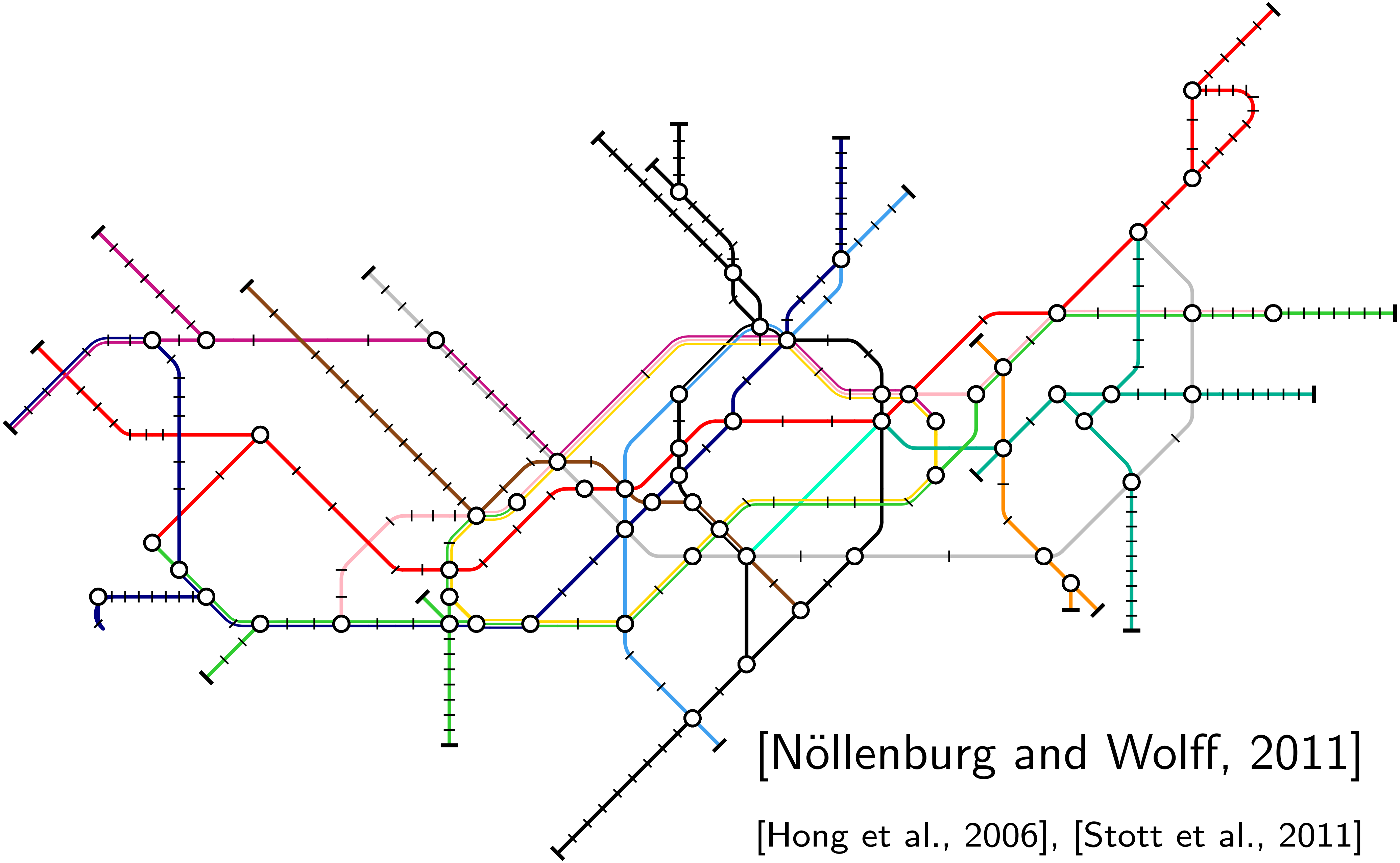
Drawing Metro Maps using Bézier Curves

Martin Fink
Lehrstuhl für Informatik I
Universität Würzburg

Joint work with
Herman Haverkort, Martin Nöllenburg, Maxwell Roberts,
Julian Schuhmann & Alexander Wolff

