


浙江大学爱丁堡大学联合学院

ZJU-UoE Institute

Lecture 08 - Complex pipelines in image analysis

Tracing, tracking and registration

Nicola Romanò - nicola.romano@ed.ac.uk

Recap

In the past lectures...

We have seen how to

- ✓ Use Numpy, Matplotlib and Scikit-image to **display** and manipulate images
- ✓ **Preprocess images**, e.g. by modifying their histograms, or by using convolutional filters. This allows us to enhance contrast, sharpen or blur the image, find edges, etc.
- ✓ Use **morphological operators** to perform image processing, e.g. erosion, dilation, opening, closing, etc.
- ✓ **Extract features** from an image, to be used for further processing (blob, lines, circles, texture...)
- ✓ **Segment** an image into different regions, e.g. by thresholding, clustering, watershed, etc.

In this lecture...

We are going to put all of this together today to see how we can perform more complex image processing.

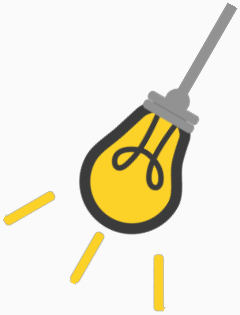
We are going to focus on three example tasks:

- ✓ **Tracing** a structures in an image
- ✓ **Tracking** a moving object in a sequence of images
- ✓ **Registration** of images

Note: this lecture will not go in depth into the details of the implementation of these tasks, but aims to give you an idea of how to build a pipeline to perform them using the tools we have seen so far.

Learning objectives

- Describe the issues associated with tracing tube-like structures
- Describe pipelines for extracting information from these datasets.
- Describe strategies to track objects in a sequence of images
- Describe strategies to register images



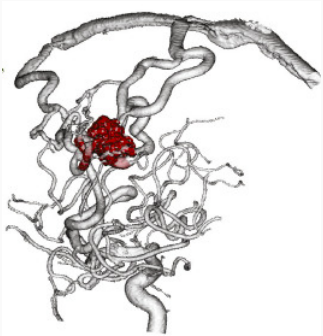
Tracing

What is tracing?

In biomedical imaging, **tracing** is the process of segmenting tube-like structures in images.

Note: some literature calls this tracking (e.g. vessel tracking), which should not be confused with motion tracking.

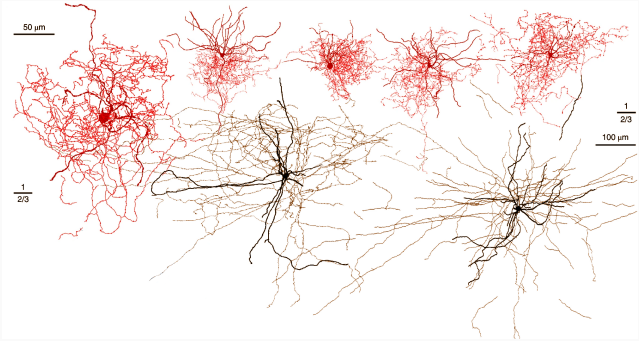
This is often applied to blood vessels and neuronal processes.



Babin et al. 2018 - Arteriovenous malformation in the brain.

Challenges

- Complex morphologies
- Often need to work in 3D
- Can require a lot of computational power

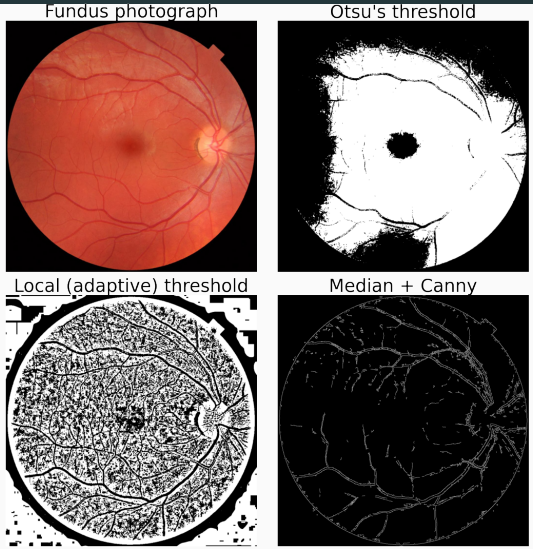


Boldog et al. 2018 - 3D reconstruction of rosehip neurons in the human brain.

