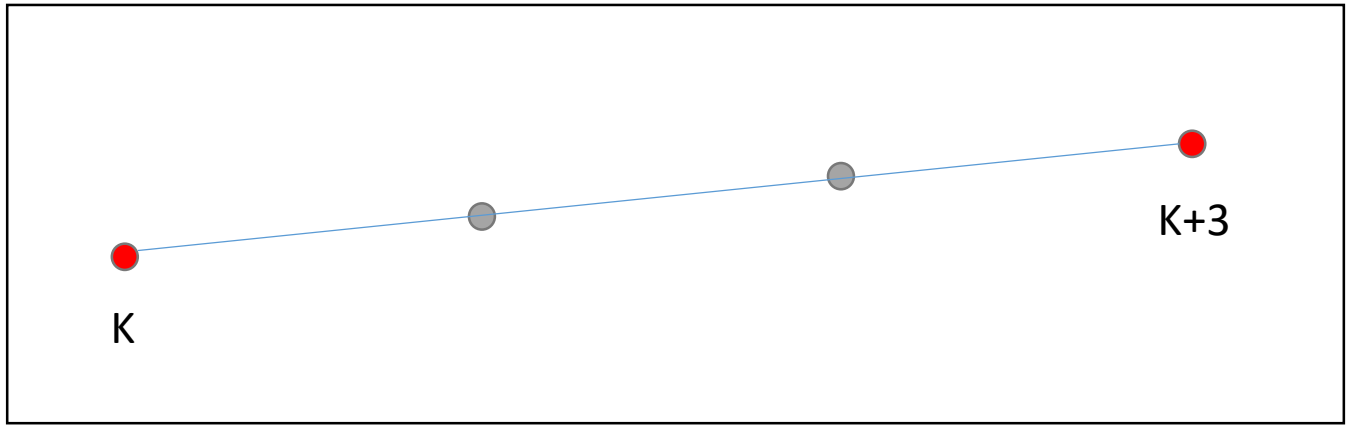
HOW DOES INTERPOLATION OF POLYSHAPES WORKS IN CVAT

Points interpolation

The simple rule is used to points interpolation:

If both neighbor positions at left and at right has one point, linear interpolation is used between these positions.

Else all intermediate frame have the same points like at left.



Polygons and polylines interpolation

There are 3 statements to take into account:

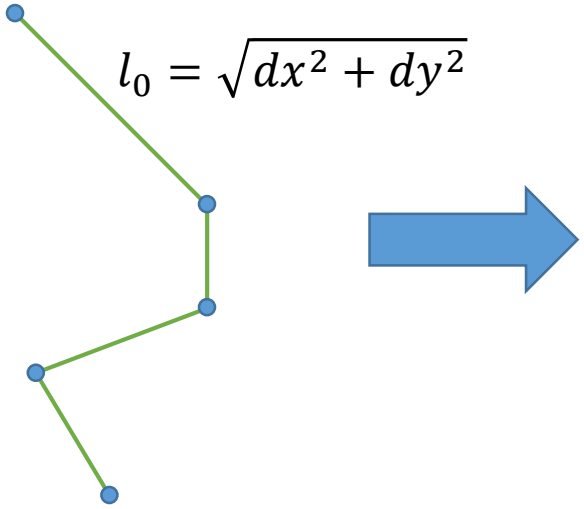
- The both kind of shapes use the same algorithm
- Polygon is considered like a polyline
- To correct working of the algorithm the first point of a polygon is copied to the end of the polygon

The algorithm consists of 5 stages:

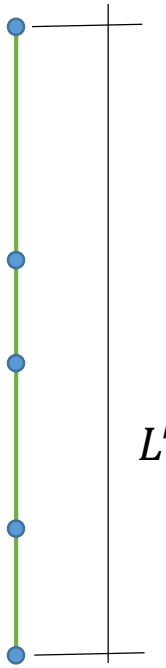
1. Changing representation of a shape
2. The points at left are matched to the points at right
3. Unmatched points at right are matched to the points at left
4. Matched points are interpolated pairwise
5. Number of interpolated points is minimized

