# Motion Estimation

## 4C8: Digital Media Processing

Ussher Assistant Professor François Pitié

2021/2022

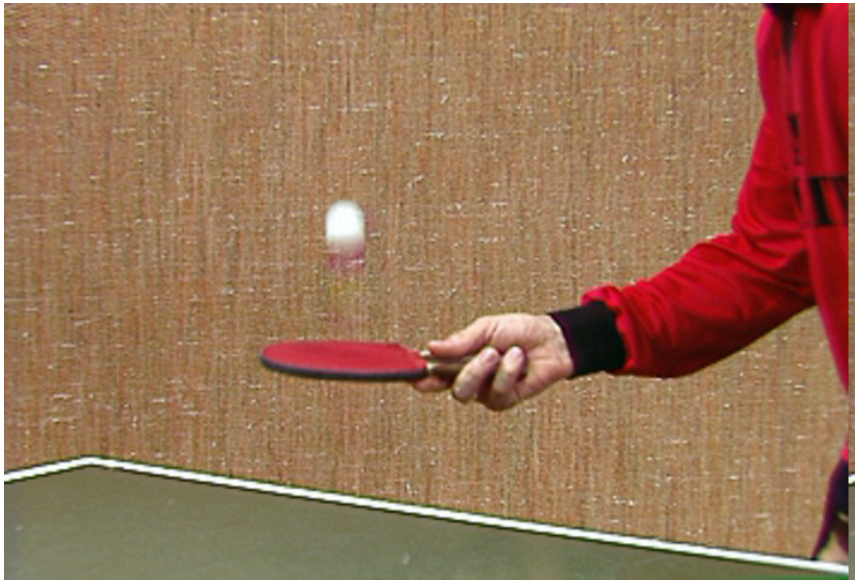
Department of Electronic & Electrical Engineering , Trinity College Dublin

# Motion

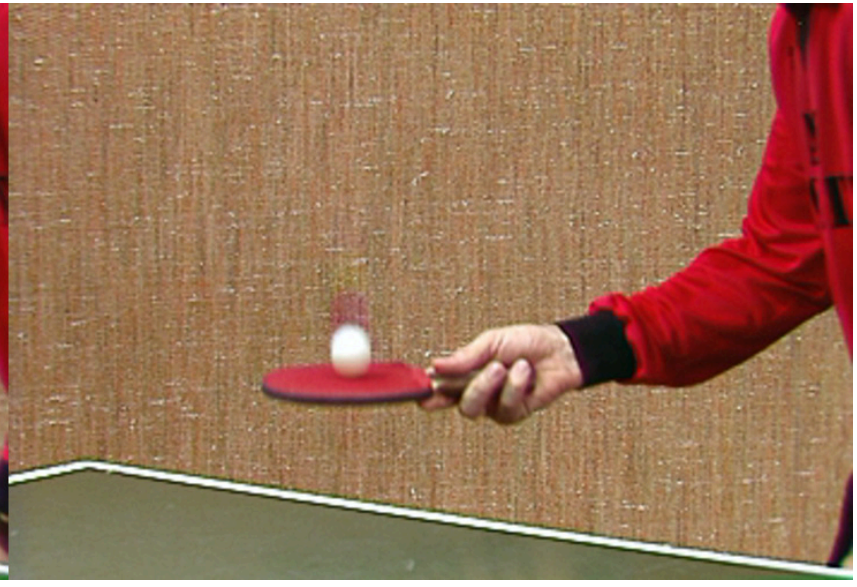
The image formed in a camera is a 2D projection of a 3D world. The task of reconstructing a 3D model of a scene from different 2D camera views has been a long standing area of research in computer vision (see topics such as structure from motion).

In this course we will only be interested in the *apparent 2D motion* between frames. This is known as Motion Estimation or Optical Flow, and this is the key to analysing and processing image sequences.

# Motion Vectors

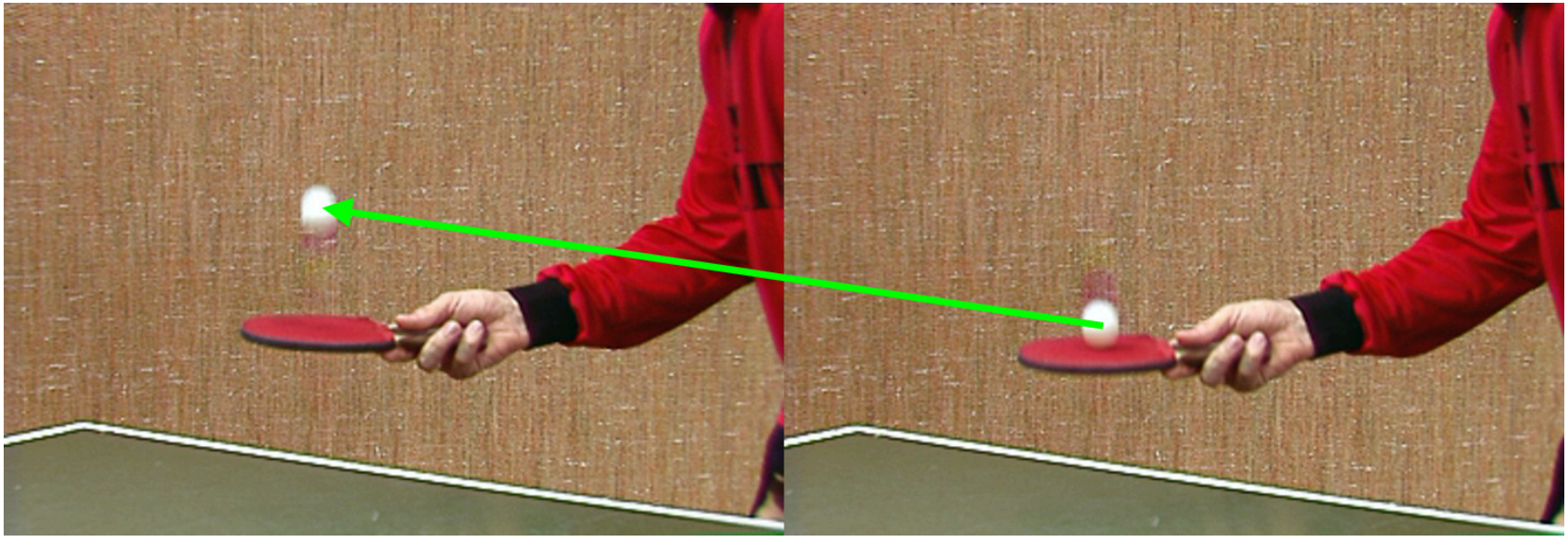


frame $n-1$



frame $n$

# Motion Vectors



frame $n-1$ frame $n$

# Motion Vectors and Motion Field

The core assumption we make here is that the intensities at all image points $(i, j)$ are the same between frames $n-1$ and $n$:

$$I_n(i,j) = I_{n-1}(i + d_i, j + d_j)$$

or using vector notations $\mathbf{x} = (i, j)$ and $\mathbf{d}_{n,n-1}(\mathbf{x}) = (d_i, d_j)$:

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x}))$$

The displacement $\mathbf{d}_{n,n-1}(\mathbf{x})$ is called the **motion vector** and maps the site at the current frame $n$ to the corresponding site in the previous frame $n-1$.

The array of all these motion vectors over the entire frame $\mathbf{d}_{n,n-1}$ is called the *motion field*.

# Displaced Frame Difference

The **Displaced Frame Difference (DFD)** measures the *reconstruction* error and is defined as:

$$DFD_{\mathbf{d}_{n,n-1}}(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x}))$$

To find the optimum motion field we need to find the motion vector field that minimises the DFD over the entire image. For instance:

$$\hat{\mathbf{d}}_{n,n-1} = \arg \min_{\mathbf{d}_{n,n-1}} \sum_{\mathbf{x}} \left( DFD_{\mathbf{d}_{n,n-1}}(\mathbf{x}) \right)^2$$

or

$$\hat{\mathbf{d}}_{n,n-1} = \arg \min_{\mathbf{d}_{n,n-1}} \sum_{\mathbf{x}} \left| DFD_{\mathbf{d}_{n,n-1}}(\mathbf{x}) \right|$$

However things are not that simple as we will see later. For one, the brightness constancy assumption is not that realistic.

# Block Matching

Block Matching is the motion estimation technique commonly used in video compression.

The two basic assumptions of Block Matching are:

1. Constant translational motion over small blocks (say $8 \times 8$ or $16 \times 16$) in the image. This is the same as saying that there is a minimum object size that is larger than the chosen block size.
2. There is a maximum (pre-determined) range for the horizontal and vertical components of the motion vector at each pixel site. This is the same as assuming a maximum velocity for the objects in the sequence. This restricts the range of vectors to be considered and thus reduces the cost of the algorithm.

