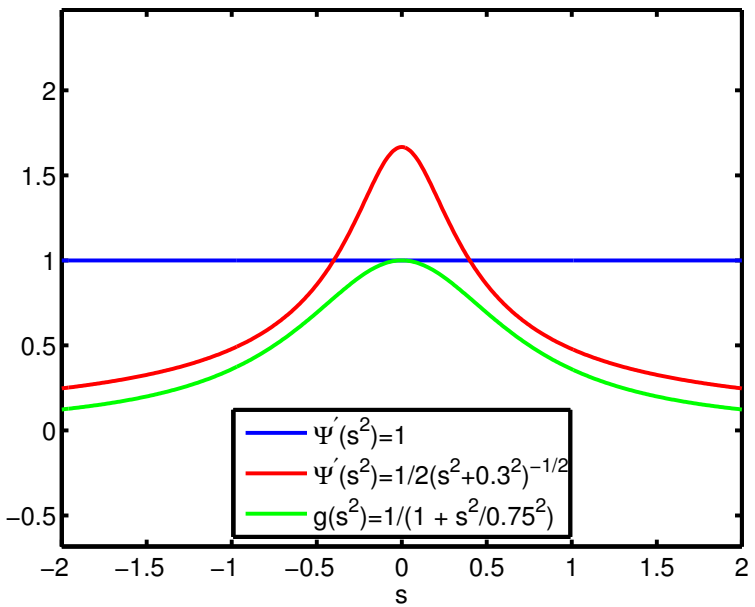
where $\Psi'(s^2)$ (derivative of $\Psi(s^2)$) is the *influence* function (i.e. how much weight is given to each approximation).

Table : Error functions used in the smoothness term.

ERROR AND CORRESPONDING INFLUENCE FUNCTIONS			
$\Psi_R(s^2) = s^2$	$\Psi'_R(s^2) = 1$	quadratic (blue)	not robust
$\Psi_R(s^2) = \sqrt{s^2 + \epsilon^2}$	$\Psi'_R(s^2) = 1/\sqrt{s^2 + \epsilon^2}$	TV (red)	robust
$ge(s^2) = ln(1 + s^2/\lambda^2)\lambda^2$	$g(s^2) = 1/(1 + s^2/\lambda^2)$	ln (green)	robust



(a) Error functions.



(b) Influence functions.

Correspondences

$$\sum_{k=1}^{K} \Psi^{'}\left((E^{l})_{D}\right)(I_t I_x - I_x^2 u_{i,j}^{l+1} - I_y I_x v_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_N(u_{i-1,j}^{l+1} - u_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_S(u_{i+1,j}^{l+1} - u_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_W(u_{i,j-1}^{l+1} - u_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_E(u_{i,j+1}^{l+1} - u_{i,j}^{l+1}) = 0$$
$$\sum_{k=1}^{K} \Psi^{'}\left((E^{l})_{D}\right)(I_t I_y - I_x I_y u_{i,j}^{l+1} - I_y^2 v_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_N(v_{i-1,j}^{l+1} - v_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_S(v_{i+1,j}^{l+1} - v_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_W(v_{i,j-1}^{l+1} - v_{i,j}^{l+1})$$
$$+K\alpha\Psi^{'}_{i,j}(E_S^l)_E(v_{i,j+1}^{l+1} - v_{i,j}^{l+1}) = 0$$