Detecting blood vessels

Vessels can be characterised by colour, gradient, and contrast.

Simple segmentation strategies are sub-optimal.



Source: Mikael Häggström 2014, Fundus photograph of a normal right eye.

Filtering with morphological operators

Morphological operators can be often used to improve the image before thresholding.

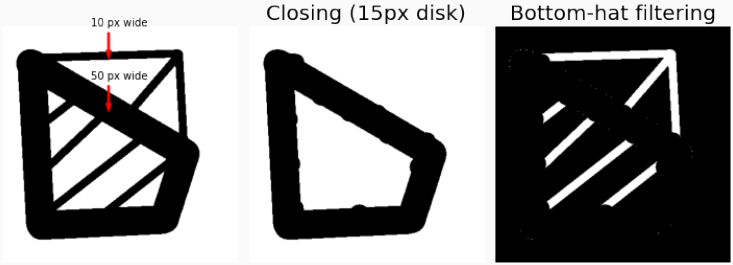
For example, **bottom-hat filtering** has been often used to help in segmenting blood vessels.

Filtering with morphological operators

Morphological operators can be often used to improve the image before thresholding.

For example, **bottom-hat filtering** has been often used to help in segmenting blood vessels.

It involves applying a morphological closing to the image and then subtracting the original image from the result. By using a structural element larger than the structures of interest, we can isolate them from the image.

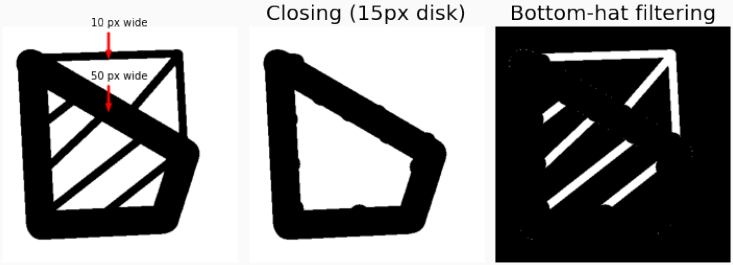


Filtering with morphological operators

Morphological operators can be often used to improve the image before thresholding.

For example, **bottom-hat filtering** has been often used to help in segmenting blood vessels.

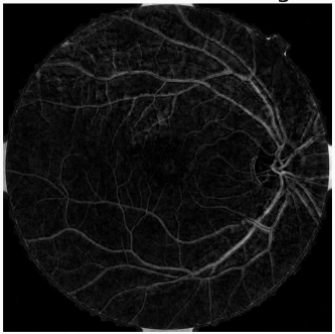
It involves applying a morphological closing to the image and then subtracting the original image from the result. By using a structural element larger than the structures of interest, we can isolate them from the image.



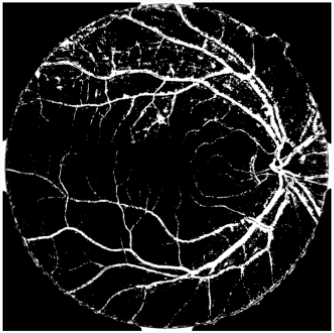
- Bottom-hat filtering extracts objects **darker** than the background.
- Lighter objects can be extracted using **top-hat filtering**
- For top-hat, you subtract a morphological opening of the image from the original image itself.