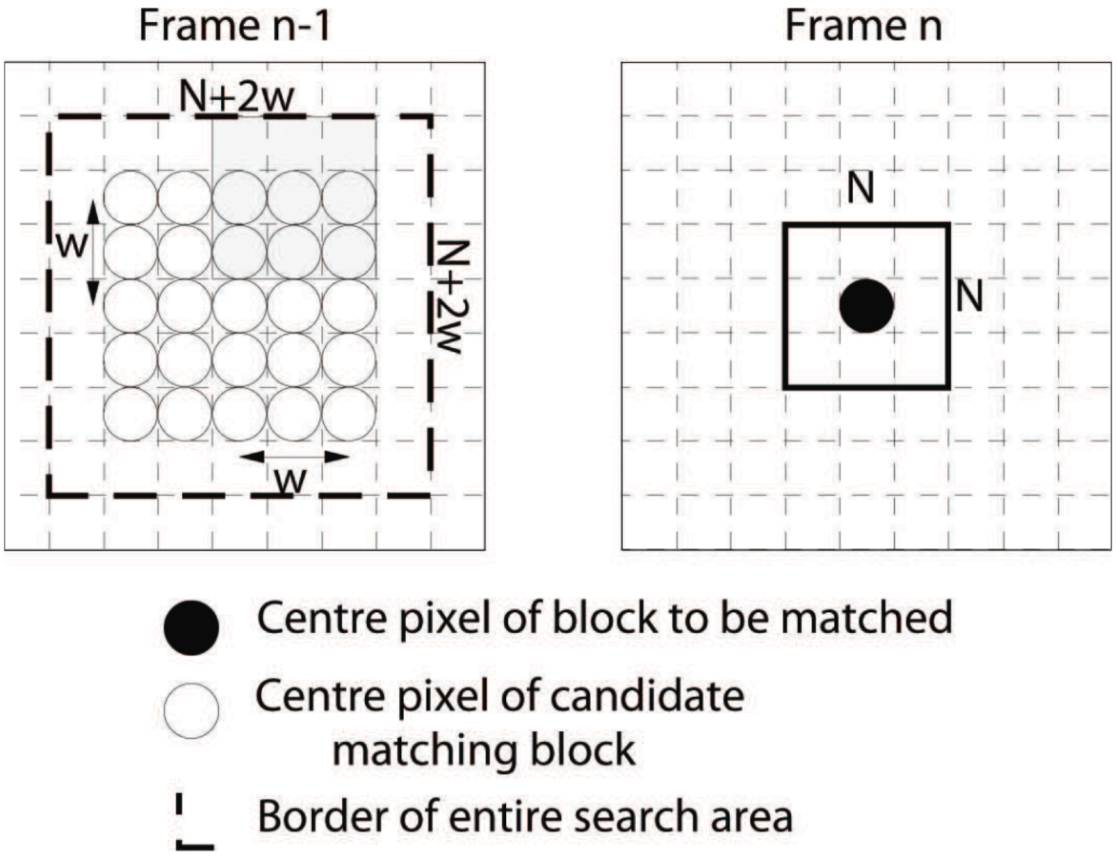
# Block Based Motion Estimation

The image in frame $n$, is divided into blocks usually of the same size $N \times N$. Each block is considered in turn and a motion vector is assigned to each. The motion vector is chosen by matching the block in frame $n$ with a set of blocks of the same size at locations defined by some search pattern in the previous frame.

For each block, a single Motion Vector is found by finding the displacement $\mathbf{d}$, within a search window of radius $w$ pixels, that minimises the MAE of the DFD:

$$MAE(\text{Block}, \mathbf{d}) = \frac{1}{N^2} \sum_{\mathbf{x} \in \text{Block}} |DFD(\mathbf{x}, \mathbf{d})|$$

# Full Motion Search Block Matching




The positions are indicated by a circle in frame $n-1$ are searched for a match with the $N \times N$ block in frame $n$. One block to be examined is located at displacement $[1, -2]$, and is shaded.

# Full Motion Search Block Matching

Previous slide shows the search space used in a *full motion search* technique. The current block is compared to every block of the same size in an area of size $(2w+N)\times(2w+N)$. The search space is chosen by deciding on the maximum displacement allowed: here the maximum displacement estimated is $\pm w$ for both horizontal and vertical components.

