Motion Estimation

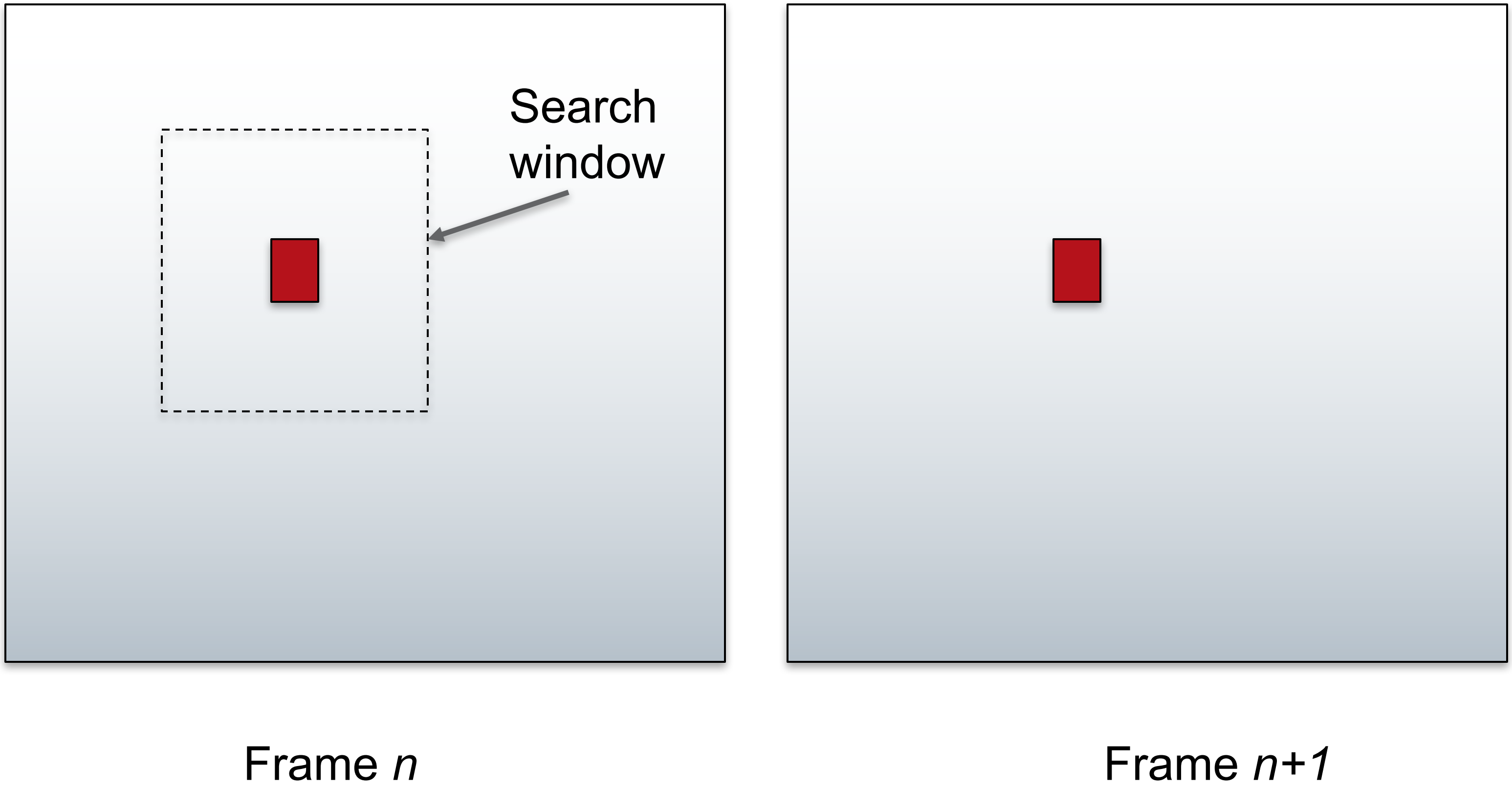
Video

- **Video is a sequence of frames**
- **The largest correlation in video is in the temporal direction**
- **Video compression is reducing the number of bits required to represent a video sequence**
  - H.264 (MP4), MPEG-2, etc.
- **An important step in video compression is predicting frame *n+1* from frame *n***
  - Only the prediction error needs to be transmitted

This prediction requires motion estimation

Motion estimation adds computational complexity

- **A window centered at a block's location in a previously encoded frame is searched**



Search Techniques

- **Exhaustive search**
  - Try every possible match in the search window
  - Extremely computationally demanding
- **Hierarchical search**
  - Create an image pyramid. Search higher levels first, then lower levels
  - Much less computationally demanding
- **Other ad-hoc techniques are sometimes used too**

Hierarchical search (Image pyramid)

CU Boulder

- **An image pyramid is created by successive filtering and sub-sampling (in this case by 2)**

Top of pyramid

Bottom of pyramid

Hierarchical search (3-level pyramid)