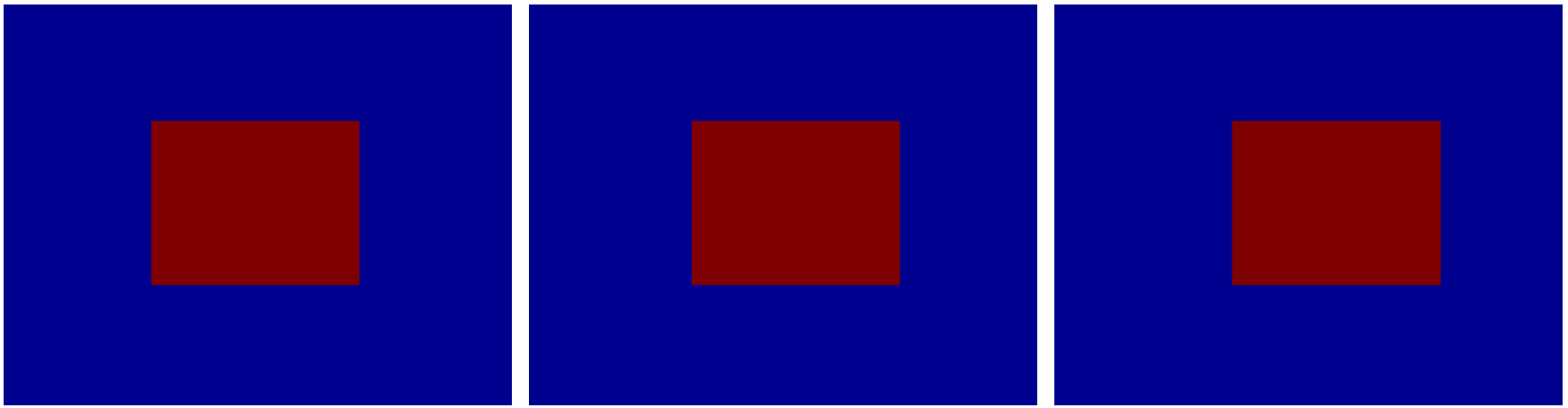
For more information, see `http://www.jarnoralli.fi/joomla/images/pdf/thesis_jralli.pdf`

Correspondences

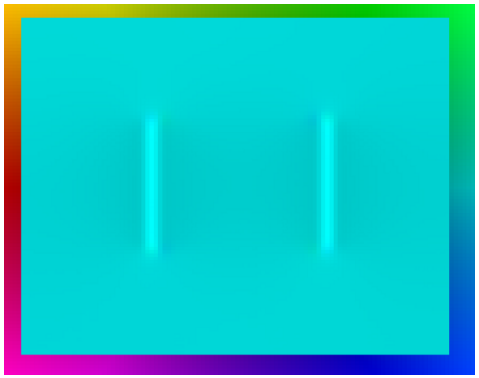
1. Pros
   1. Global solution, in theory more robust
   2. Non-linear, more complicate to solve
   3. Smoothness of the solution
   4. Non-quadratic data term: the model takes into account outliers
   5. Non-quadratic smoothness term: the model does not propagate optical-flow field from one object to another
2. Cons
   1. Only works with small displacements
   2. Global solution, propagates even erroneous solutions

Correspondences

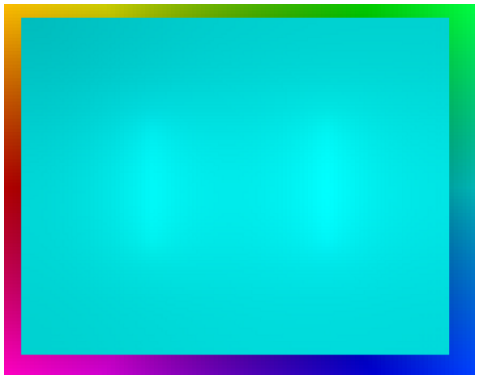
No background information, object moving to right



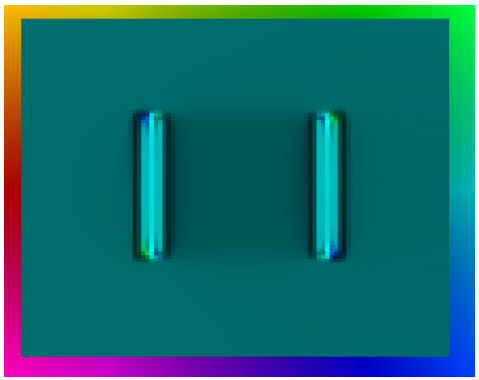
(a) $I(:,:,1)$ (b) $I(:,:,2)$ (c) $I(:,:,3)$