# Full Motion Search Block Matching - Example 1

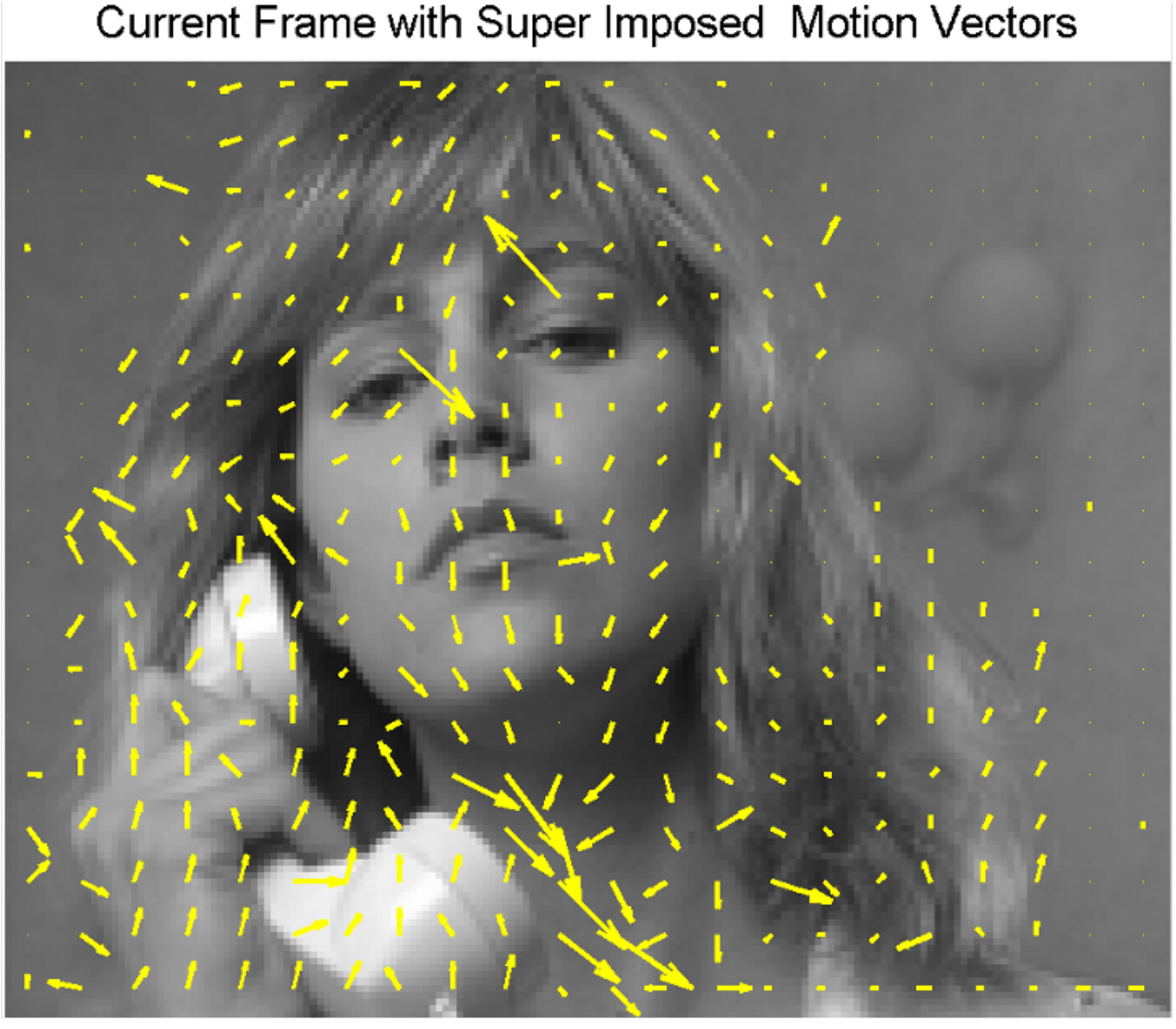
Previous Frame

# Full Motion Search Block Matching - Example 1

Current Frame with Super Imposed Motion Vectors

# Full Motion Search Block Matching - Example 2

# Full Motion Search Block Matching - Example 2

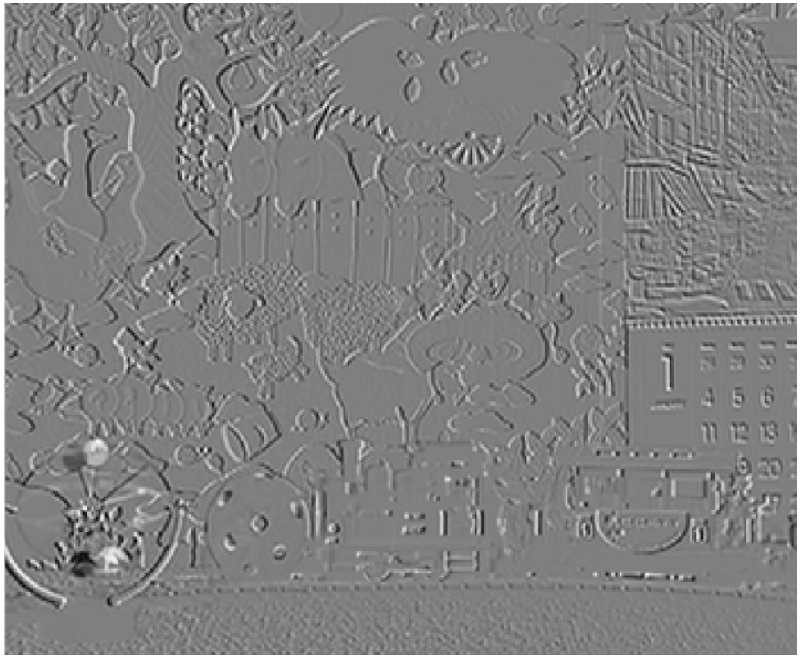


# Motion Compensation

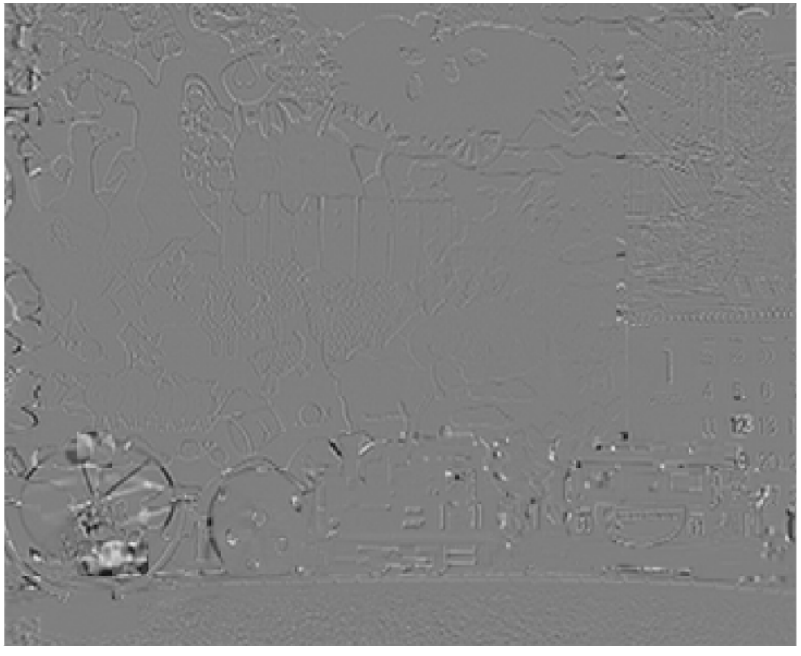
Motion Compensation is when the frame is reconstructed from the previous frame matching blocks as found by motion estimation.

$$I_n^{mc}(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x}))$$

Effect of motion compensation on the DFD:

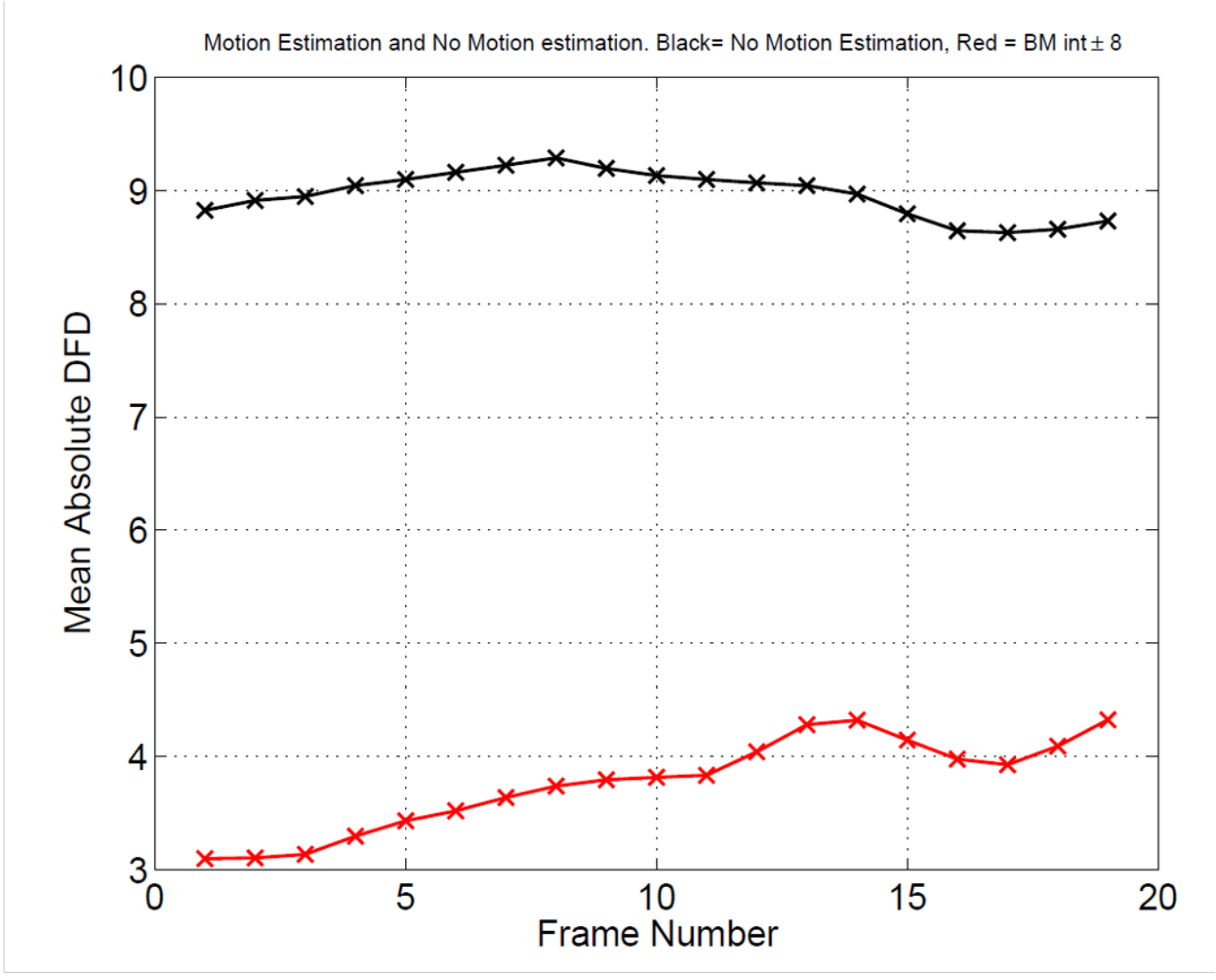


without motion compensation



with compensation

# Motion Compensation and DFD



# Motion Compensation and DFD

As we can see the DFD is reduced but not quite zero. This is because the motion vectors in Black-Matching are assigned to blocks and that we assume a translational model over the entirely of the block.

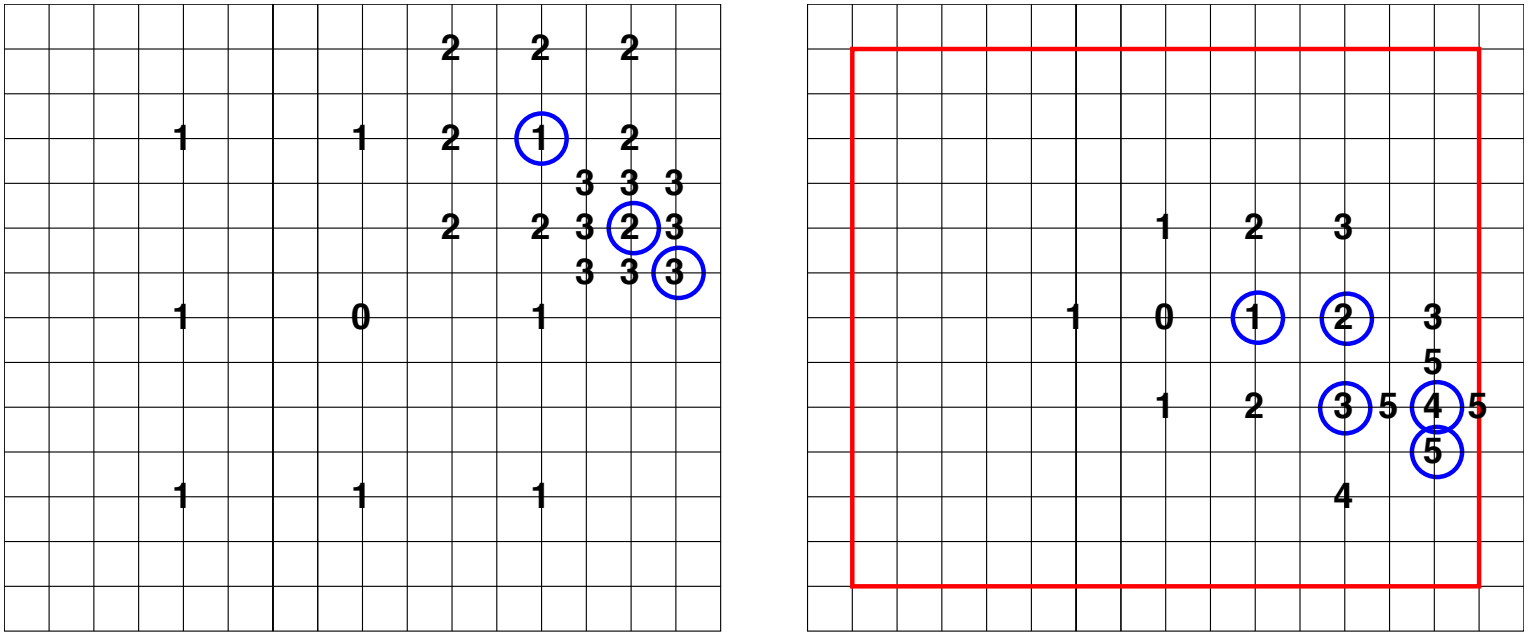
# Full Search Block Matching Complexity

There are $(2w+1)^2$ candidates (assuming vectors are accurate to 1 pel). Computing the mean absolute error requires 1 sub, 1 abs, 1 add per pixel. Thus if the blocksize is $N$, the number of ops is $3N^2(2w+1)^2$ per block. There is an extra operation to find the min of $N^2$ values but the cost is much less compared to calculating the MAE.

The complexity order is $\mathcal{O}(w^2)$, i.e. quadratic wrt the search radius.