Bottom-hat filtering of blood vessels

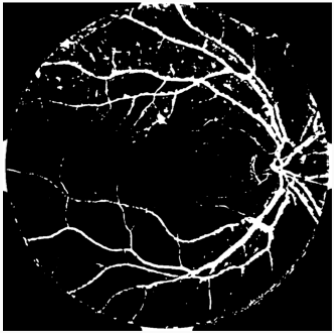
Bottom-hat filtering



BH + Otsu's

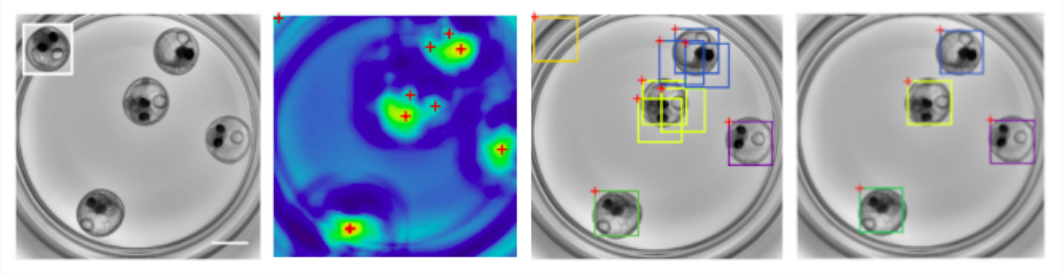


BH + Median + Otsu's



Template matching

Template matching allows finding small parts of an image matching a template image. The cross-correlation between the template and the image is calculated pixel-wise and the maxima are taken as matches.



Thomas and Gehrig, 2020 - Template matching (followed by non-maxima suppression) to identify fish embryos in a plate.

Template matching

Template matching allows finding small parts of an image matching a template image. The cross-correlation between the template and the image is calculated pixel-wise and the maxima are taken as matches.

Template matching can be used for blood vessel detection by assuming that the cross section of the vessels is approximated by a Gaussian.

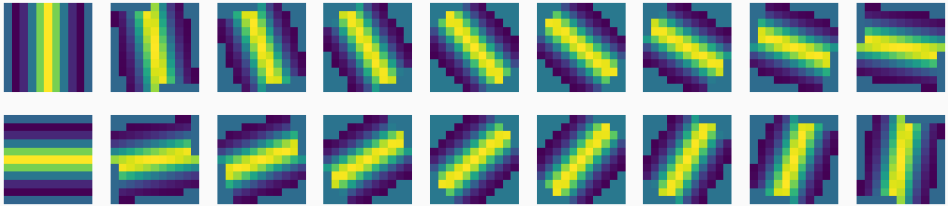
Template matching

Template matching allows finding small parts of an image matching a template image. The cross-correlation between the template and the image is calculated pixel-wise and the maxima are taken as matches.

Template matching can be used for blood vessel detection by assuming that the cross section of the vessels is approximated by a Gaussian.

We design a series of templates, at various angles for piece-wise detection (in pieces of length L).

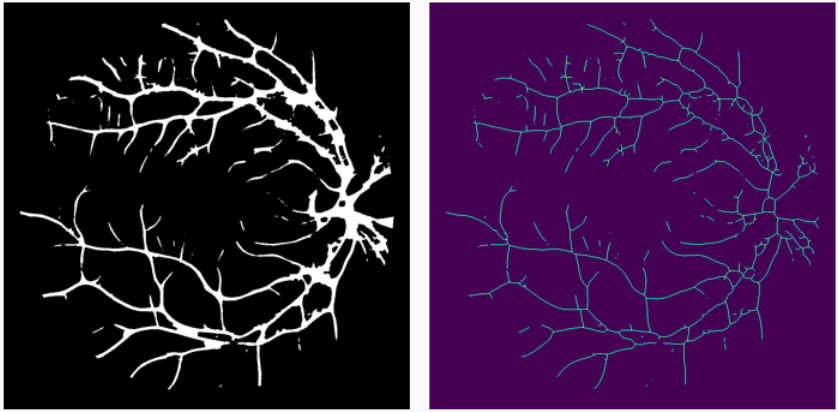
$$K(x, y) = -e^{\frac{-x^2}{2*\sigma_x^2}} \qquad \text{for } |y| \leq \frac{L}{2}$$



Skeletonization

Skeletonization, or center-line extraction is an useful morphological operation that can help delineating tube-like structures in images.

It works by thinning the image repeatedly until no more thinning is possible.



Green channel → bottom-hat filtering → template matching → remove small objects → skeletonization.

From skeleton to graphs and features

Skeletonization can be mapped to a graph, a mathematical structure which allows for easier analysis of the structure.