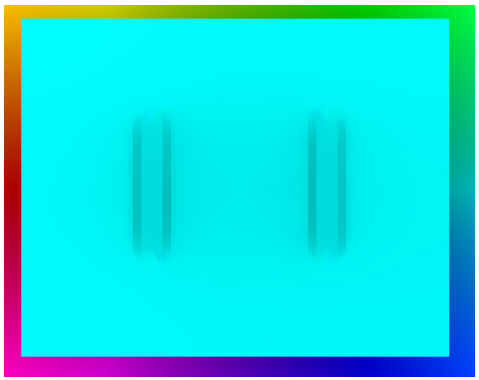
(a) HS $\alpha = 0.1$



(b) HS $\alpha = 10$



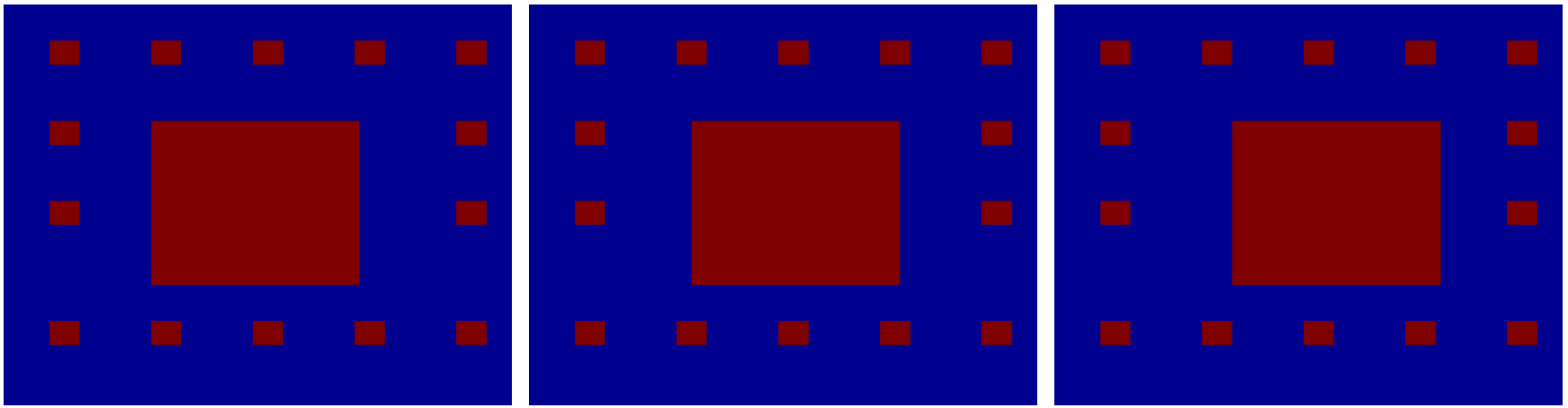
(c) HS robust $\alpha = 0.01$



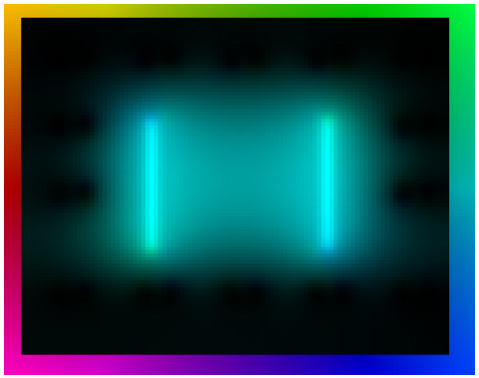
(d) HS robust $\alpha = 0.05$

Correspondences

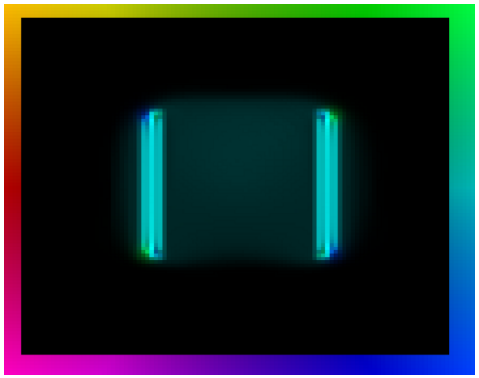
With background information, object moving to right



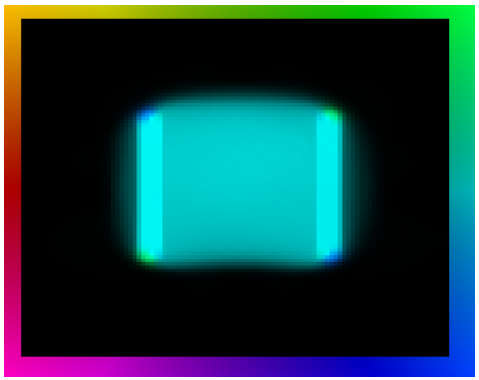
(a) $I(:,:,1)$ (b) $I(:,:,2)$ (c) $I(:,:,3)$