# Sub-Sampled Block Based Motion Estimation

The computational burden of Full Search Block-Matching can be reduced by sub-sampling the range of possible motion vectors.




Left: Tree-Step Block-Matching. Right: Cross-Search Block-Matching. Central pixel of the searched block is shown. The search size is shown in red. The best matches at each step is shown in blue.

# Sub-Sampled Block Based Motion Estimation

The advantage of such hierarchical sampling strategies is the much reduced computational cost.

The downside is that we might miss the actual best motion vector.

These approaches are still very efficient in practice, especially for motion compression, where obtaining a low DFD is more important than knowing the exact motion vector.

# Sub-Pixelic Accuracy

So far we have only considered integer valued displacements. However it is critical in most applications to consider sub-pixelic displacements, as most points in the scene typically undergo fractional pel motion.

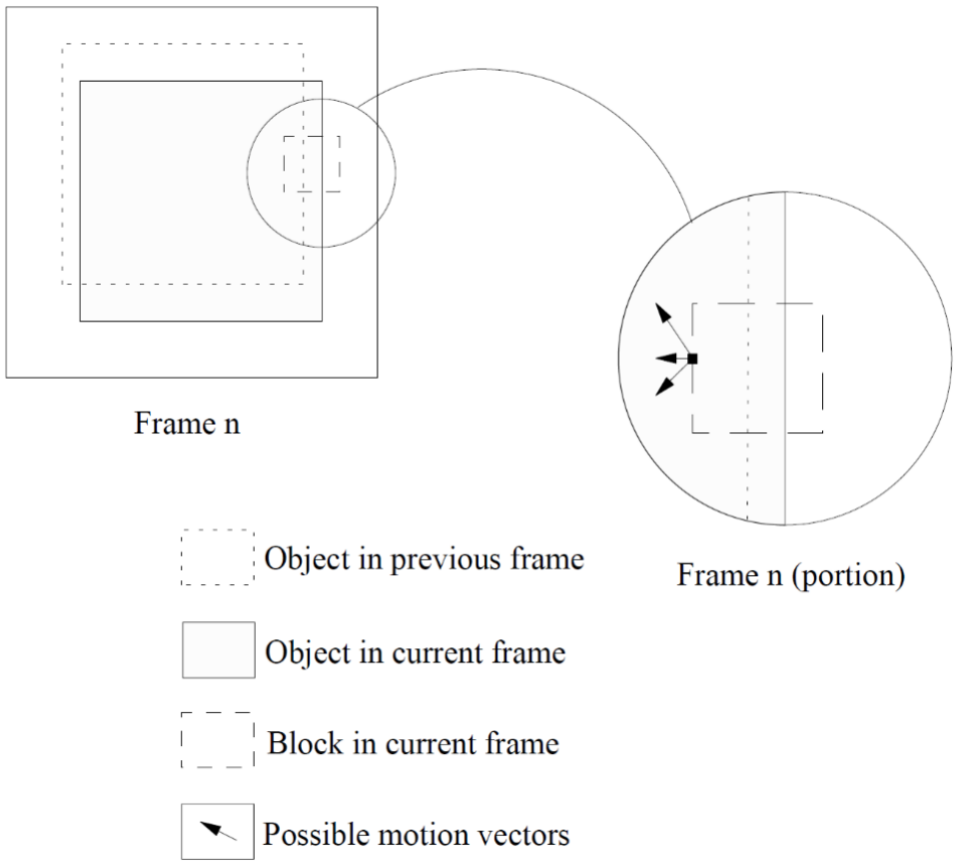
This can be addressed by using image interpolation (usually bilinear) to evaluate the pixels values at arbitrary locations $I_{n-1}(\mathbf{x} + \mathbf{d})$.

This is not so much a problem with the iterative sub-sampled schemes such as Tree-Step or Cross-Search Block-Matching, as it can be obtain with one or two further search steps. It will, however, considerably increase the candidate list in a Full Search BM.

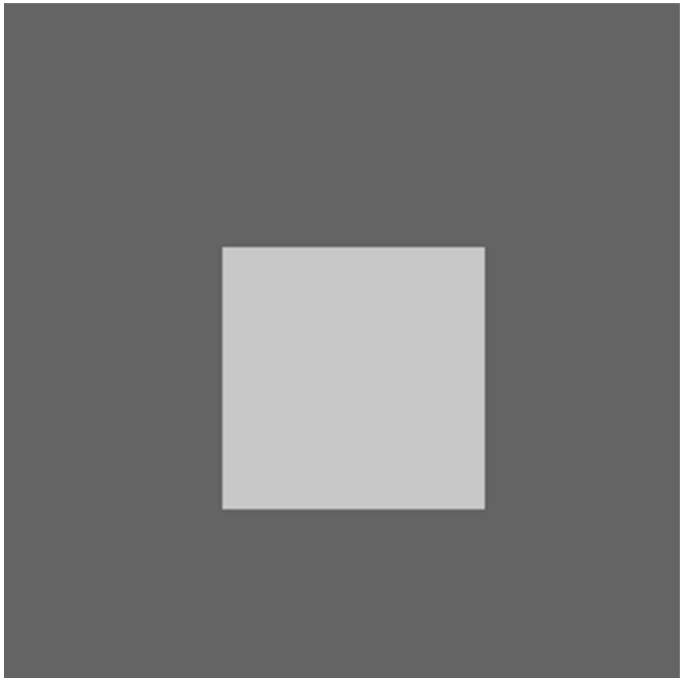
Note: going from pixel accurate to sub-pixelic accurate motion was one of the major cause of improvement when we went from MPEG2 to MPEG4.

# Aperture

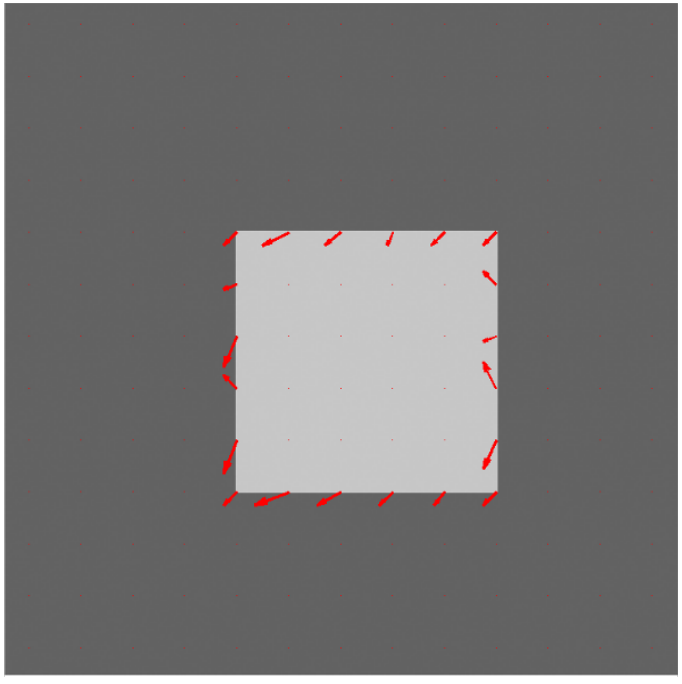
A fundamental fact about our model of motion is that it cannot be solved using one observation at one pixel alone. The problem of matching the brightness is fundamentally under-constrained and there is ambiguity in motion estimation when the block size is too small as multiple candidate vectors can have the same error.



# Aperture



prev frame



curr frame

The only accurate vectors are at object corners. At edges, only the component of motion perpendicular to the edge is correct.

# Aperture: Uniform Areas, Edges and Corners

Aperture Effect is thus associated with regions of uniform intensity and straight edges.

Corners are immune to it and are good features for motion estimation. Feature detectors, such as Harris, SIFT, Shi-Tomasi etc., build on this and detect and track corners through a sequence. These trackers are used for tracking and matching applications in computer vision.

# Aperture: Block Size

Note that the Aperture problem affects all types of motion estimator.

This can be mitigated by increasing the block size. With larger blocks, there is a greater chance that the block contain unique features, and it is more likely that the vector that minimises the DFD is indeed unique and is the correct motion vector.

# Aperture: Block Size

Unfortunately, increasing the block size is also problematic. It means lowering the resolution of the motion field. A typical issue would be at object boundaries. A larger block is more likely to encompass two objects that undergo different motions (eg. foreground object vs. background scene). In that case assigning a unique vector to the block is erroneous, as there are in fact at least two vectors.

# Motion Failures: Pathological Motion

More generally, the idea that you can always find a correspondence in another frame is not always correct. There are many situations where a match cannot be found (typically 3-15% of an image).

- Fast Motion – eg. the size of the motion bigger than the search window.
- Occlusions – eg. the object is not visible in the other frame.
- transparency/Reflections – there is in fact 2 motion vectors per pixel
- non-rigid objects – eg. Hair, flames. The notion of “motion” can be dubious for some fluids.
- motion blur – the object is moving too fast and appears blurred.

# Exercise

Where will motion success/fail?

# Gradient-Based Motion Estimation

# Gradient-Based Motion Estimation

Gradient-Based Motion Estimation is a class of algorithms that allows us to efficiently find the optimal displacement, provided that the displacement is small. They offer a considerable numerical advantage over direct search methods.