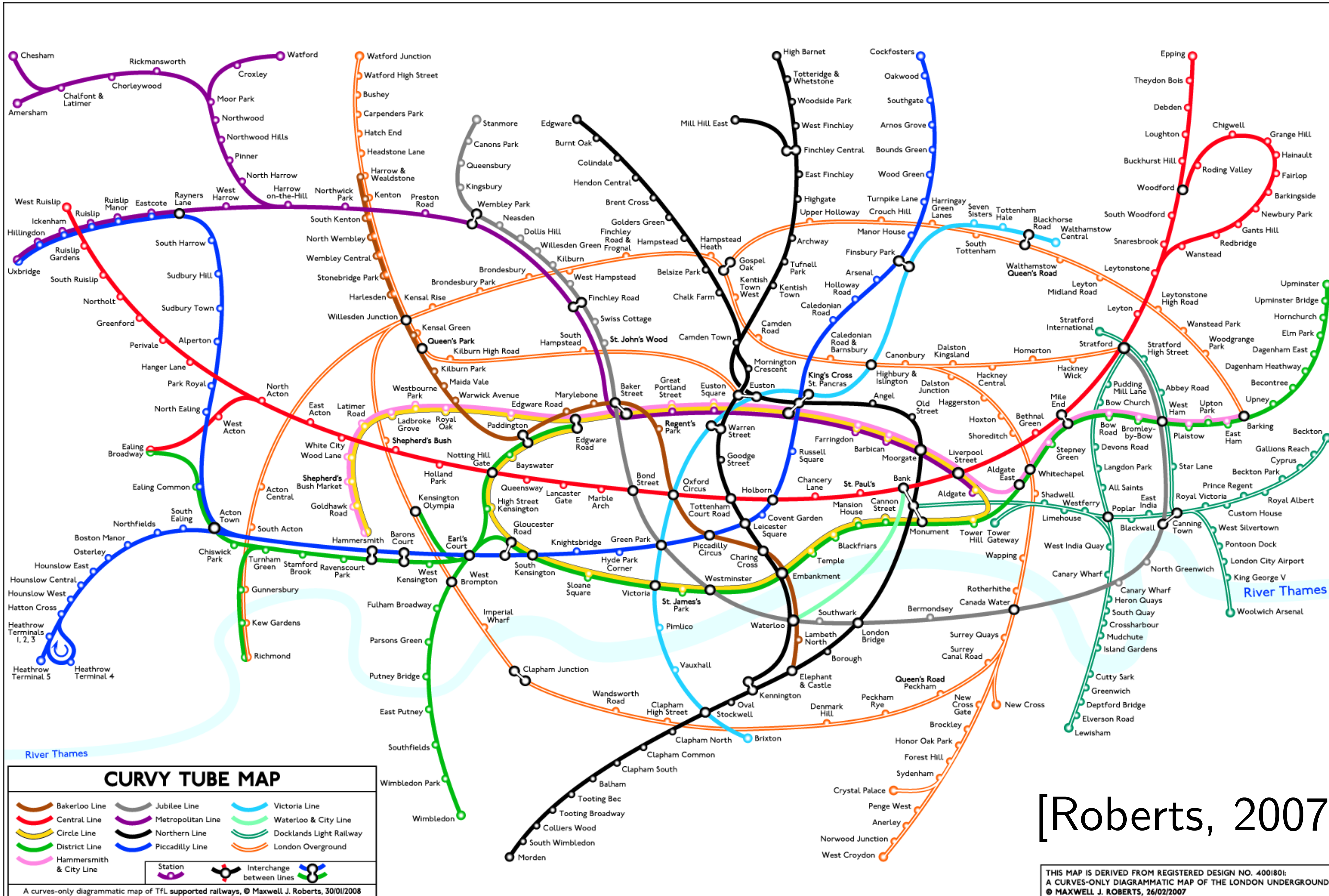
We have ...



[Nöllenburg and Wolff, 2011]

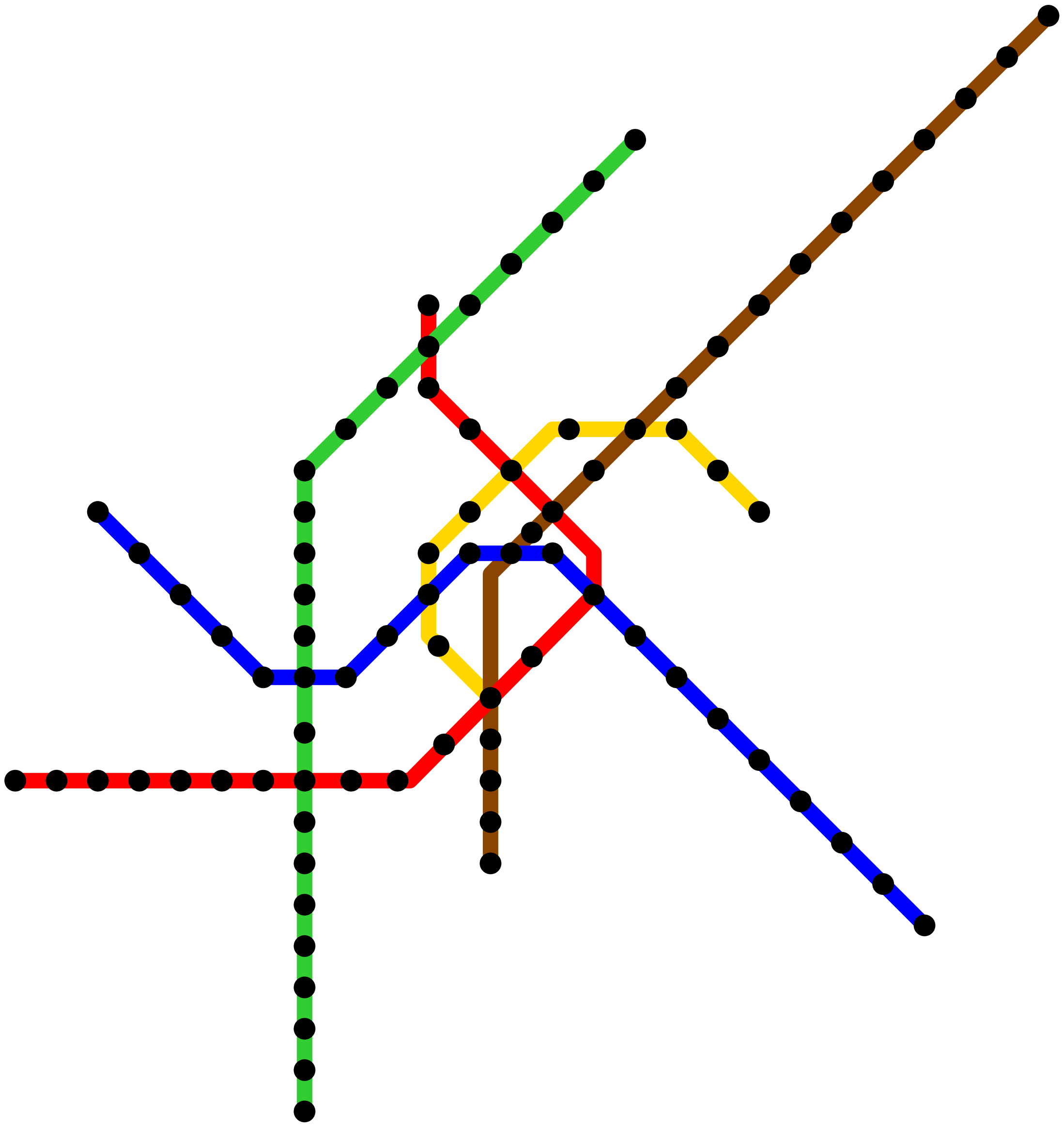
[Hong et al., 2006], [Stott et al., 2011]

We want to create . . .