Changing representation

From x, y coordinate system to l coordinate system



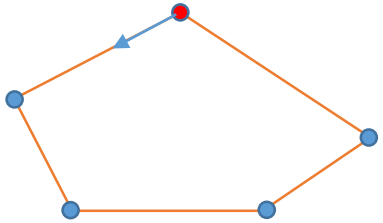
$$l_0 = \sqrt{dx^2 + dy^2}$$



$$l'_0 = \frac{l_0}{L}$$

$$L' = \sum l'_i = 1$$

The same approach

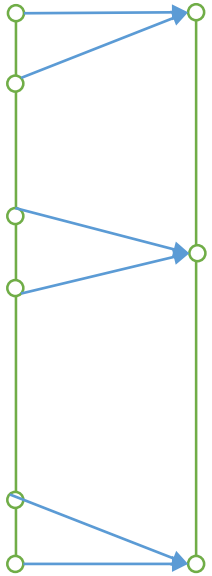
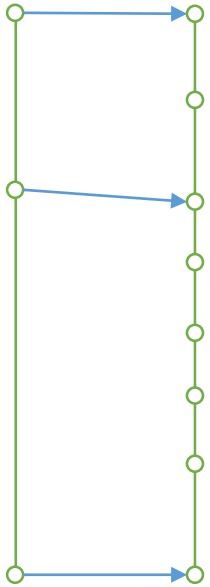
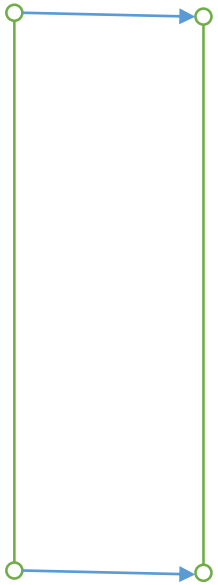
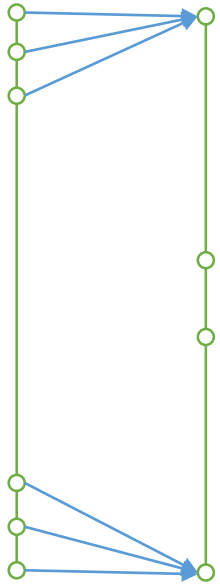


$$l_o = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \; - \; lenght \; of \; the \; first \; segment \; of \; a \; curve$$

$$L \; = \sum l_i \; - \; length \; of \; a \; curve$$

Finding left-right matching

For each left point find the nearest point at right (using absolute distance in l metric).
Examples of possible solutions are shown below:

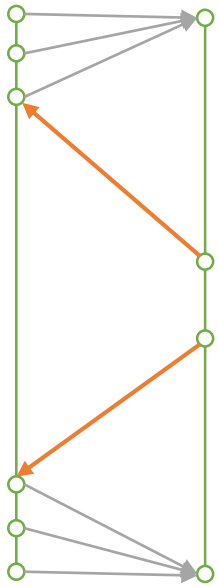





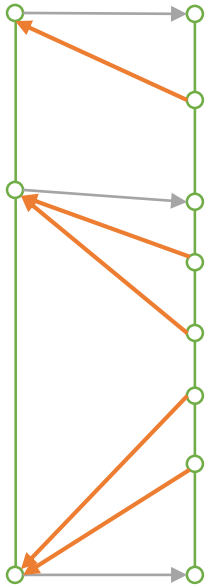
After this stage:

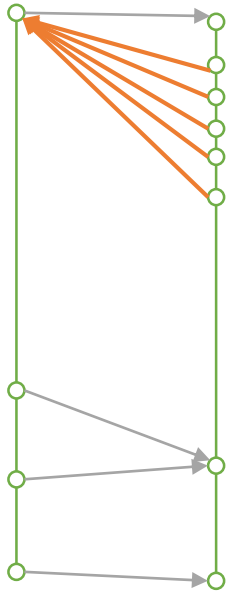
- All points at left are always matched to points at right
- Not all points at right are matched to points at left
- Several points at left may be matched to the same points at right

Finding right-left matching

For each unmapped point at right find the nearest point at left using the same approach like in left-right matching:
Examples of possible solutions are shown below:





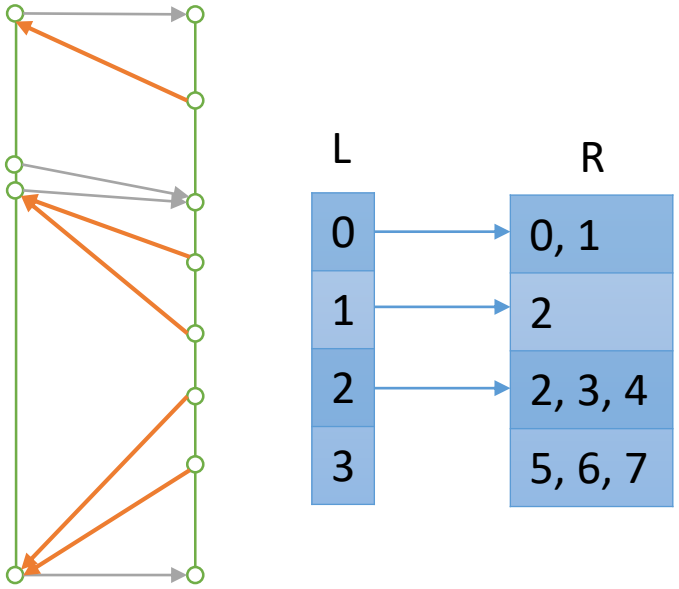


After this stage:

- All points at left are always matched to points at right
- All points at right are always matched to points at left
- Several points at left may be matched to the same points at right
- The same point at left may be matched to several points at right

Pairwise interpolation

Now we have a matching. Something like that:



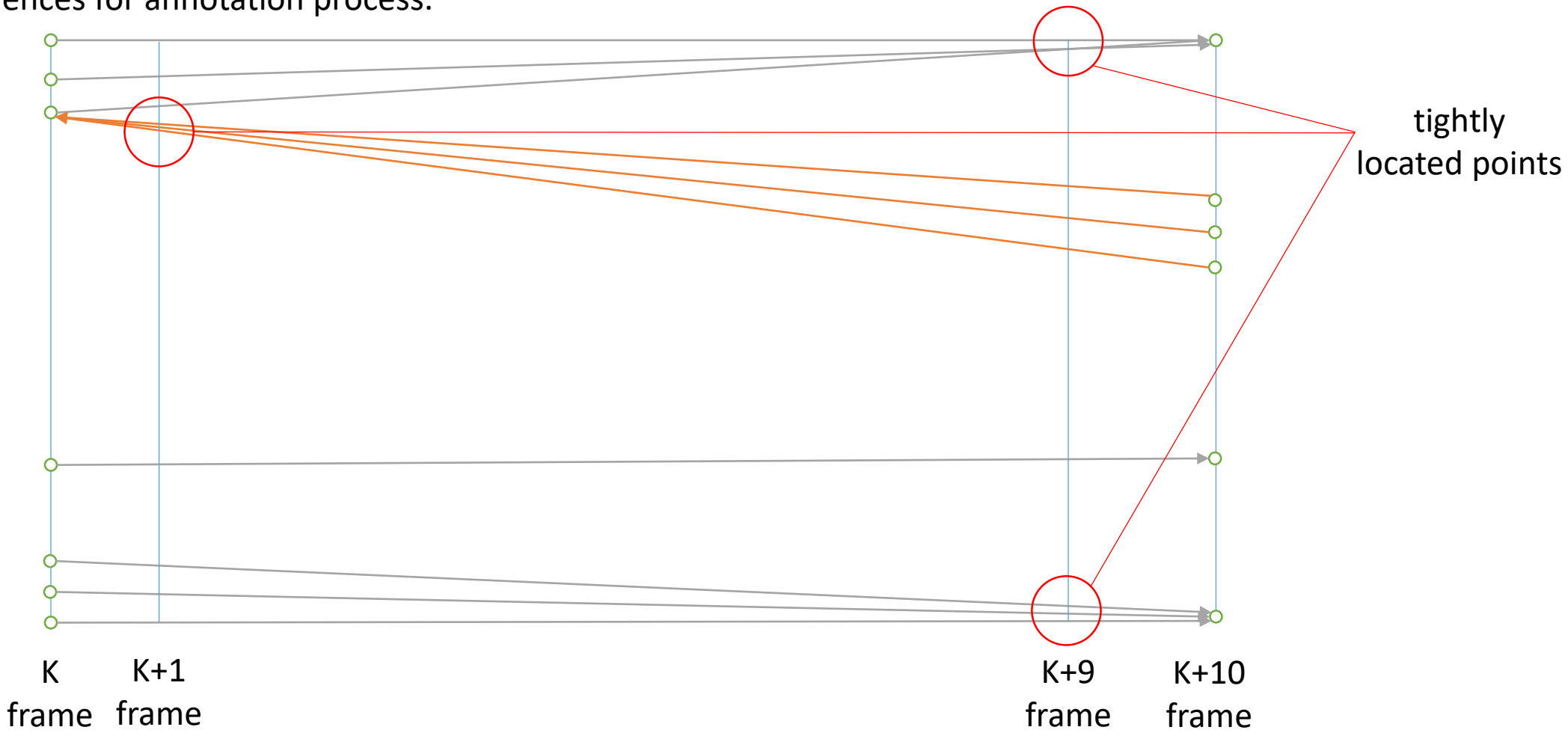
For each pair of points (**0**,**0**; **0**,**1**; **1**,**2**; **2**,**2**; **2**,**3**; **2**,**4**; **3**,**5**; **3**,**6**; **3**.**7**): Create linearly interpolated point (x_i, y_i) using **left point** coordinates, **right point** coordinates and $offset \in [0, 1]$

