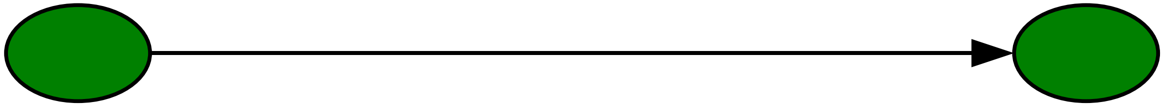
Navguide Automatic place graph generation

Olivier Koch

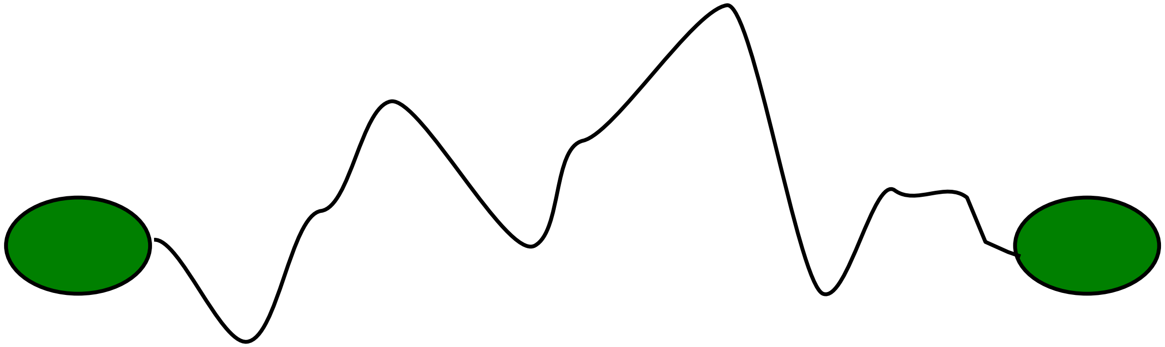
April 4, 2008

Problem Statement

The current system imposes the user to walk in straight forward motion between nodes and to manually create nodes in the map.



Our goal is to allow the user to follow any motion during exploration and to have the system automatically create nodes in the map (in addition to allowing the user to add nodes manually).



Example

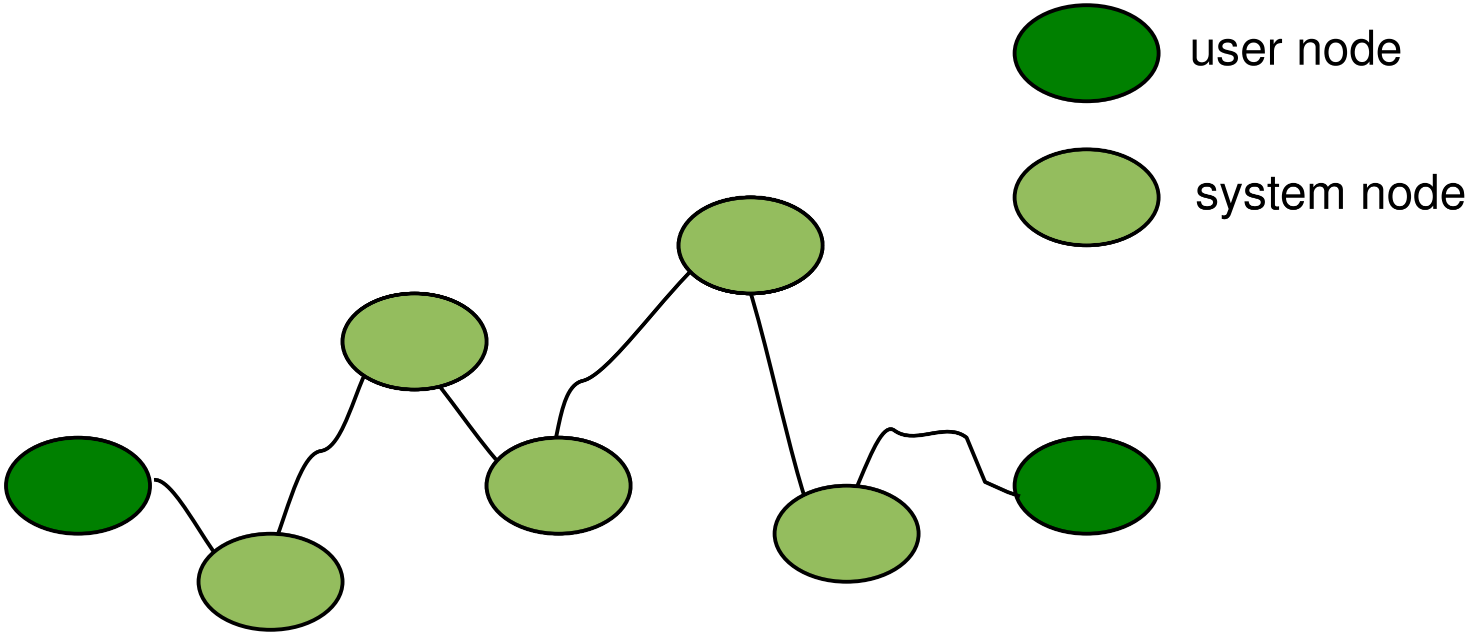
A few examples of situation that shall be handled successfully:

- Abrupt turn at intersection
- Random activity (e.g. searching for an object on the floor)
- Rotation in place then keep going in the same direction

Suggested Solution

Filter out noisy user motion and subdivide an edge into straight line segments.

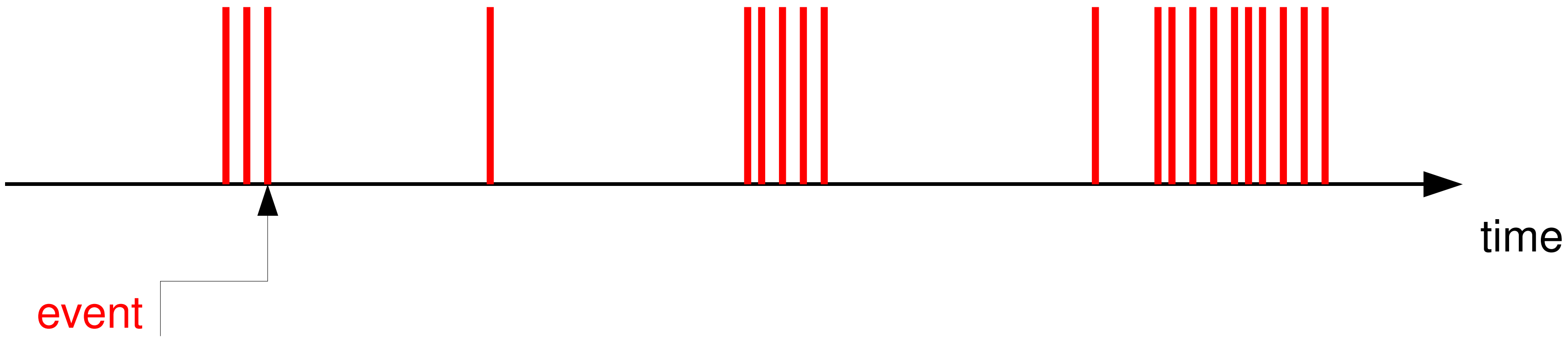
Generate nodes that are not visible to the user but used by the guidance system.



Method Overview

We build a *decider* that checks whether the user motion is straight between consecutive frames.

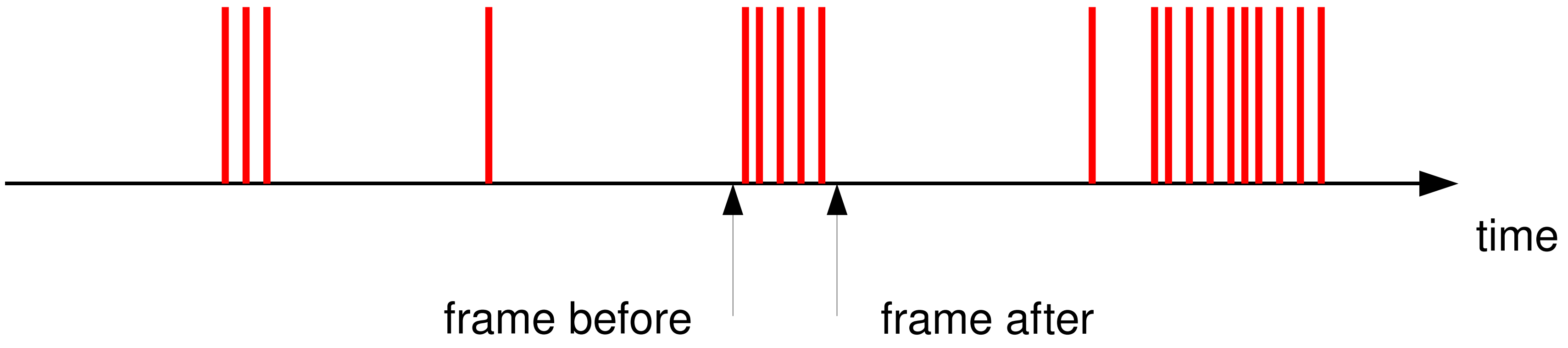
Every time a non-straight motion is detected, the *decider* emits an *event*.