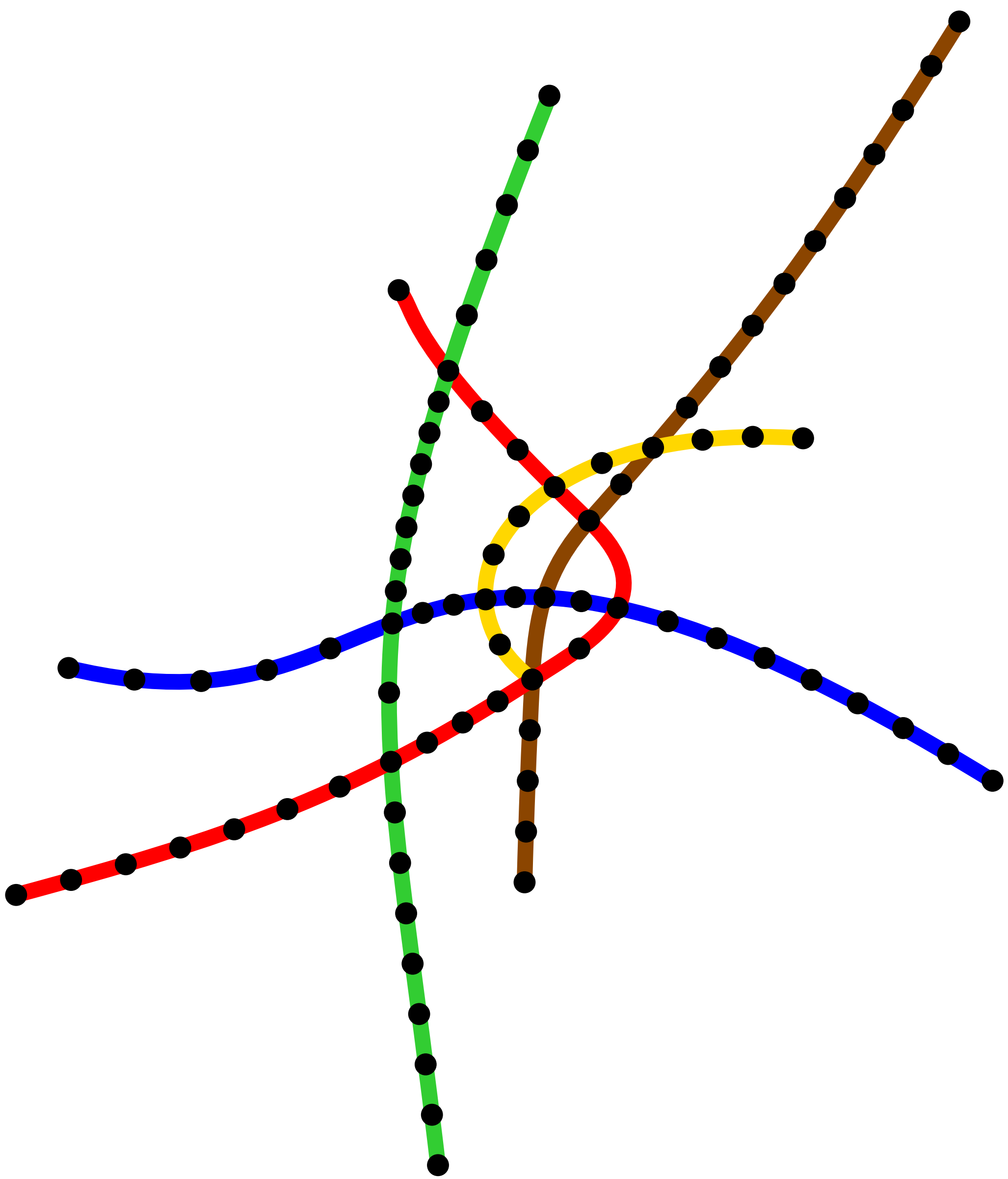
[Roberts, 2007]