# Small Motion - Assumption

Recall the motion equation:

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x}))$$

We are going to assume that $\mathbf{d}_{n,n-1}(\mathbf{x}) \approx \mathbf{d}_0(\mathbf{x})$, or, more precisely, that $\mathbf{d}_{n,n-1}(\mathbf{x}) = \mathbf{d}_0(\mathbf{x}) + \mathbf{d}(\mathbf{x})$, where $\mathbf{d}(\mathbf{x})$ is a small update. Then, using a Taylor expansion allows us to externalise the term $\mathbf{d}(\mathbf{x})$:

$$I_n(\mathbf{x}) \approx I_{n-1}(\mathbf{x} + \mathbf{d}_0) + \left(\frac{\partial I_{n-1}}{\partial \mathbf{x}}(\mathbf{x} + \mathbf{d}_0)\right)^\top \mathbf{d}(\mathbf{x})$$

where $\frac{\partial I_{n-1}}{\partial \mathbf{x}} = \left(\frac{dI_{n-1}}{dx}, \frac{dI_{n-1}}{dy}\right)$ is the gradient of $I_{n-1}$.

# Small Motion - Linear Model

With Block Matching, the motion vector is assumed to be the same for the entire block. Using $\mathbf{x} = (x, y)$, $\mathbf{d_0} = (u_0, v_0)$ and $\mathbf{d} = (u, v)$ we can write this model within the block as:

$$I_n(x, y) \approx I_{n-1}(x + u_0, y + v_0) + u\frac{dI_{n-1}}{dx}(x + u_0, y + v_0) + v\frac{dI_{n-1}}{dy}(x + u_0, y + v_0)$$

This is a linear model and finding the best $(u, v)$ is finding the least square solution over the block.

# Small Motion - Solution

Namely, the block matching criteria can be written as:

$$E_{\text{Block}}(\mathbf{d}) = \|\mathbf{A}\mathbf{d} - b\|_2^2 = (\mathbf{A}\mathbf{d} - \mathbf{b})^{\mathsf{T}}(\mathbf{A}\mathbf{d} - \mathbf{b})$$

with

$$A = \begin{bmatrix} \frac{dI_{n-1}}{dx}(x_1 + u_0, y_1 + v_0) & \frac{dI_{n-1}}{dy}(x_1 + u_0, y_1 + v_0) \\ \vdots & \vdots \\ \frac{dI_{n-1}}{dx}(x_{N^2} + u_0, y_{N^2} + v_0) & \frac{dI_{n-1}}{dy}(x_{N^2} + u_0, y_{N^2} + v_0) \end{bmatrix}$$

$$b = \begin{bmatrix} I_{n-1}(x_1 + u_0, y_1 + v_0) - I_n(x_1, y_1) \\ \vdots \\ I_{n-1}(x_{N^2} + u_0, y_{N^2} + v_0) - I_n(x_{N^2}, y_{N^2}) \end{bmatrix}$$

The Least Square Solution can be found as:

$$\hat{\mathbf{d}} = \left(\mathbf{A}^{\mathsf{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{b}$$

# Gradient Based Iterations

You can first start by precomputing the gradient $\frac{dI_{n-1}}{dx}(x,y)$ and $\frac{dI_{n-1}}{dy}(x,y)$ using for instance simple forward difference as approximation:

$$\frac{dI_{n-1}}{dx}(x,y) \approx I_{n-1}(x+1,y) - I_{n-1}(x,y)$$
$$\frac{dI_{n-1}}{dy}(x,y) \approx I_{n-1}(x,y+1) - I_{n-1}(x,y)$$

.

Start at $\mathbf{d_0} = (u=0, v=0)$ and then iterate as follows:

1. interpolate $\frac{dI_{n-1}}{dx}$ and $\frac{dI_{n-1}}{dy}$ at $\mathbf{x} + \mathbf{d_0}$ using bilinear interpolation.
2. update $\mathbf{d_0} \leftarrow \mathbf{d_0} + \left(\mathbf{A}^\mathsf{T}\mathbf{A}\right)^{-1}\mathbf{A}^\mathsf{T}\mathbf{b}$

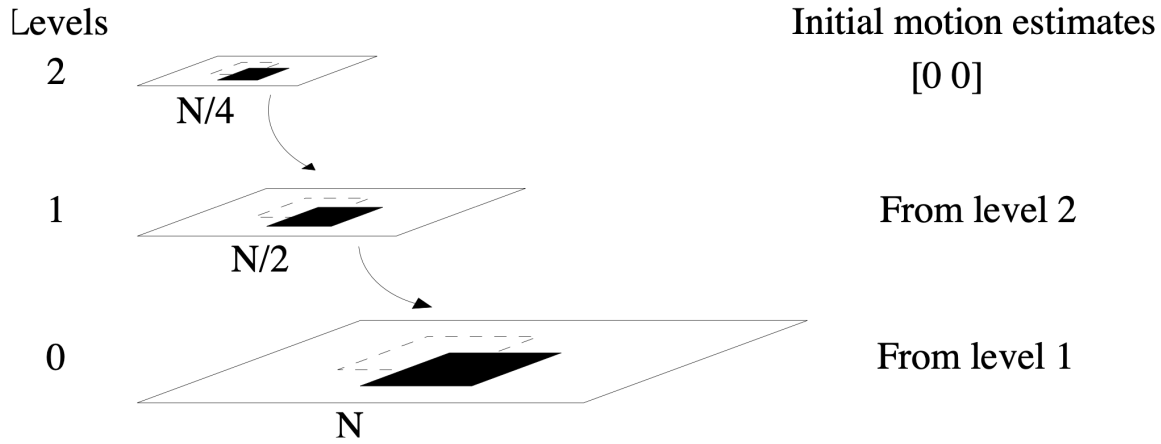
# Complexity

The advantage of this technique is that a sub-pixelic accuracy can be obtained with a few iterations. There is thus no need to make the full $(2w+1)^2$ evaluations of the full search BM.

# Large Displacements and Multi-Scale Approaches

The important drawback with this approach, however, is that the Taylor series expansion is only valid over very small distances (typically $\|\mathbf{d}\| < 4$).

To overcome this problem, a mutli-resolution approach is usually adopted, starting at a coarser resolution of the images, where large displacements correspond to fewer pixels. The updates are then propagated to the higher resolution to be refined.



# Aperture Effect?

Note that that the aperture problem still exists. In fact, the aperture effect now manifests itself in the invertibility of $\mathbf{A}^\mathsf{T}\mathbf{A}$. Typically $\mathbf{A}^\mathsf{T}\mathbf{A}$ is singular at edges or on uniform area but it is invertible at corners.

Taking $\mathbf{d_0} = (0, 0)$, gives us the following classic $2 \times 2$ matrix:

$$\mathbf{A}^\mathsf{T}\mathbf{A} = \sum_{\mathbf{x}} \begin{pmatrix} \frac{dI_{n-1}}{dx}(\mathbf{x})\frac{dI_{n-1}}{dx}(\mathbf{x}) & \frac{dI_{n-1}}{dx}(\mathbf{x})\frac{dI_{n-1}}{dy}(\mathbf{x}) \\ \frac{dI_{n-1}}{dx}(\mathbf{x})\frac{dI_{n-1}}{dy}(\mathbf{x}) & \frac{dI_{n-1}}{dy}(\mathbf{x})\frac{dI_{n-1}}{dy}(\mathbf{x}) \end{pmatrix}$$

As it turns out, studying the eigenvalues $\lambda_1$ and $\lambda_2$ of this symmetric matrix can give a good indication of the cornerness of $I_{n-1}$ at $\mathbf{x}$. For instance, in the KLT feature detector, a pixel is considered to be a corner if both $\lambda_1$ and $\lambda_2$ are bigger than some threshold.

# Going Further - Horn & Schunk

Most Motion Estimation algorithms are based on this gradient-based approach. Most of the research has mainly been around dealing with large displacements and, perhaps most importantly, on how to constrain the *smoothness* of the motion field.

For instance, early work by Horn and Schunk proposed to add a penalty for having a non-smooth motion field as follows:

$$E(\mathbf{d}) = \sum_{\mathbf{x}} (I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}(\mathbf{x})))^2 + \left\| \frac{\partial \mathbf{d}}{\partial \mathbf{x}}(\mathbf{x}) \right\|^2$$

where $\left\| \frac{\partial \mathbf{d}}{\partial \mathbf{x}}(\mathbf{x}) \right\|^2 = (\mathbf{d}(x+1, y) - \mathbf{d}(x, y))^2 + (\mathbf{d}(x, y+1) - \mathbf{d}(x, y))^2$. This added term favours nearby motion vectors that are similar.

Starting from the same gradient derivations, it can be shown that this can be solved using partial differential equation solvers.

# Going Further

Around 2010, new development in convex optimisation allowed to solve for more complex smoothness penalties, such as this TV-L1 formulation:

$$E(\mathbf{d}) = \sum_{\mathbf{x}} \left| I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right| + \left\| \frac{\partial \mathbf{d}}{\partial \mathbf{x}}(\mathbf{x}) \right\|_1$$

The optimisation is here more complex, but new iterative schemes have allowed to find the optimum of such systems. Using the $\|\|_1$ norm instead of $\|\|_2$ as in Horn & Schunk allows us to obtain sharper discontinuities in the motion field.

# Going Further - CNNs

The idea behind all these approaches is that the aperture ambiguity can be partially solved if we have prior knowledge about what the motion field should look like.

Nowadays, motion estimation algorithm heavily rely on Deep Neural Networks. The idea is that, instead of assuming a set formula for the smoothness of the motion field (ie. $\left\| \frac{\partial \mathbf{d}}{\partial \mathbf{x}}(\mathbf{x}) \right\|_1$ should be small), we learn the actual statistics of motion field over large synthetic datasets.