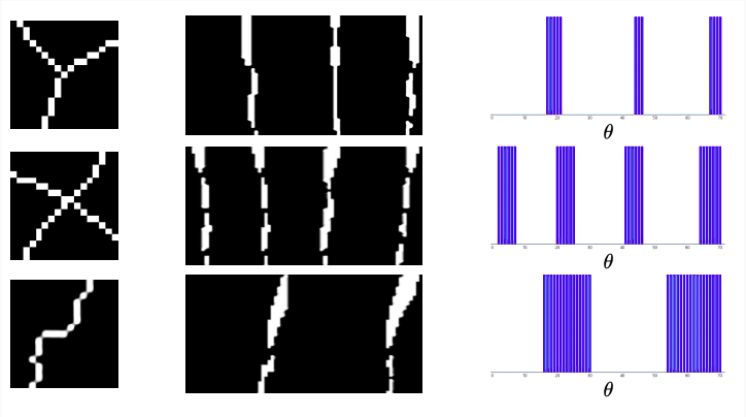
Features can be extracted to find vessel length, number and location of branching points, tortuosity, discriminate arteries and veins etc

HOG features can be used to find the direction and bifurcation points of vessels.

Srinidhi, 2017

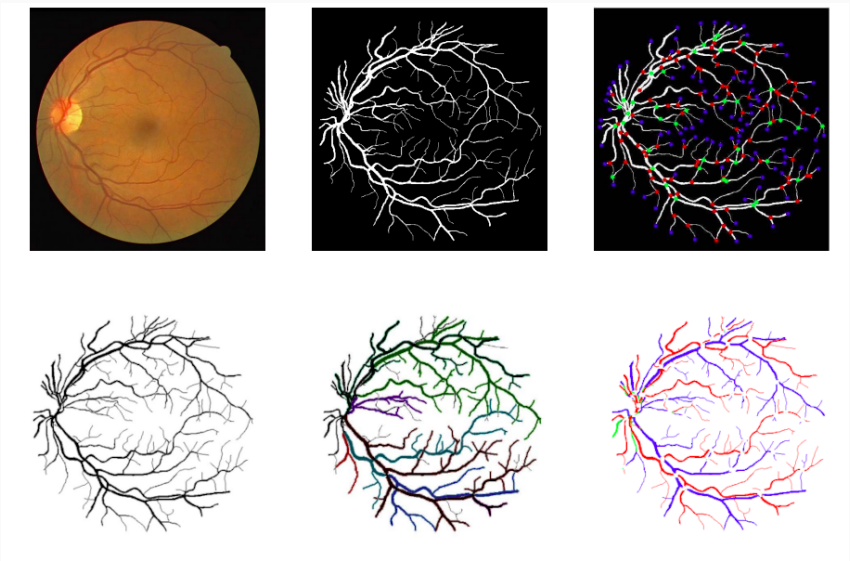
Classification of intersections

Other than HOG features, other strategies have been used for classifying structural features of a vascular network. For example, detection of key-points followed by polar projection has been used to classify branches.



Srinidhi, 2017

Example of a complete pipeline

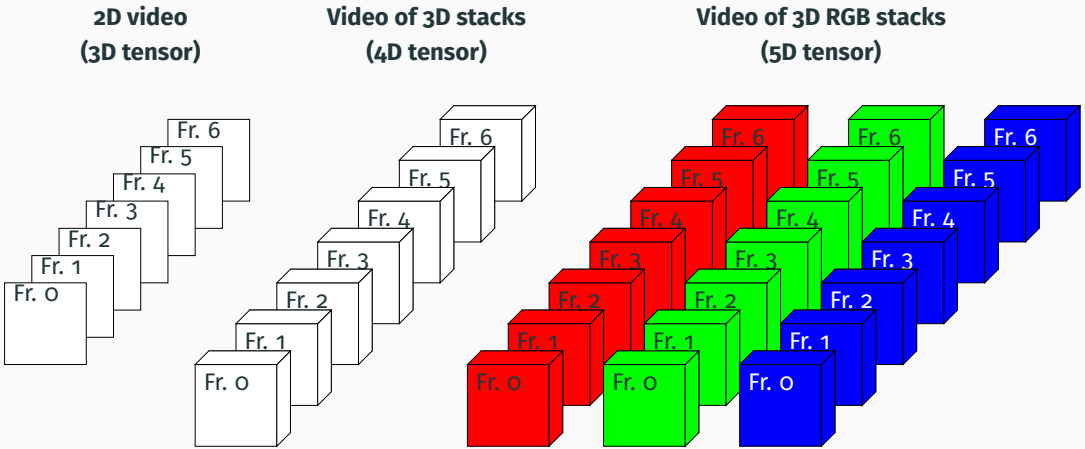


Srinidhi et al. 2109 - extracting colour and other features allows for discrimination of arteries and veins.