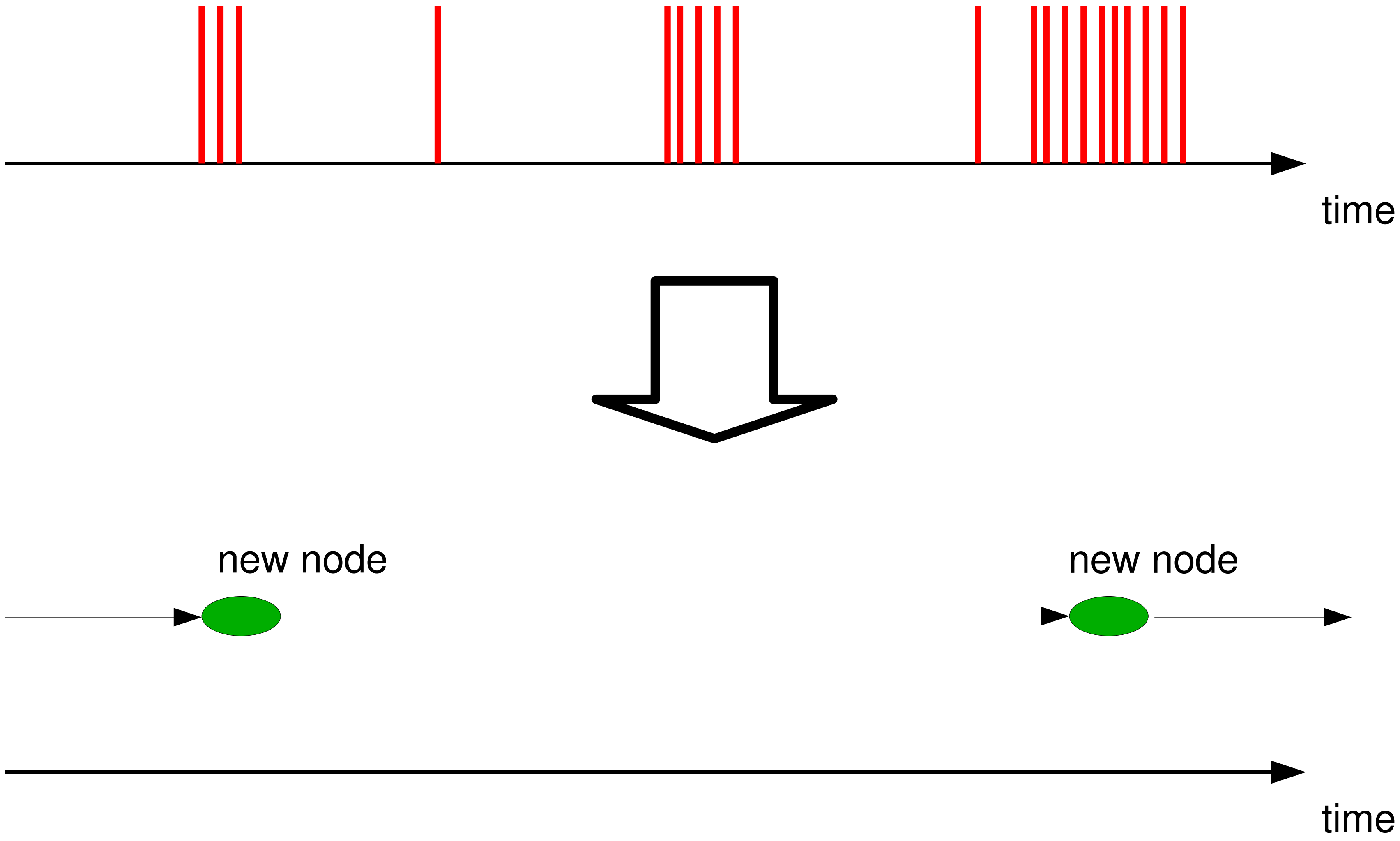
Method Overview

After each series of events, the algorithm checks whether the frame before and the frame after are *equivalent* (i.e high # of feature matches and rotation guidance returns an angle close to 0).