Octilinear vs. Curvy Drawings



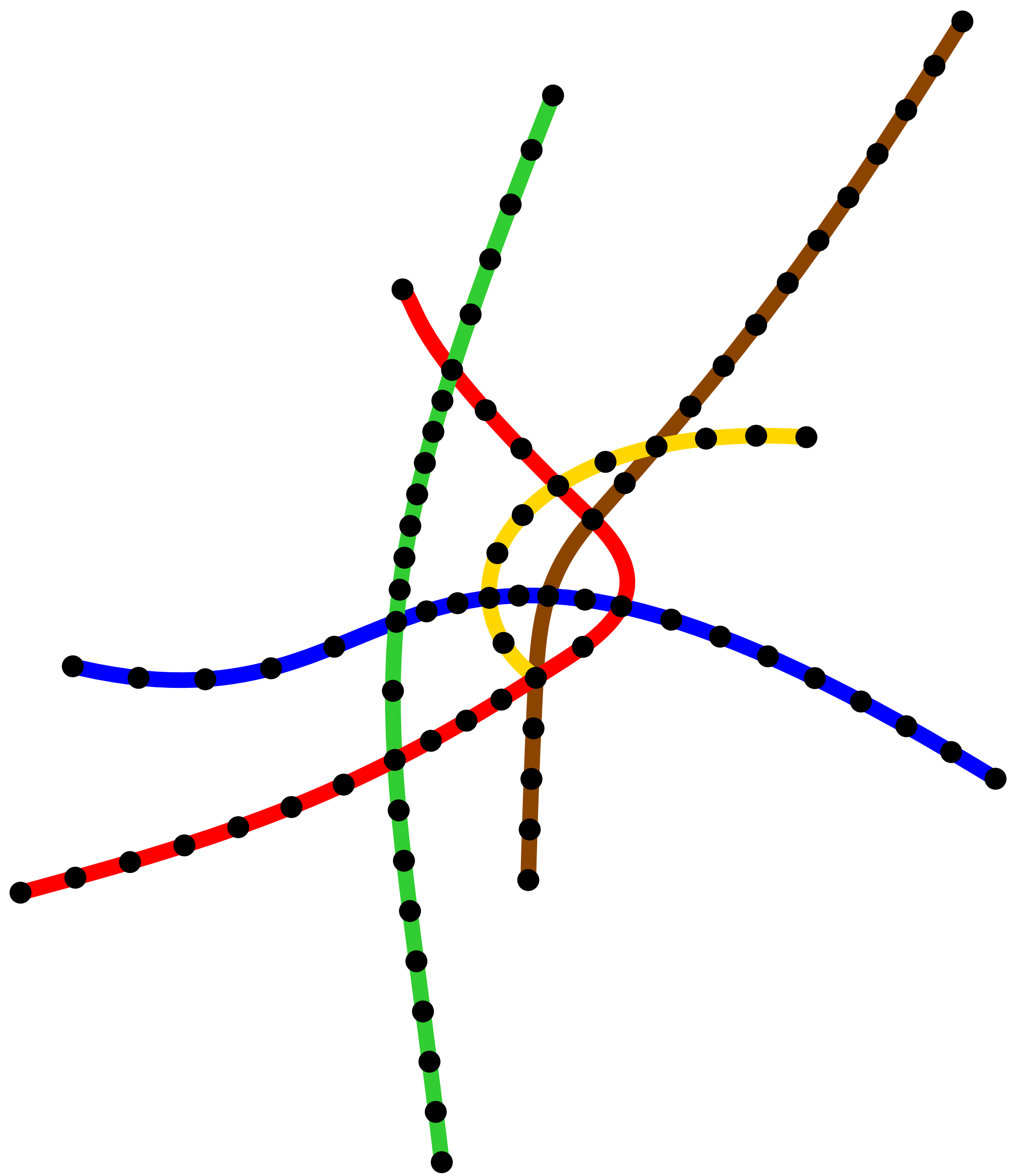
- metro line: polyline with bends (possibly in stations)
- very schematized