# Motion Estimation - Comparisons

In the next few slides, we show the estimated motion fields of various motion estimation methods on two examples.

# Motion Estimation - Comparison 1

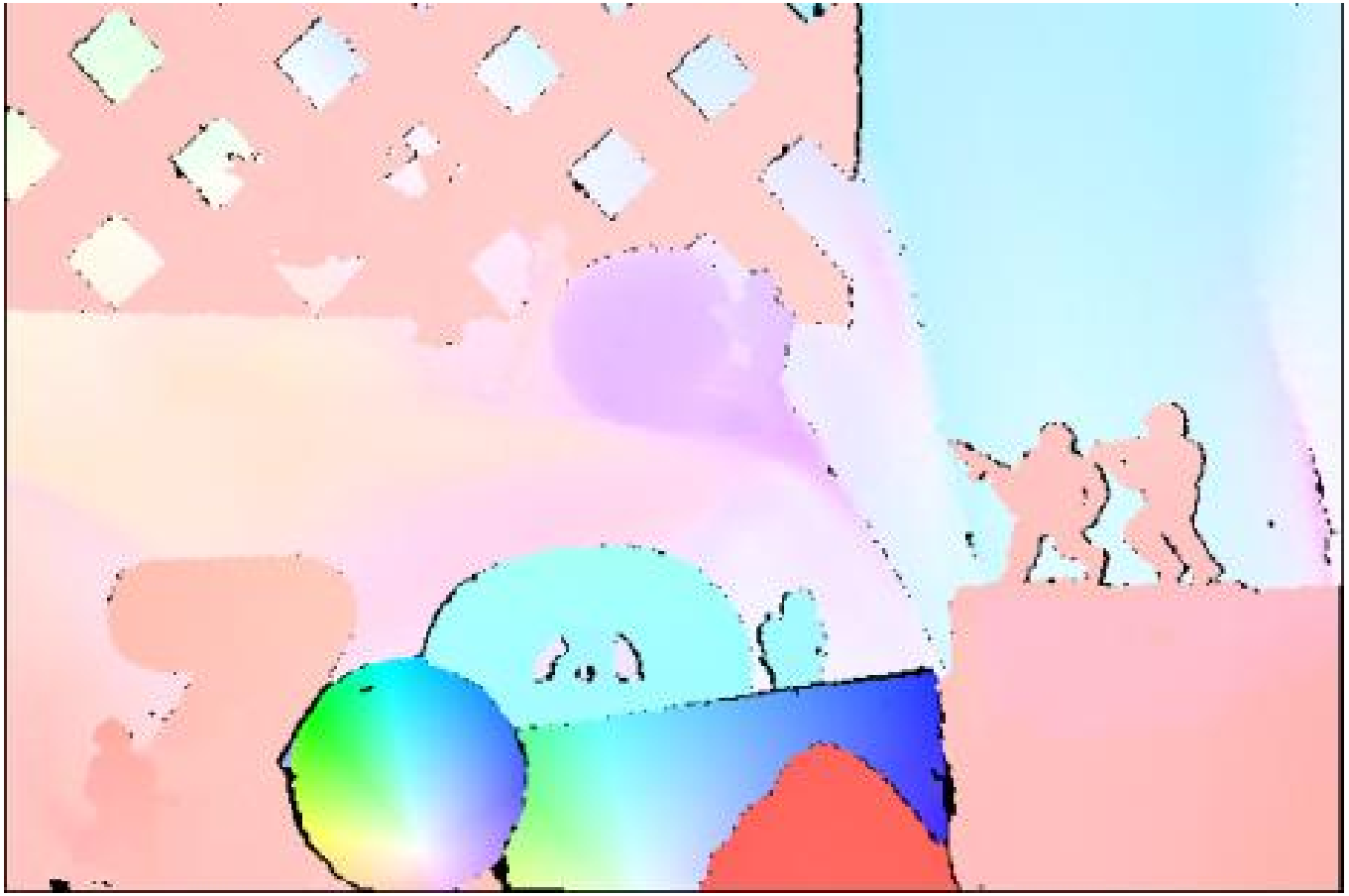
frame $n-1$

# Motion Estimation - Comparison 1

frame $n$

# Motion Estimation - Comparison 1

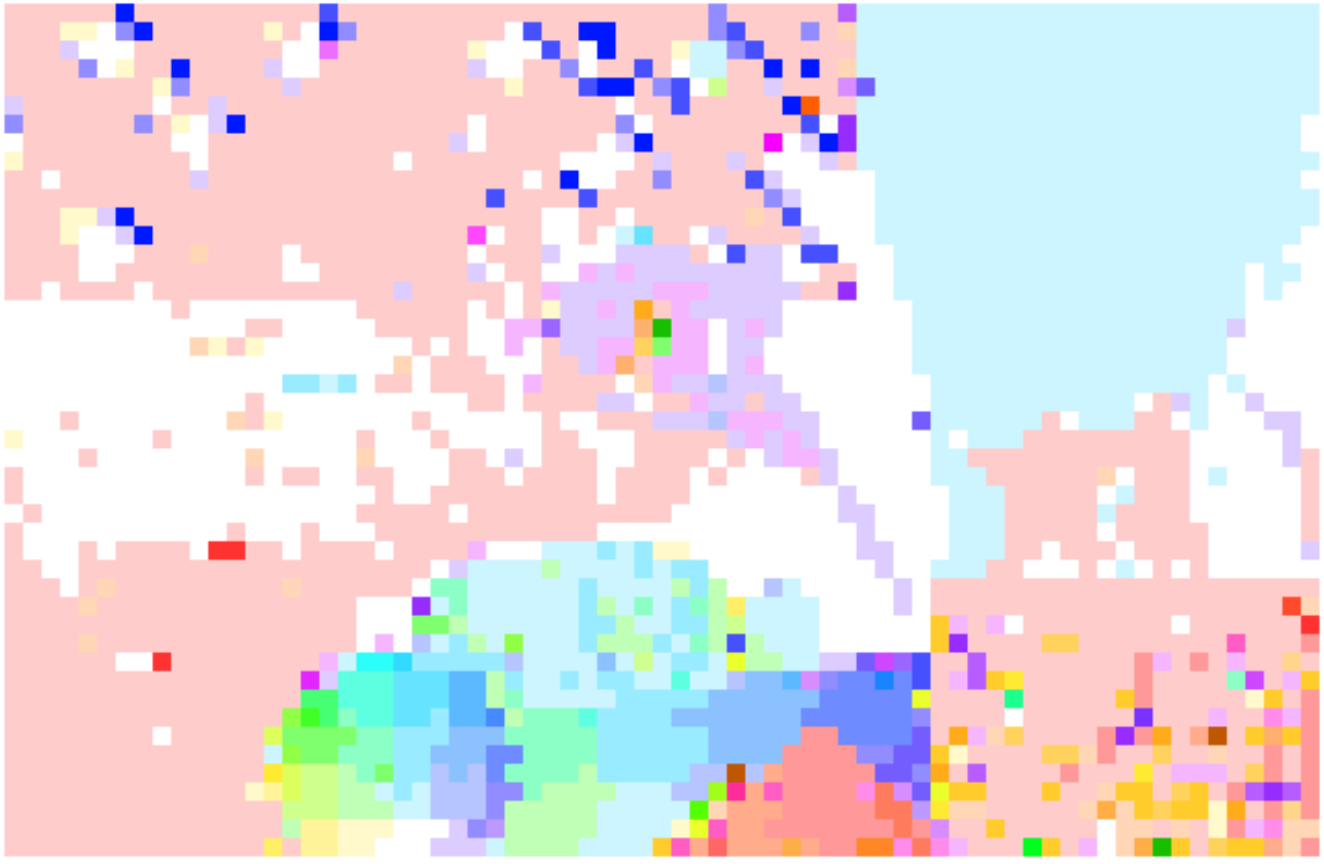
Ground-Truth Motion field (colour coded, hue=direction)



For each pixel we code the (ground-truth) motion vector with a colour (hue=vector direction, saturation= amplitude). The exact colours values don't matter: we only use this image as a point of reference.

# Motion Estimation - Comparison 1

Full Search BM ($8 \times 8$) (colour coded, hue=direction)



As expected the block processing reduces the resolution. Also occlusions and aperture effect cause errors.

# Motion Estimation - Comparison 1

Horn & Schunk (colour coded, hue=direction)

The motion field is defined at each pixel. The accuracy is sub-pixelic. The smoothness allows to avoid many of the BM errors.