If the two frames are not *equivalent*, the system creates a new node.



Method Overview



Straight motion decider

- Must be fast and reliable
- Makes strong assumption of motion continuity

As a consequence, highly descriptive feature such as SIFT are not appropriate (too slow, too sparse, too descriptive).

Straight motion decider

We extend the idea of the *translation classifier* to dense, non-descriptive features.

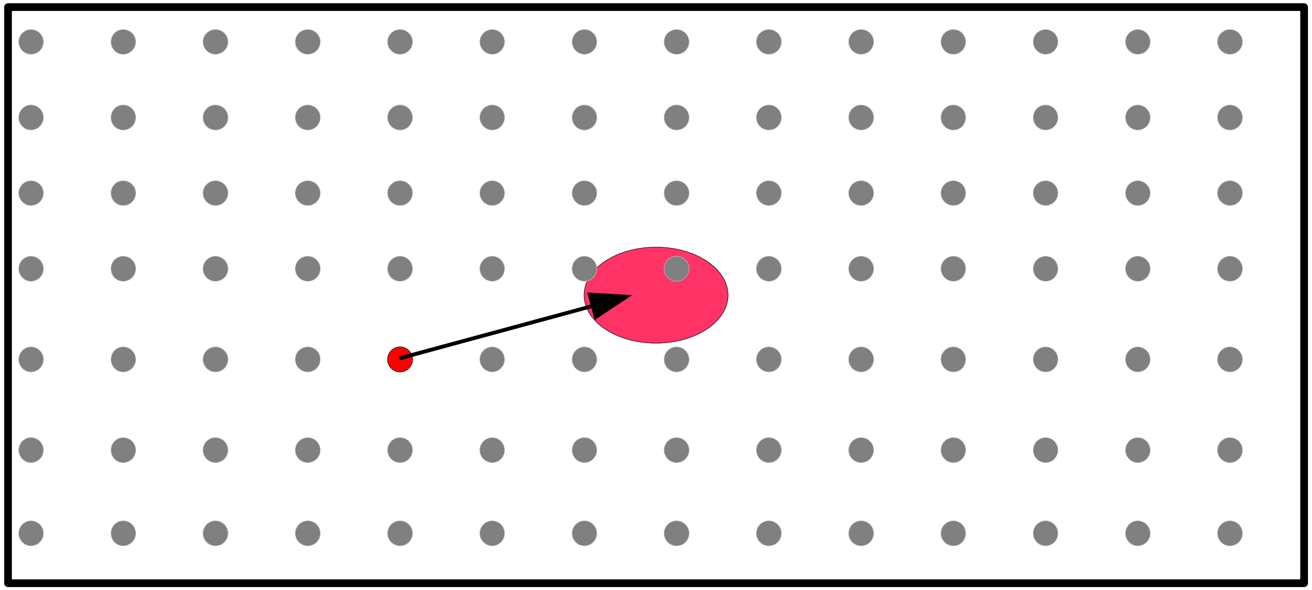
We assume that one or more calibration sequences are provided (user walking straight forward)

We subdivide each camera image into a dense, regular grid of points (say 100x80).

For each pair of successive frames, we keep track of where each grid point goes to on the image (using exhaustive search of normalized cross correlation patches).

Finally, for each grid point, we store the (small) subset of the image where points ended after leaving this grid point. We refer to this subset as the “landing area”.