Registration and motion tracking

Reminder from lecture 1...

A video is just a sequence of images, that we can store in a tensor with 3 or more dimensions.



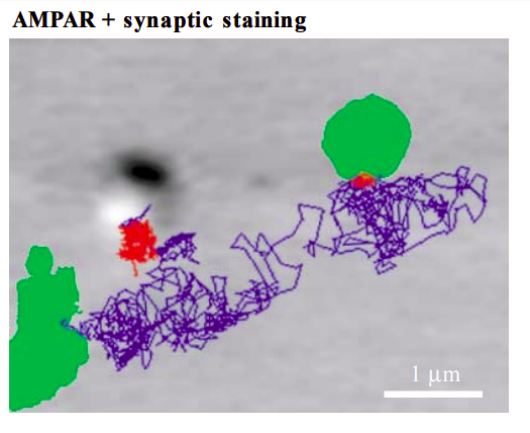
Videos in biomedical imaging

- Many biological processes are dynamic and capturing/analysing them is a challenging task.
- Often tradeoff between acquisition speed and image quality is needed.
- Need analysis methods that are robust to noise and outliers.
- Generally we want to find object(s) of interest in each frame and be able to identify them across frames.

Some examples

Examples of dynamic biological processes captured in videos.

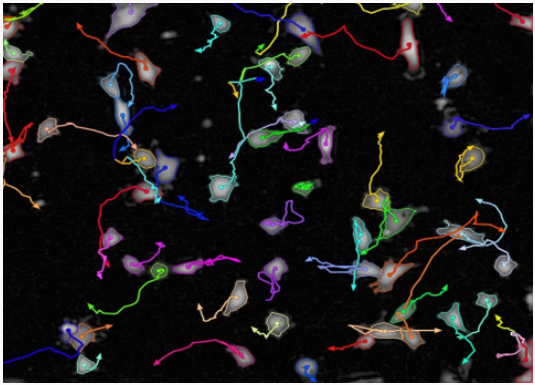
- Tracking single molecules, vesicles, organelles in a cell



Some examples

Examples of dynamic biological processes captured in videos.

- Tracking single molecules, vesicles, organelles in a cell
- Tracking cells in a dish or in a tissue

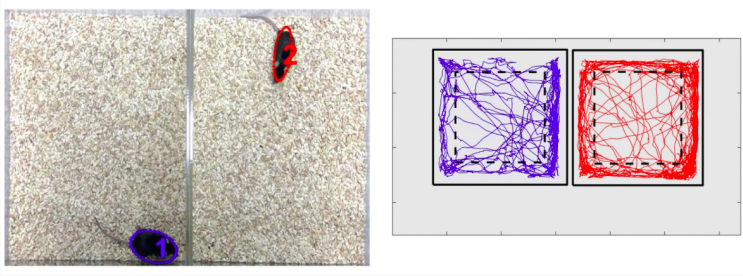


Perkin Elmer - Tracking of cells in a dish.

Some examples

Examples of dynamic biological processes captured in videos.

- Tracking single molecules, vesicles, organelles in a cell
- Tracking cells in a dish or in a tissue
- Tracking a mouse during a behavioural experiment



Mouse tracking in a behavioural experiment.

A general motion tracking pipeline

1. (*optional*) Registration of the video
2. Object detection
3. Object tracking
4. Measurement of object properties over time

Registration

Registration

Registration (or video stabilization) is the process of aligning the video frames to a reference frame; this is done to correct for the movement of the camera or of the sample.

Registration

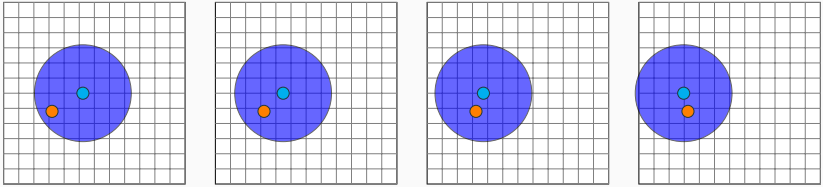
Registration (or video stabilization) is the process of aligning the video frames to a reference frame; this is done to correct for the movement of the camera or of the sample.