Octilinear vs. Curvy Drawings



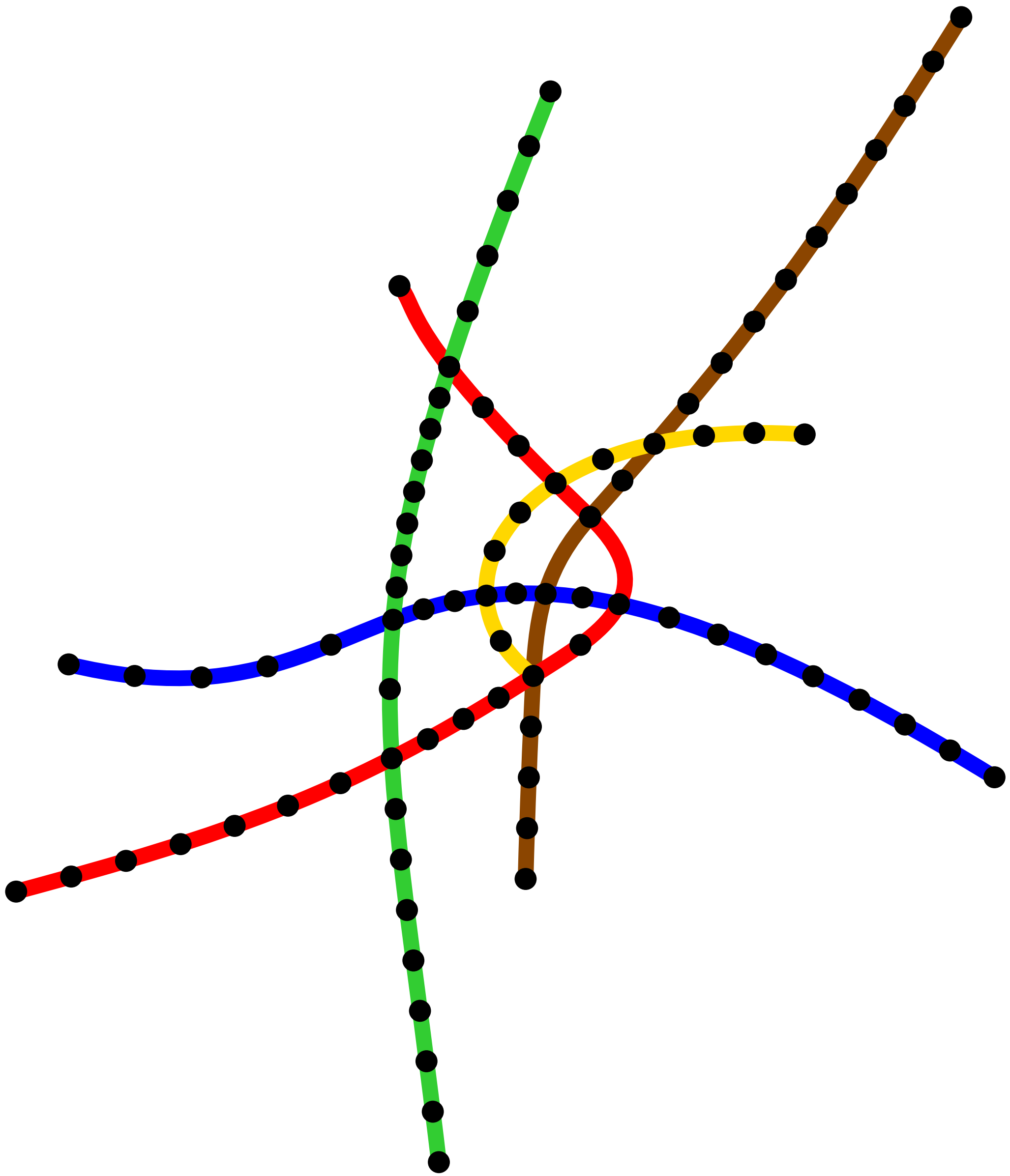
- metro line: polycurve without bends

Octilinear vs. Curvy Drawings



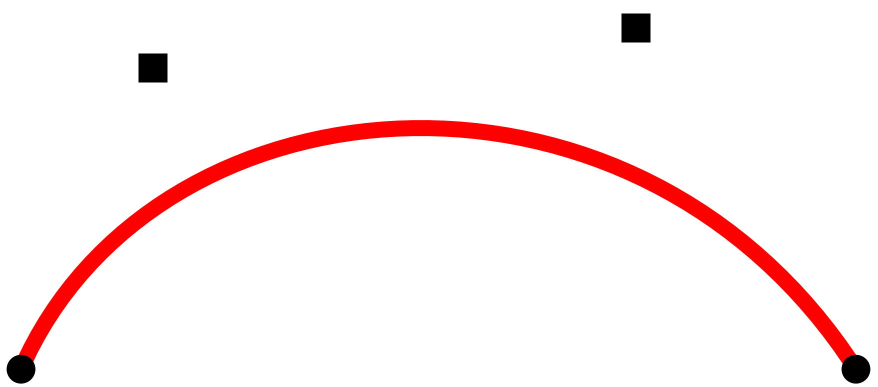
- metro line: polycurve without bends
- [Roberts et al., 2012]: improved planning speed

Octilinear vs. Curvy Drawings



- metro line: polycurve without bends
- [Roberts et al., 2012]: improved planning speed
- more artistic demand of Peter Eades! [GD'10]

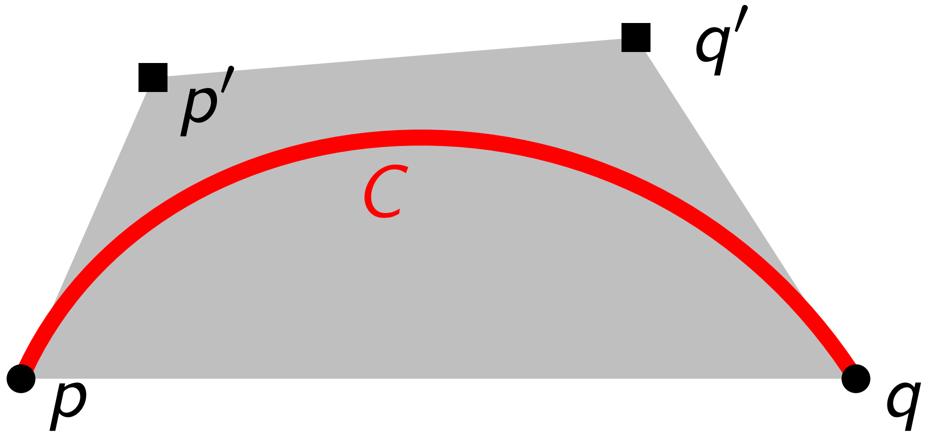
(Cubic) Bézier Curves



- parametric curves

(Cubic) Bézier Curves

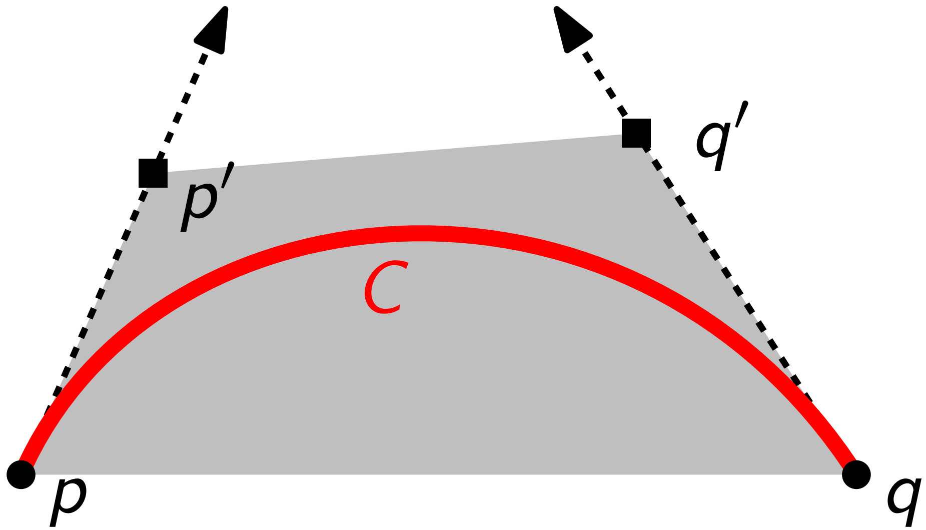
- parametric curves



$$\begin{aligned} C\colon [0,1] &\to \mathbb{R}^2 \\ t &\mapsto (1-t)^3 p + 3(1-t)^2 t p' + 3(1-t) t^2 q' + t^3 q \end{aligned}$$