$$x_i = x_l + (x_r - x_l) * offset$$
$$y_i = y_l + (y_r - y_l) * offset$$

Finally get an array of points with length equal to number of pairs in matching

Minimization the number of points

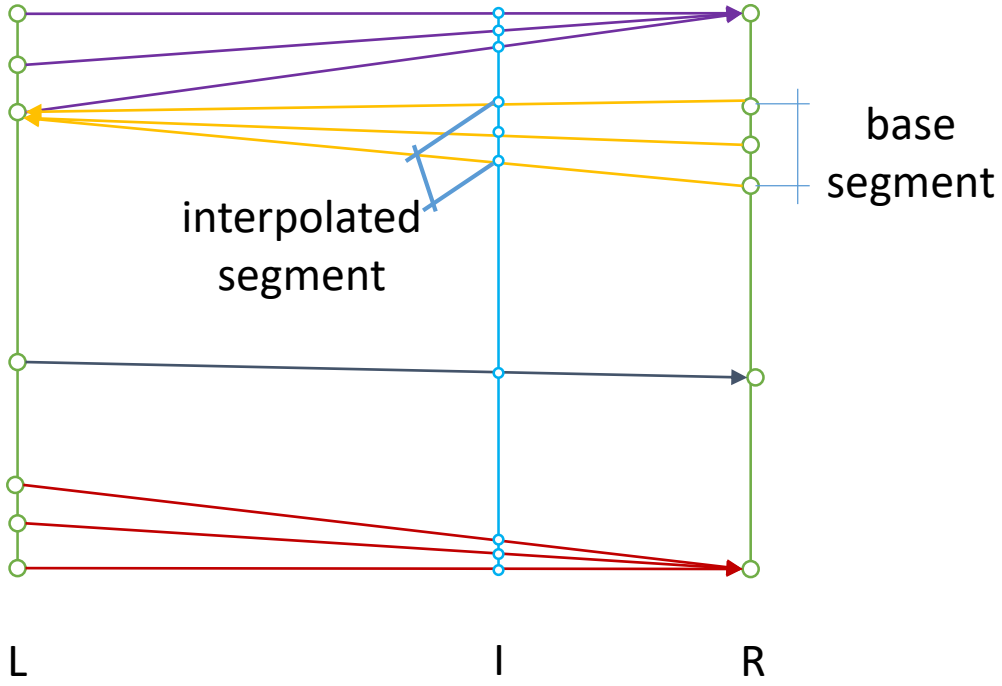
Problem: Near with keyframes interpolated curve can have some areas with tightly located points. Something like in the example below. Need to minimize these points since they create inconveniences for annotation process.



Minimization the number of points

Segment – is a part of matching where one point is matched to one or more points

1. Find all segments (colorized below)



2. For each **one to many** segment compute:

$$threshold = \frac{length\ of\ base\ segment}{2\ *number\ of\ points}$$

3. And go throughout **(first, last)** points on an **interpolated** segment and if length between the next and the previous points less then $threshold$, ignore the next point.

If all intermediate points have been excluded, and interpolated segment consists of two points (first and last), and distance between these points less than threshold, create one average point instead.

4. Concat results of each segment. This is the result of the algorithm.