For instance, registration is useful for correcting:

- In vivo imaging where breathing artefacts can shift the field of view
- Drifting of the sample under a microscope
- Aligning MRI from the same subject in different sessions

Registration - motivation

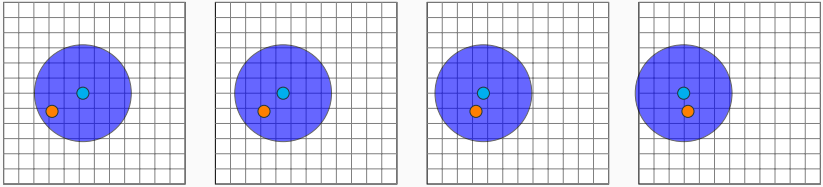
Registration is important because otherwise quantification could be inaccurate.



The orange cell is moving horizontally to the right, while the cyan cell is fixed; because the coverslip is drifting to the left, the coordinates of the orange cell stay fixed, while the cyan cell appers to move to the left.

Registration - motivation

Registration is important because otherwise quantification could be inaccurate.



The orange cell is moving horizontally to the right, while the cyan cell is fixed; because the coverslip is drifting to the left, the coordinates of the orange cell stay fixed, while the cyan cell appers to move to the left.

While it is relatively straightforward to correct for this drift, drift in the z-axis is much more complex, as it changes the plane of focus that is being imaged.

Types of registration algorithms

We can classify registration algorithms into

- **Intensity-based** - compare intensity patterns between the reference and the current frame via correlation
- **Feature-based** - find correspondences between features (points, corners etc) in the reference and the current frame
- **Mixed approaches**

Types of registration algorithms

We can classify registration algorithms into

- **Intensity-based** - compare intensity patterns between the reference and the current frame via correlation
- **Feature-based** - find correspondences between features (points, corners etc) in the reference and the current frame
- **Mixed approaches**

Registration can be performed through:

- **Rigid-body transformations** - translations and rotations only
- **Non-linear transformations** - including affine transformations (scaling, shearing)perspective transformations, curved transformations etc

Rigid-body registration

Rigid-body transformation is the simplest form of registration, consisting only of **translations and rotations**.

Remember from Lecture 2:

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rigid-body registration process

We want to find a transformation (translation+rotation) that maps the reference frame to the current frame.

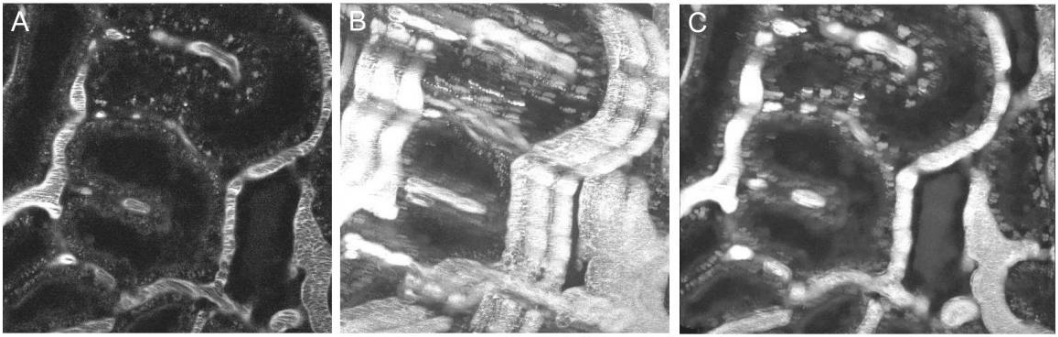
For **intensity-based registration**, given a frame at time t and a reference frame at time t_1, we can find the offset (u, v) that minimizes the MSE:

$$(u, v) = \underset{(u,v)}{\operatorname{argmin}} \sum_x \sum_y (I_{i-1}(x, y) - I_i(x - u, y - v))^2$$

Alternatively, one could maximise the correlation between the two frames:

$$(u, v) = \underset{(u,v)}{\operatorname{argmax}} \sum_x \sum_y (I_{i-1}(x, y) \cdot I_i(x - u, y - v))^2$$