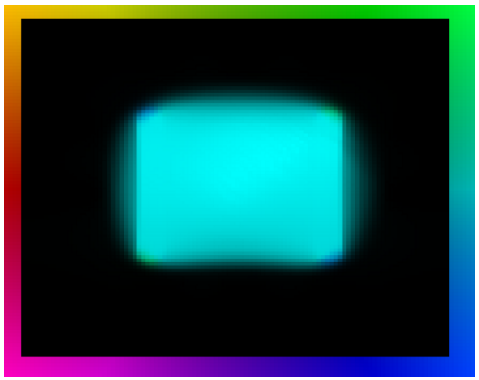
(a) HS $\alpha = 0.1$



(b) HS robust $\alpha = 0.01$

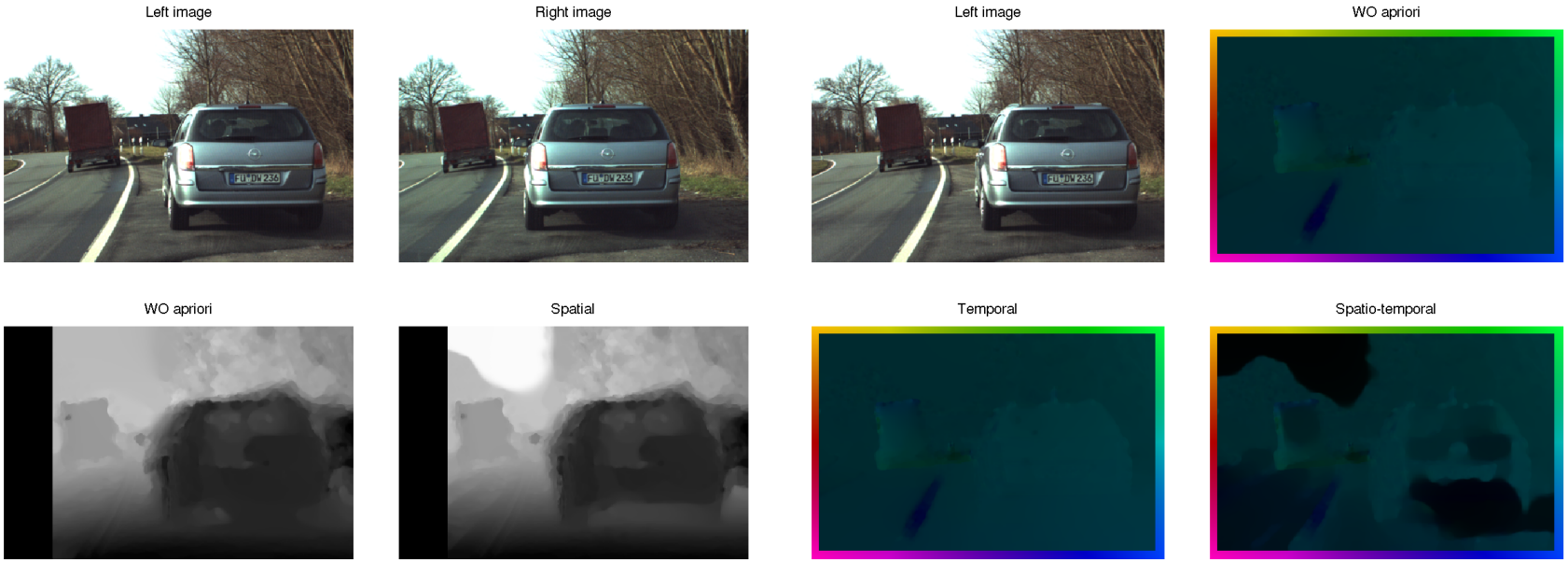


(c) HS robust $\alpha = 0.05$



(d) HS robust $\alpha = 0.1$

Results related to DRIVSCO project



(a) Disparity

(b) Optical-flow

Bibliography

- From Pixels to Regions: Partial Differential Equations in Image Analysis, PhD thesis of Thomas Brox, 2005.
- Estimación del Flujo Óptico En Secuencias de Imágenes y de la Carta de Disparidad en Pares Estéreo, tesis doctoral de Javier Sánchez Pérez, 2001.
- Fusion and Regularisation of Image Information in Variational Correspondence Methods, PhD thesis of Jarno Ralli, 2011.
- PDE Based Image Diffused and AOS, `http://www.jarnoralli.fi/joomla/images/pdf/diffusion_and_aos.pdf`
- `www.jarnoralli.com`

End

THANK YOU FOR LISTENING!