Straight motion decider



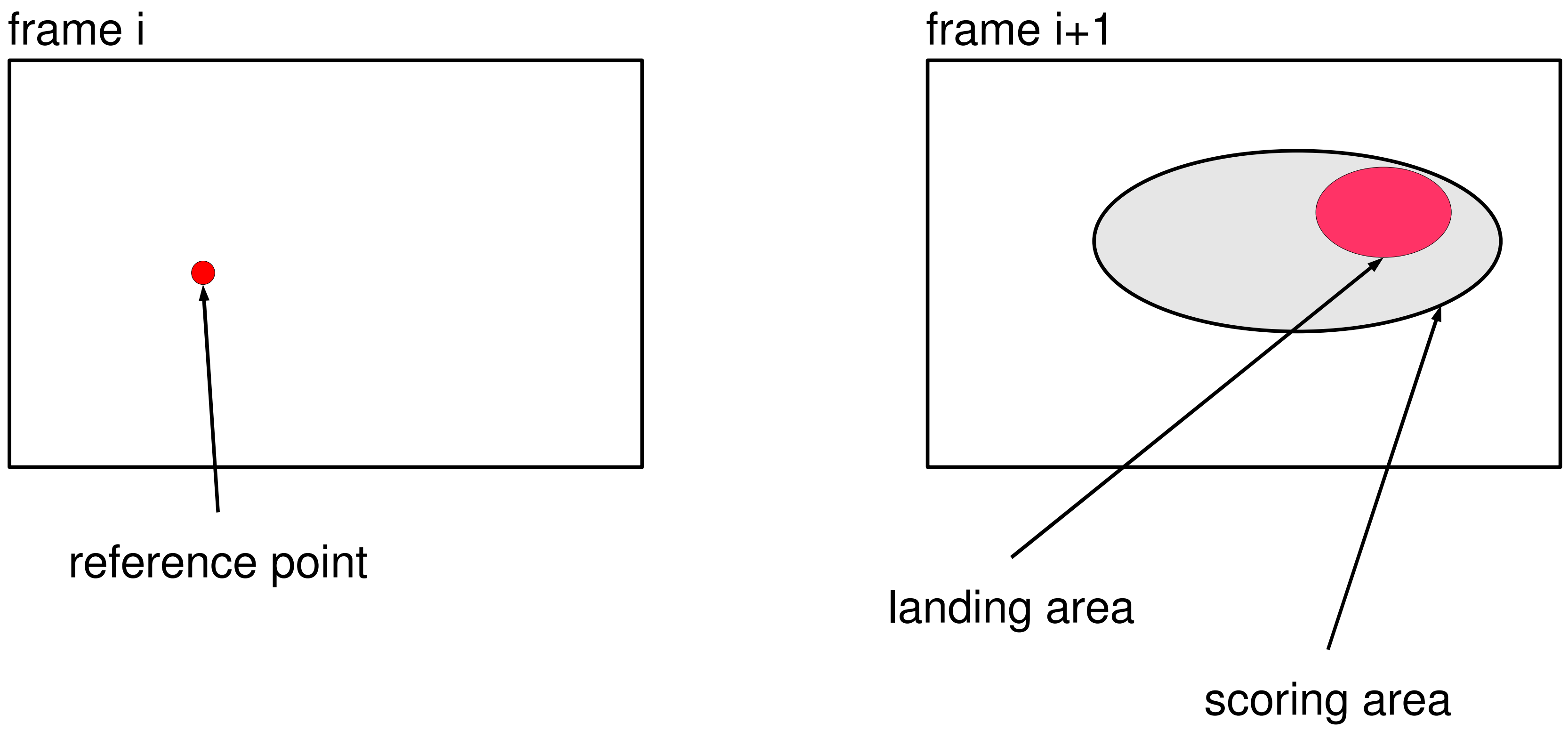
● reference point X

⬬ “landing area” for points emerging from X

Note that FOE and FOC correspond to the points of smallest landing area.