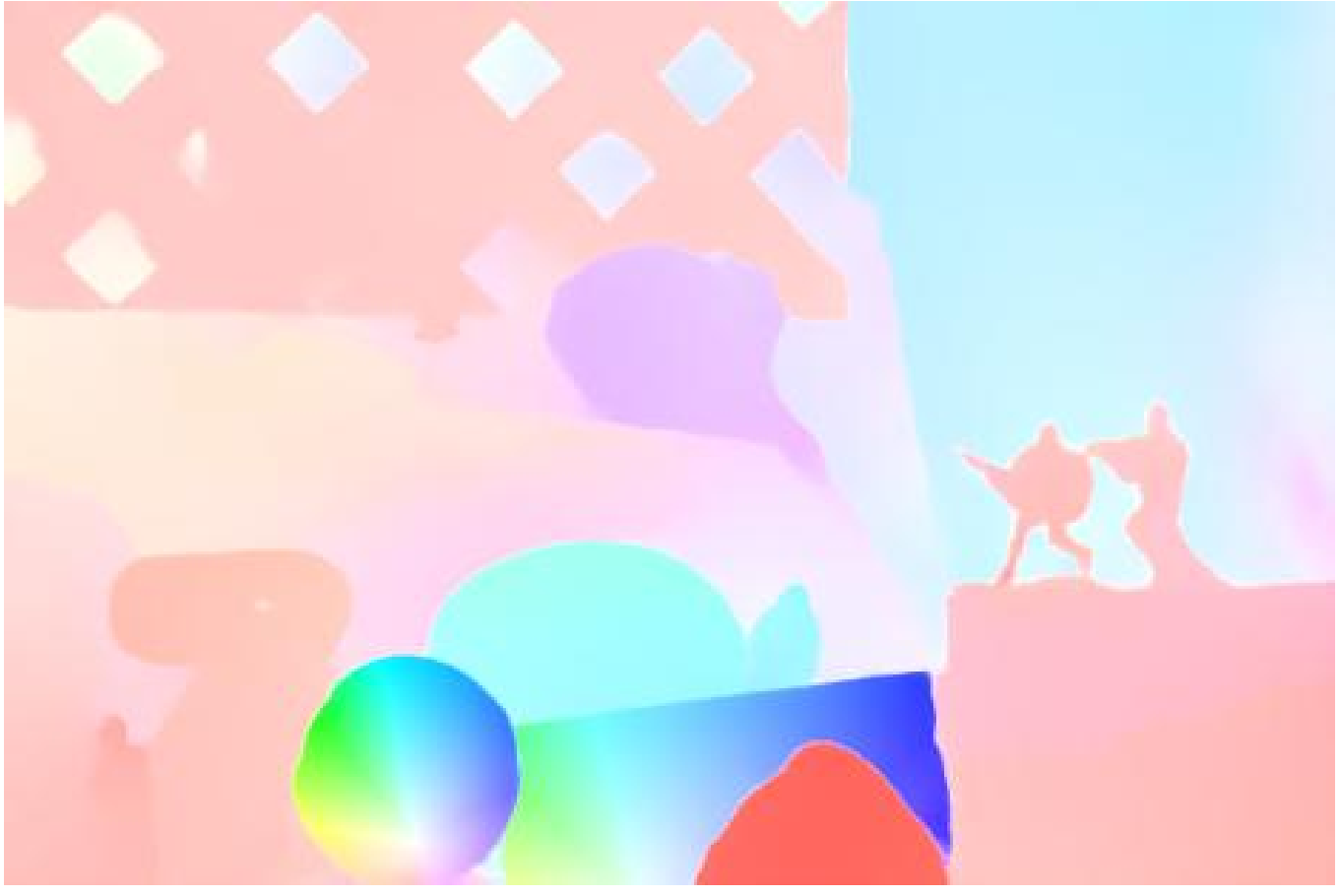
# Motion Estimation - Comparison 1

TV-L1 convex optimisation (colour coded, hue=direction)

Note the sharper boundaries.

# Motion Estimation - Comparison 1

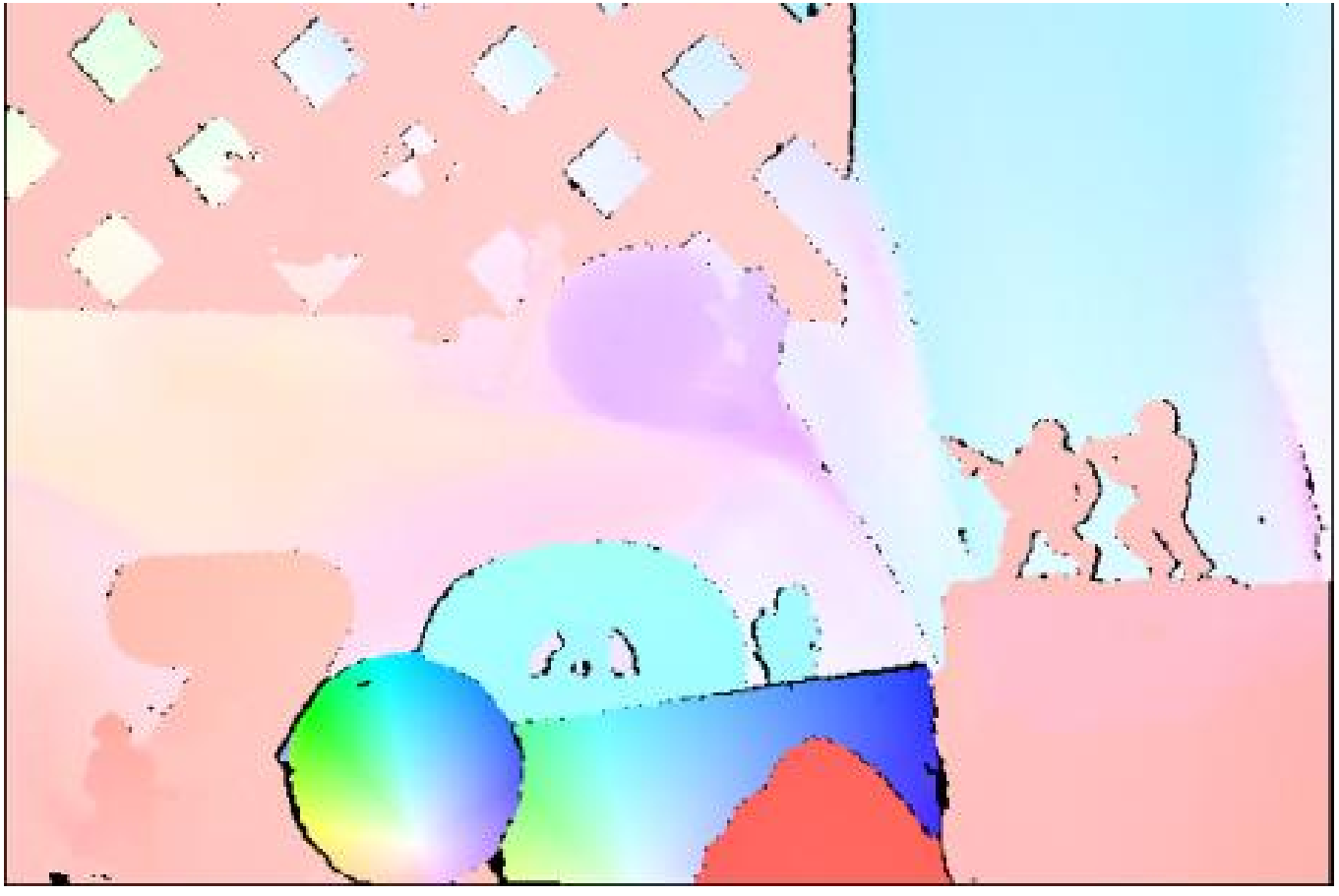
RAFT (Deep Learning method) (colour coded, hue=direction)



Deep Learning rules!

# Motion Estimation - Comparison 1

Ground-Truth Motion field (colour coded, hue=direction)



# Motion Estimation - Comparison 2

frame $n-1$

another example with much larger displacements and complex motion.

# Motion Estimation - Comparison 2

frame $n$

# Motion Estimation - Comparison 2

Ground-Truth Motion field (colour coded, hue=direction)

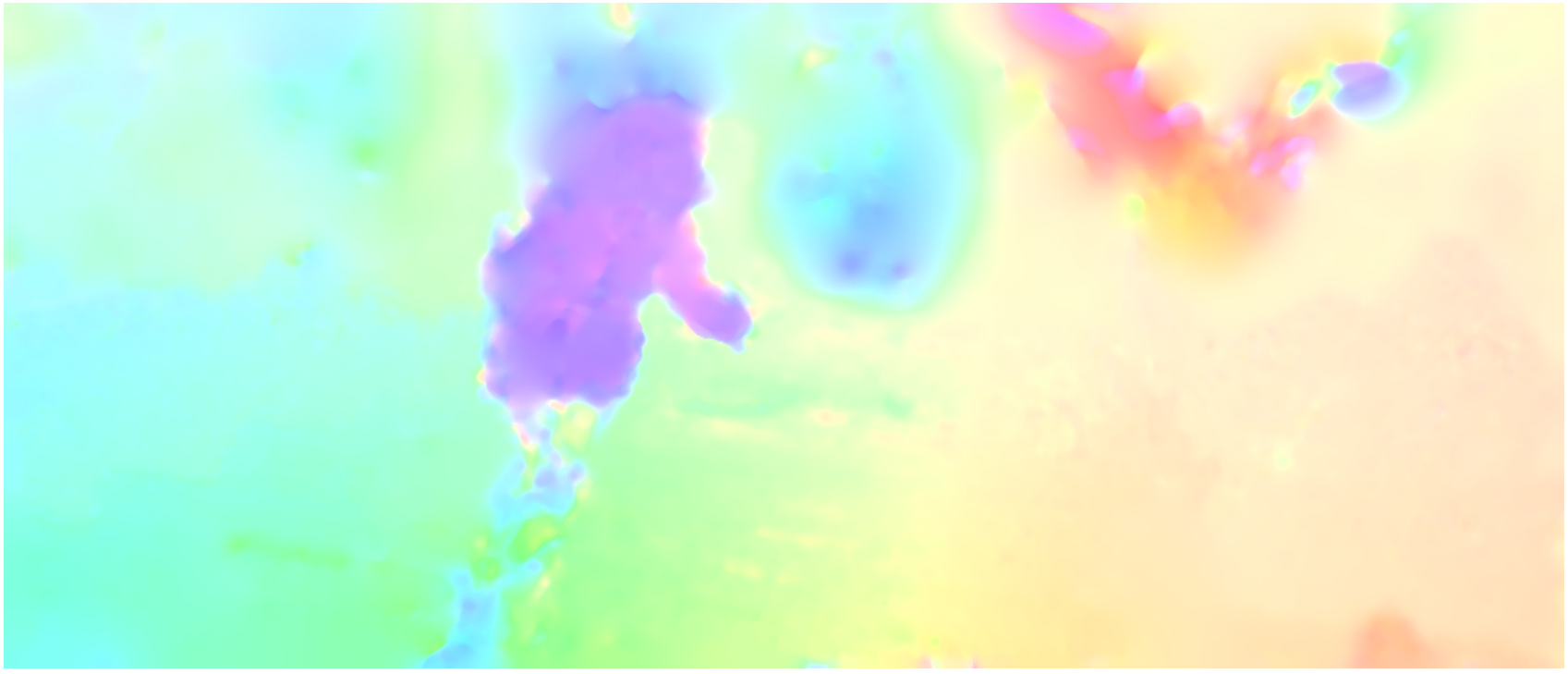
# Motion Estimation - Comparison 2

Full Search BM ($8 \times 8$) (colour coded, hue=direction)