- **A 16-by-16 block in the full size image is a 4-by-4 block in the smallest image**
- **A 1 pixel displacement at the top of the pyramid is therefore a 4 pixel displacement at the bottom**
- **Search at the top of the pyramid using 4-by-4 blocks to find a potential match**
  - Use this match to do a refined search one level lower in the pyramid
    - ✓ Use this refinement to seed a search at the very bottom of the pyramid
- **Can massively reduce complexity: but sub-optimal**

Search metric

- **Matches are often ranked by MAE (mean absolute error)**
- **Let k and l denote the location of the upper-left hand corner of the *NxN* block; x and y denote the vector offset**

$$\mathrm{MAE}(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{N} |I_{n-1}(k+i+x, l+j+y) - I_n(k+i, l+j)|$$

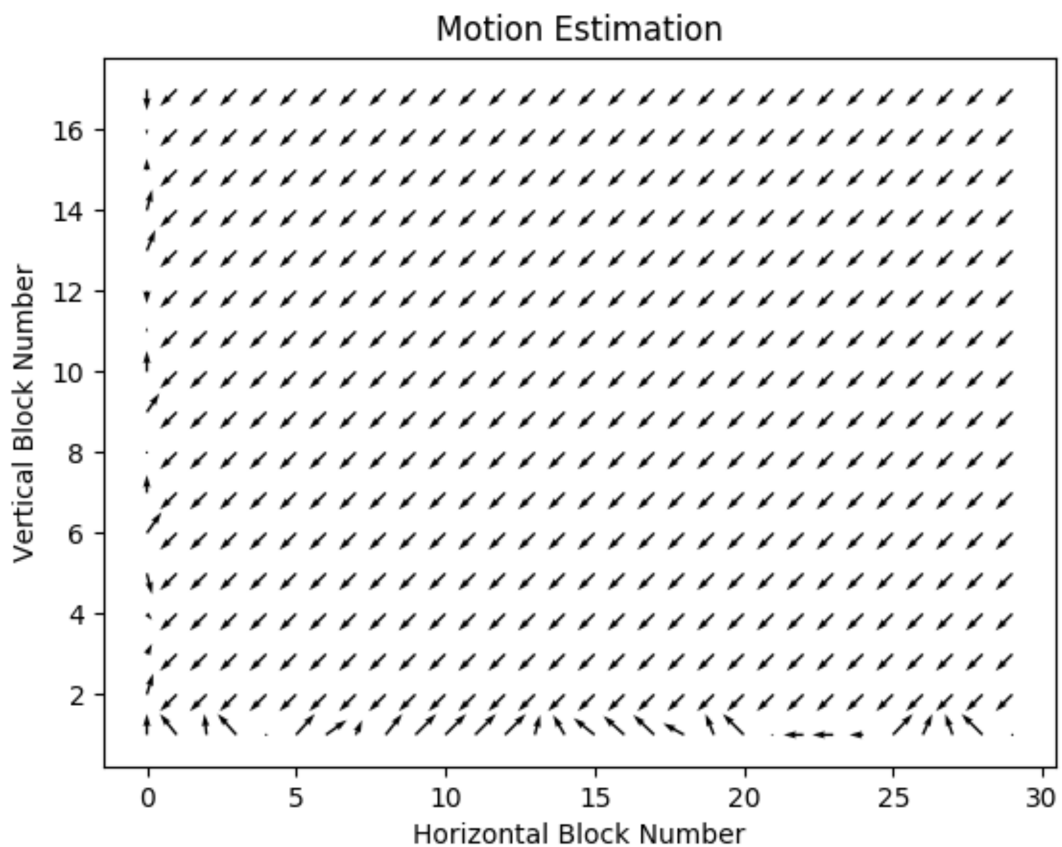
Example

- **Two frames of "video"**
  - In this artificial example, one frame is identical to the other except for an intentionally introduced offset

Example cont.

- **Motion vector field**
  - Vectors point from frame *n+1* to matching block in frame *n*

Note that the black edges of picture cannot be well matched



Example cont.

- **Frame *n*, Frame *n+1*, and the predicted picture**

Frame *n*

Frame *n+1*

Predicted frame

Helper Function.

- **I am providing a function to do the motin searching**
  - me_method.py

```
# block_bw: black/white block from frame n+1 (will search for a
#           match for this block in Frame n)
# im_bw, im_rgb: Frame n in black/white and rgb numpy arrays
# row_start: image row where search in Frame n will start
# col_start: image col where search in Frame n will start
# search_range: +/- pixel range over which a match will be sought
# RETURN VALUES: a prediction for "block" (in color), the best mse,
#                the row/col offsets of the best match (positive offsets
#                are down and to the right)
# NOTE: search range is automatically restricted to ensure that we don't
#       search outside the boundaries of frame n

def motion_match(row_start, col_start, search_range, block_bw, im_bw, im_rgb):
  # row 0 is the top row in the image
  # col 0 is the left most column in the image
```

```
# extract the prediction for the block in rgb format, and return
# relevant information about the best match's offset and mse
pred_block = im_rgb[rt+min_row:rt+min_row+16, cl+min_col:cl+min_col+16]
return pred_block, mse[min_row, min_col], \
       rt + min_row - row_start, cl + min_col - col_start
```