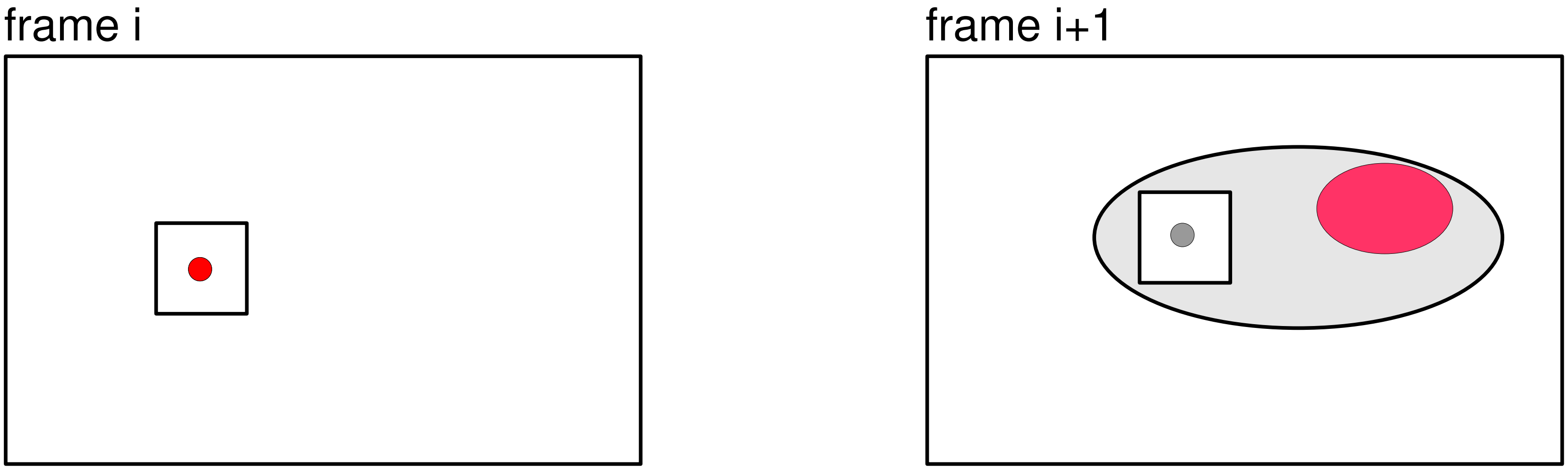
Straight motion decider

Given two successive frames in a video sequence, process each grid point in the following way:



Straight motion decider

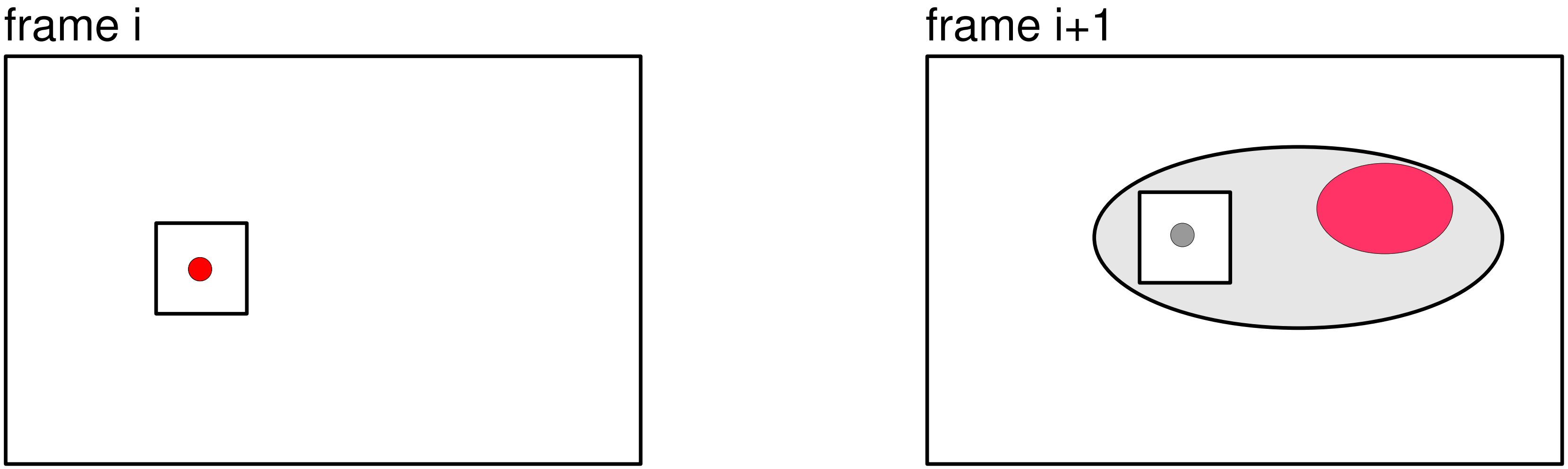
Given two successive frames in a video sequence, process each grid point in the following way:



- Consider a patch P_{ref} of n x n pixels around the reference point (e.g n=5).
- Randomly pick a set of points S_{land} in the landing area; compute the best normalized cross-correlation c_{land} between P_{ref} and any of these points.
- Randomly pick a set of points S_{score} in the scoring area; compute the best normalized cross-correlation c_{score} between P_{ref} and any of these points.