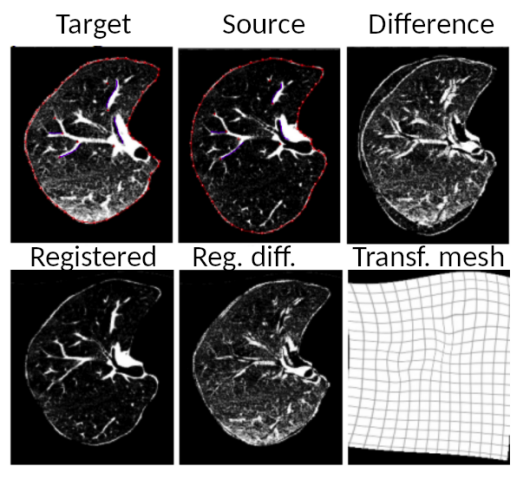
Registration - example



Maximum intensity projection (over the time axis) of a video of kidney vascular flow. Rigid body registration is used to correct for the drift of the sample.

Feature-based registration

Feature-based registration is based on the idea that there are static features in the video, that can be used to match images from one frame to another.



Feature-based non-linear registration of brain MRI images.

Object detection and tracking

Object detection

Object detection in a video is the same process than in a single image.

All the segmentation techniques we have seen so far are applicable in this case as well!

Object tracking

Once we found the object(s) of interest in each frame we want to be able to track them over time.

That is, given n object in a frame, $O_1, O_2, \ldots, O_n$ and n we want to match each of them to the same object in the previous frame.

Nearest-neighbour traking

If the objects are spaced far apart and do not make large movements from frame to frame, we can use a **nearest-neighbour** approach.

Nearest-neighbour traking

If the objects are spaced far apart and do not make large movements from frame to frame, we can use a **nearest-neighbour** approach.

- For each detected object we create a window of size w containing it.
- We find the object in the next frame that is closest to the object of interest within that window.

Nearest-neighbour traking

If the objects are spaced far apart and do not make large movements from frame to frame, we can use a **nearest-neighbour** approach.

- For each detected object we create a window of size w containing it.
- We find the object in the next frame that is closest to the object of interest within that window.
- Improved by predicting the position of the window surrounding the object, based on the displacement in previous frames
- The window could be expanded if no object is found.

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).
- We select the object with the highest score and keep it as a match.

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).
- We select the object with the highest score and keep it as a match.
- If subsequent frames contradict the match, we step back and take the second best object.

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).
- We select the object with the highest score and keep it as a match.
- If subsequent frames contradict the match, we step back and take the second best object.
- We can also add dummy object with a score of zero, which we can use as placeholders for disappearing objects (either ending that track or "putting it on hold" until it reappears).

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).
- We select the object with the highest score and keep it as a match.
- If subsequent frames contradict the match, we step back and take the second best object.
- We can also add dummy object with a score of zero, which we can use as placeholders for disappearing objects (either ending that track or "putting it on hold" until it reappears).

Greedy algorithms

Nearest-neighbour tracking can only handle simple situations. For example, it fails if the object disappears for a certain time.

In this case, we can use a **greedy algorithm** instead.

- For each object O in frame n we define a window surrounding it.
- We look in frame $n + 1$ for any object in the window.
- We score each object according to a metric (e.g. intensity correlation, or position).
- We select the object with the highest score and keep it as a match.
- If subsequent frames contradict the match, we step back and take the second best object.
- We can also add dummy object with a score of zero, which we can use as placeholders for disappearing objects (either ending that track or "putting it on hold" until it reappears).

This type of algorithm works well with relatively sparse and slow moving objects.

Histogram tracking

Another popular algorithm is the **mean-shift** algorithm.

- We compute the histogram for our object of interest

Histogram tracking

Another popular algorithm is the **mean-shift** algorithm.

- We compute the histogram for our object of interest
- We then "back-project" the histogram onto the new frame. This means that we take each pixel in the search window and sum the probability of their intensities in the histogram.

Histogram tracking

Another popular algorithm is the **mean-shift** algorithm.

- We compute the histogram for our object of interest
- We then "back-project" the histogram onto the new frame. This means that we take each pixel in the search window and sum the probability of their intensities in the histogram.
- We then try to maximise this probability by shifting the search window.

Measurement of object properties

Measuring cell properties

Just as we saw in the segmentation lecture we can use the `regionprops` function to measure properties of the detected objects.

We will simply measure these properties in each frame by applying the function multiple times.

Furthermore knowing the position of the object(s) in each frame we can calculate their direction and speed.