(Cubic) Bézier Curves

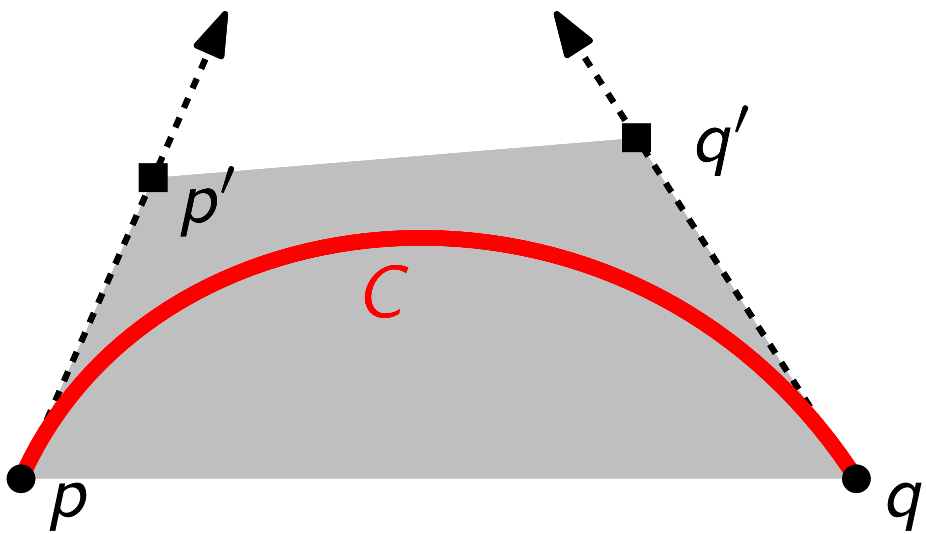
- parametric curves
- leave vertices in direction of tangents

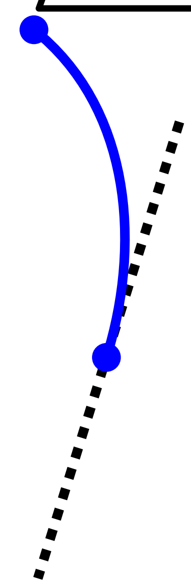


$$
\begin{aligned}
C\colon [0,1] &\to \mathbb{R}^2 \\
t &\mapsto (1-t)^3 p + 3(1-t)^2 t p' + 3(1-t) t^2 q' + t^3 q
\end{aligned}
$$

(Cubic) Bézier Curves

- parametric curves
- leave vertices in direction of tangents



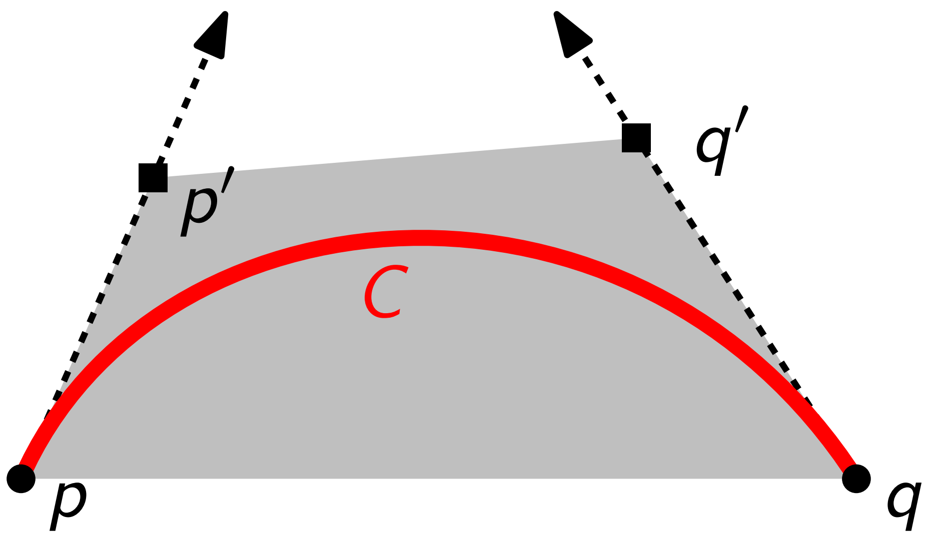


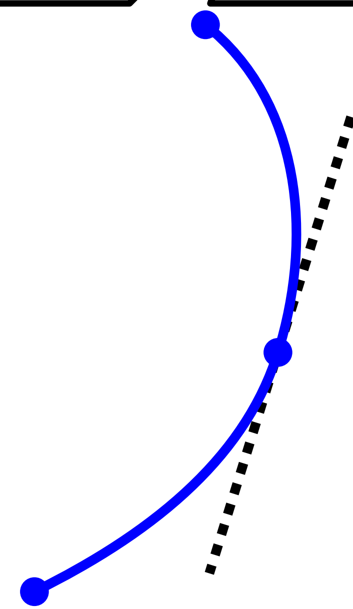
curves share tangents
→ fit smoothly together