Straight motion decider

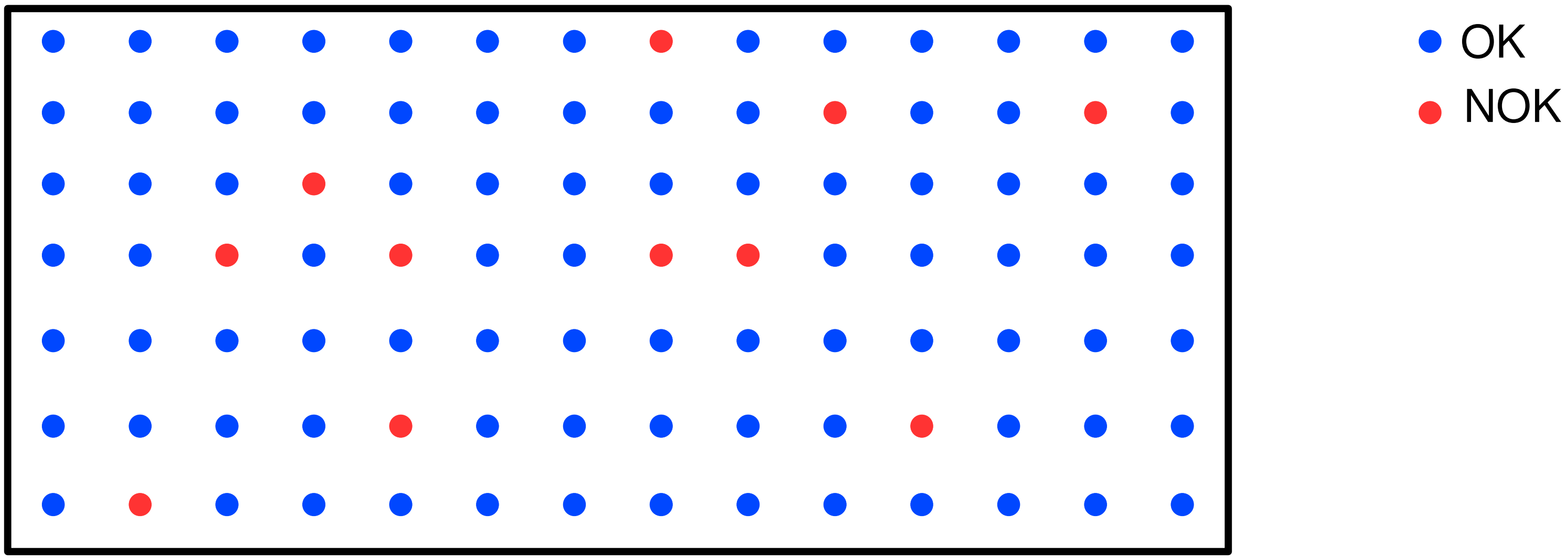
Given two successive frames in a video sequence, process each grid point in the following way:



If $k \times c_{land} > c_{score}$: mark reference point as OK.

$0 < k < 1$ is a parameter to tune (e.g. k = 70%).

Straight motion decider



If the image has at least k% of positive answers, return a positive answer for that camera.

0 < k < 1 is a parameter to tune (e.g. k = 70%)

Straight motion decider

To keep things simple, we process each camera independently and disregard cross-camera correlations.

The yes/no unary may be replace by a scalar between 0 and 1 that incorporates the NCC values (i.e. weight strong matches).

If we assume that a coarse depth of field is given for each calibration sequence (e.g tight corridor = close range, open space = medium range, lobby = long range), the DOP may be incorporated in the landing area and then collected during query, which allows to extract a coarse estimate of the depth of field around the user during straight forward motion.

A note on the Focus Of Expansion

If we assume that the user has only one way of walking forward (e.g no side-stepping), then the location of the FOE can be determined once and for all at calibration time.

Is that an acceptable idea?