Complex non-translational motion and large displacements cause many errors.

# Motion Estimation - Comparison 2

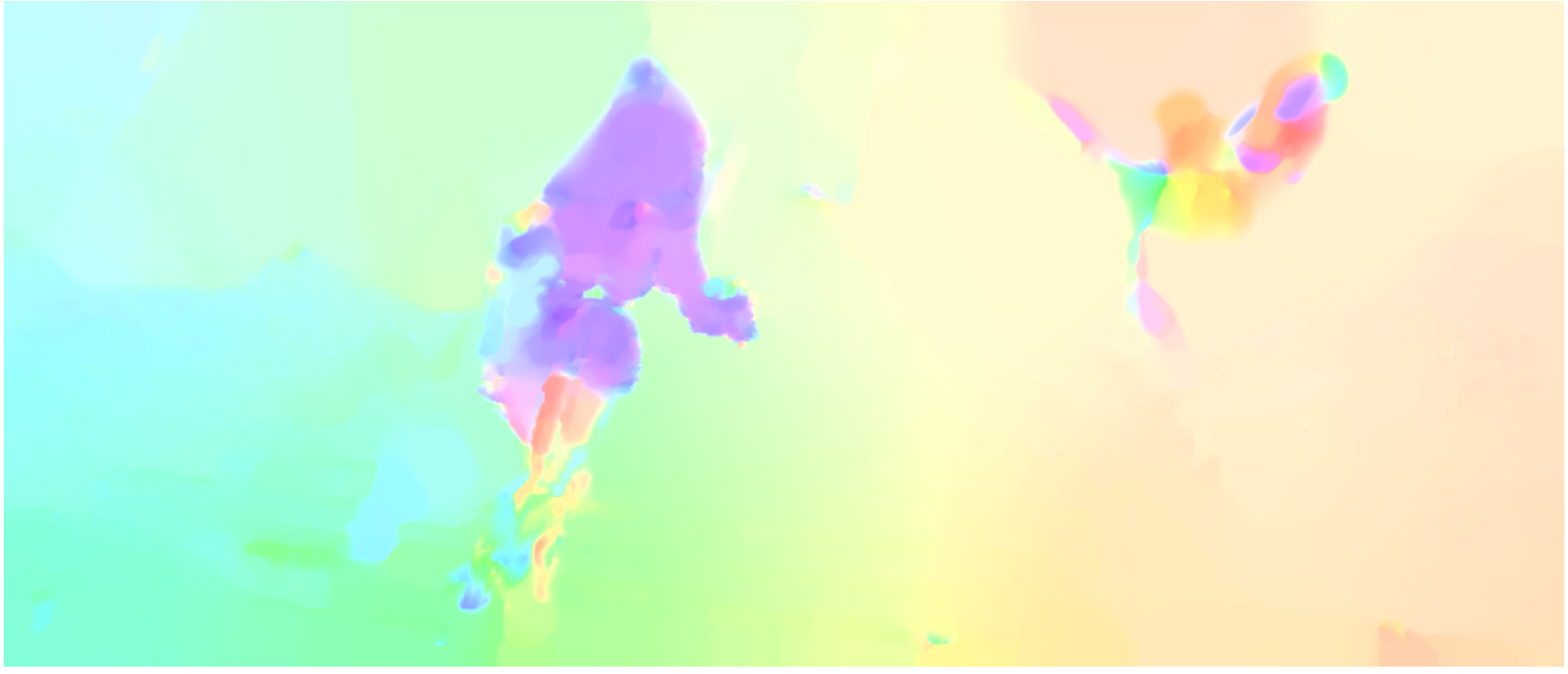
Horn & Schunk (colour coded, hue=direction)



...gradient-based techniques struggle with large displacements.

# Motion Estimation - Comparison 2

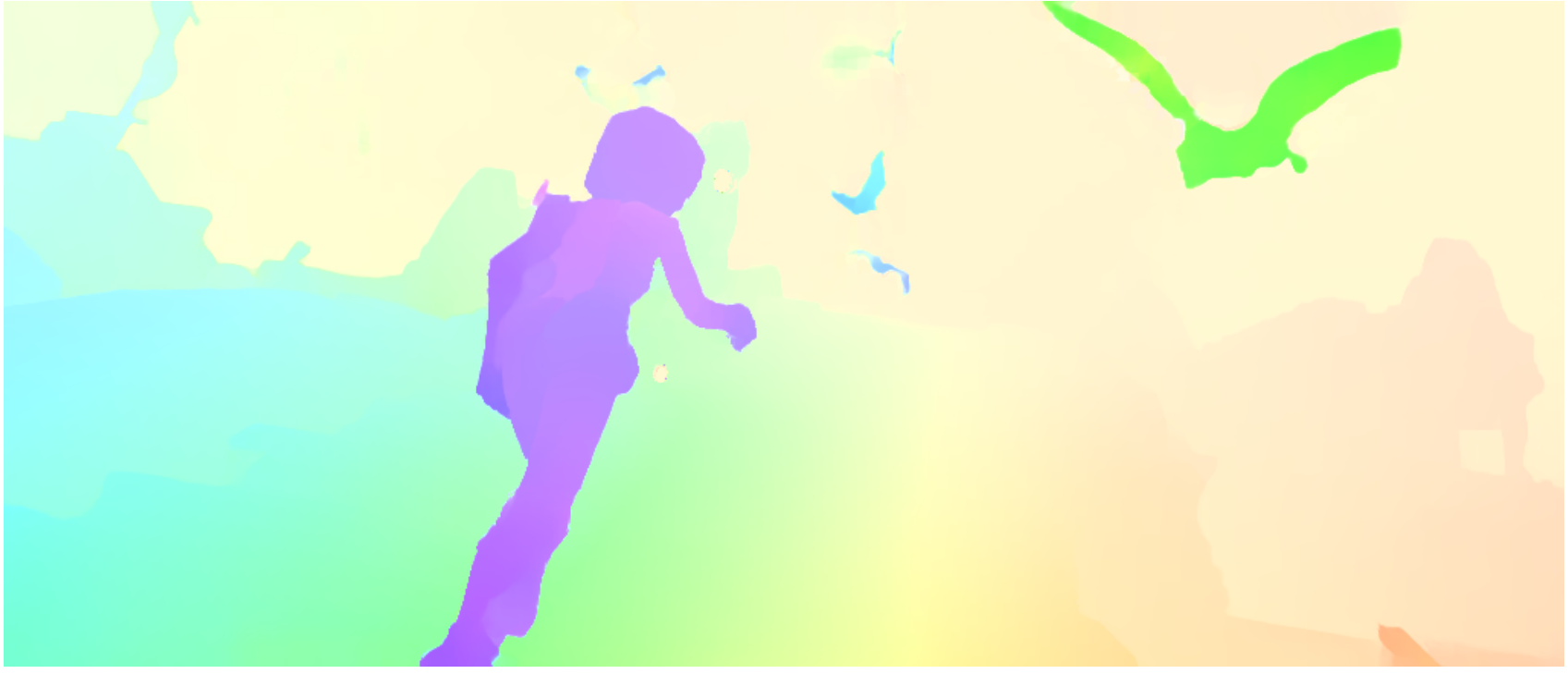
TV-L1 convex optimisation (colour coded, hue=direction)



...sharper boundaries but still fails with large displacements.

# Motion Estimation - Comparison 2

RAFT (Deep Learning method) (colour coded, hue=direction)



impressive!

# Motion Estimation - Comparison 2

Ground-Truth Motion field (colour coded, hue=direction)

# Conclusions

Motion estimation is key for the processing of digital video. There are two main classes of motion estimator, those based on Block Matching and those based on gradient analysis of some kind. All the motion estimators can estimate motion pixel by pixel or on a block basis, however it is understood that the motion equation cannot be solved at a single pixel site alone.

The ambiguity in the solutions is also known as the Aperture problem. Motion estimation is therefore most accurate at corners in the image. The solution to this problem can be mitigated by imposing a smoothness constraint to the motion field, which can be achieved in BM by simply increasing the block size.

Note that not all pixels can have a match (eg. occlusions, motion blur, etc.).