$$C\colon [0,1] \to \mathbb{R}^2$$
$$t \mapsto (1-t)^3 p + 3(1-t \ldots q$$

(Cubic) Bézier Curves

- parametric curves
- leave vertices in direction of tangents



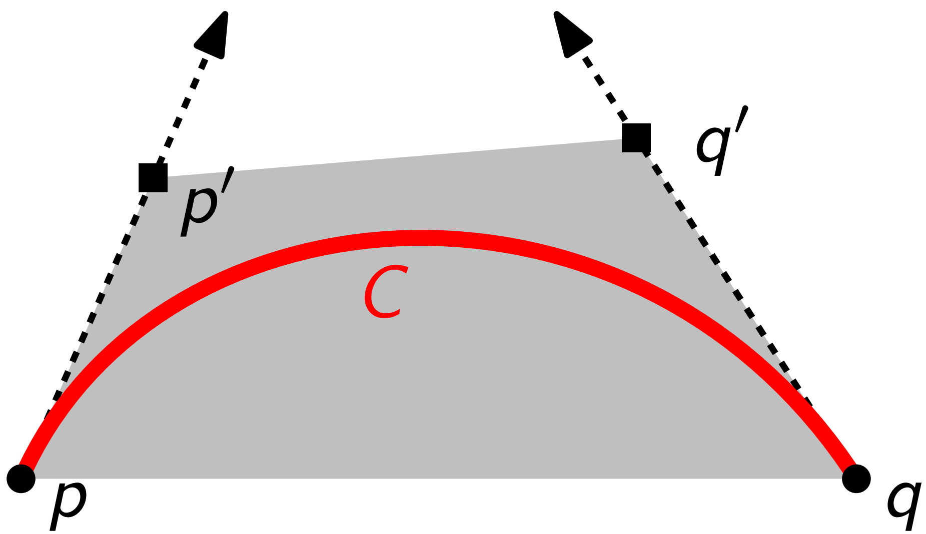


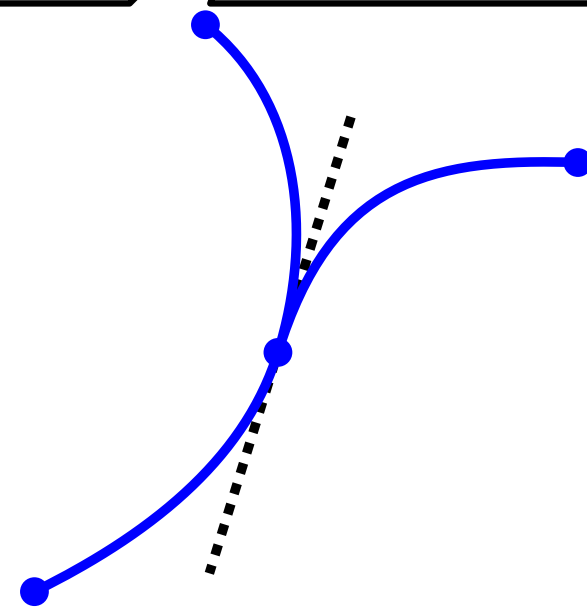
curves share tangents
→ fit smoothly together

$$C\colon [0,1] \to \mathbb{R}^2$$
$$t \mapsto (1-t)^3 p + 3(1- \ldots q$$

(Cubic) Bézier Curves

- parametric curves
- leave vertices in direction of tangents

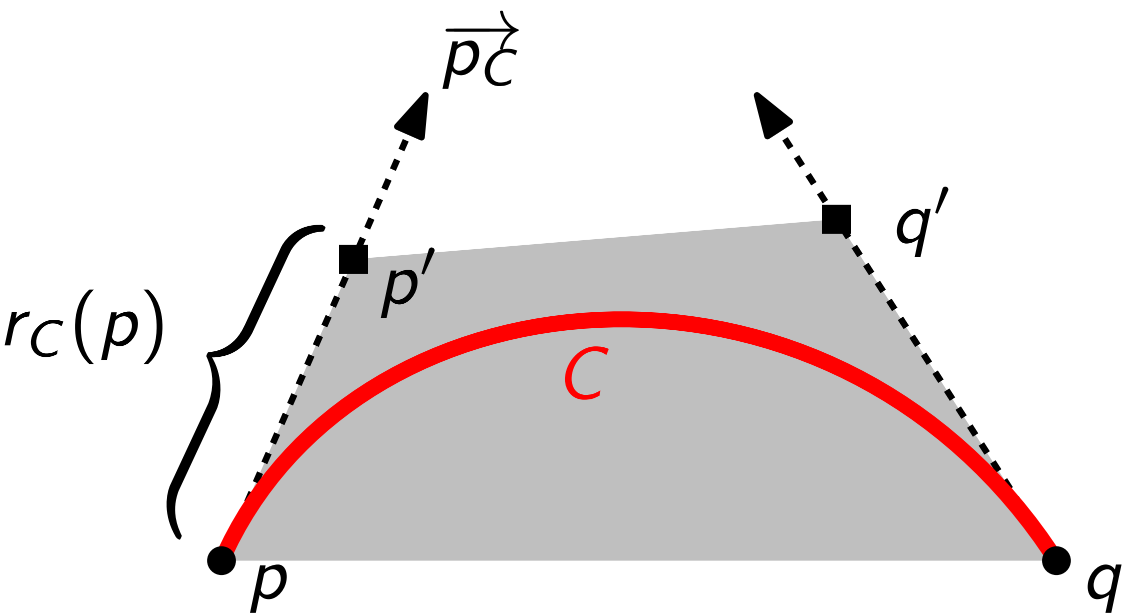




curves share tangents
→ fit smoothly together

$$C\colon [0,1] \to \mathbb{R}^2$$
$$t \mapsto (1-t)^3 p + 3(1-t \ldots q$$

(Cubic) Bézier Curves



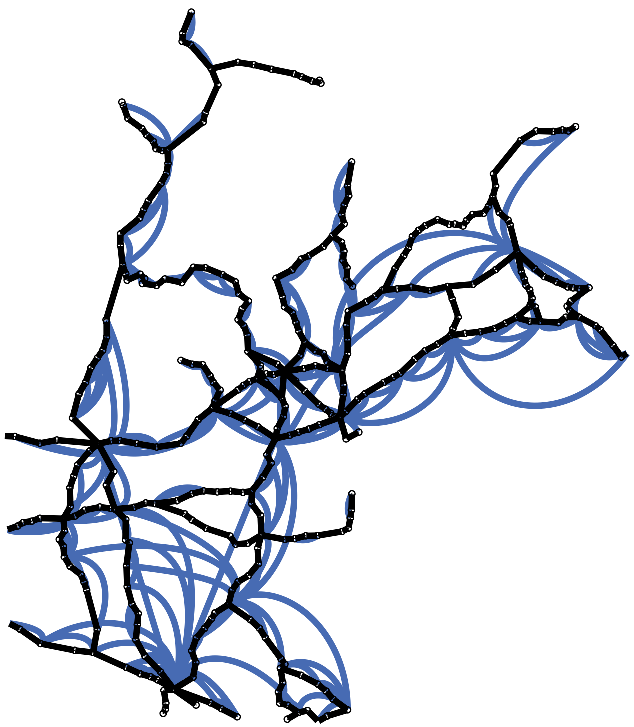
- parametric curves
- leave vertices in direction of tangents
- our representation: control point = tangent + distance

$$\begin{aligned} C\colon [0,1] &\to \mathbb{R}^2 \\ t &\mapsto (1-t)^3 p + 3(1-t)^2 t p' + 3(1-t) t^2 q' + t^3 q \end{aligned}$$

Previous Work

- [Brandes, Shubina, Tamassia, Wagner, 2001]: Bézier curves used for visualizing train connections.

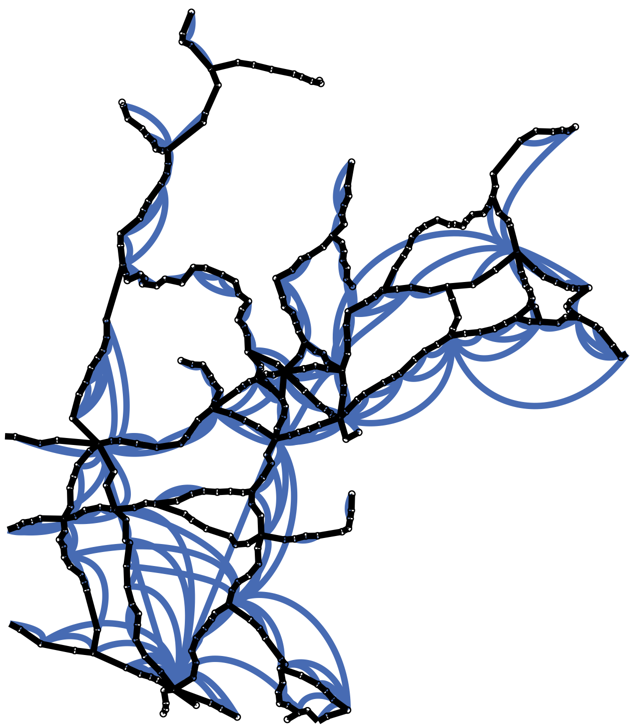
  [Brandes, Shubina, Tamassia, 2000], [Brandes, Wagner, 1998]



Previous Work

- [Brandes, Shubina, Tamassia, Wagner, 2001]: Bézier curves used for visualizing train connections.

  [Brandes, Shubina, Tamassia, 2000], [Brandes, Wagner, 1998]



Our approach:
– share tangents
– move vertices

Previous Work

- [Brandes, Shubina, Tamassia, Wagner, 2001]: Bézier curves used for visualizing train connections.

  [Brandes, Shubina, Tamassia, 2000], [Brandes, Wagner, 1998]

  Our approach:
  – share tangents
  – move vertices

- [Finkel and Tamassia, 2005]: Force-directed algorithm for drawing graphs with Bézier Curves

Previous Work

- [Brandes, Shubina, Tamassia, Wagner, 2001]: Bézier curves used for visualizing train connections.

  [Brandes, Shubina, Tamassia, 2000], [Brandes, Wagner, 1998]

  Our approach:
  – share tangents
  – move vertices

- [Finkel and Tamassia, 2005]: Force-directed algorithm for drawing graphs with Bézier Curves

  Method: Control-points as extra vertices.

Previous Work

- [Brandes, Shubina, Tamassia, Wagner, 2001]: Bézier curves used for visualizing train connections.

  [Brandes, Shubina, Tamassia, 2000], [Brandes, Wagner, 1998]

  Our approach:
  – share tangents
  – move vertices

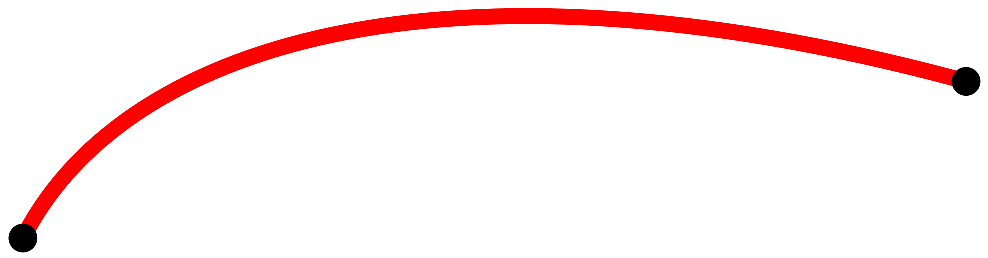
- [Finkel and Tamassia, 2005]: Force-directed algorithm for drawing graphs with Bézier Curves

  Method: Control-points as extra vertices.

  our approach: control points are no vertices

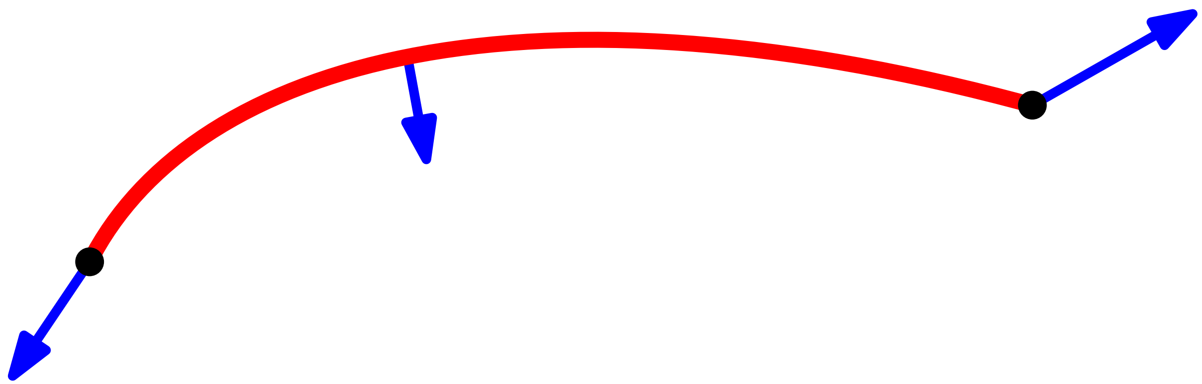
Our Approach

- use Bézier curves for representing edges



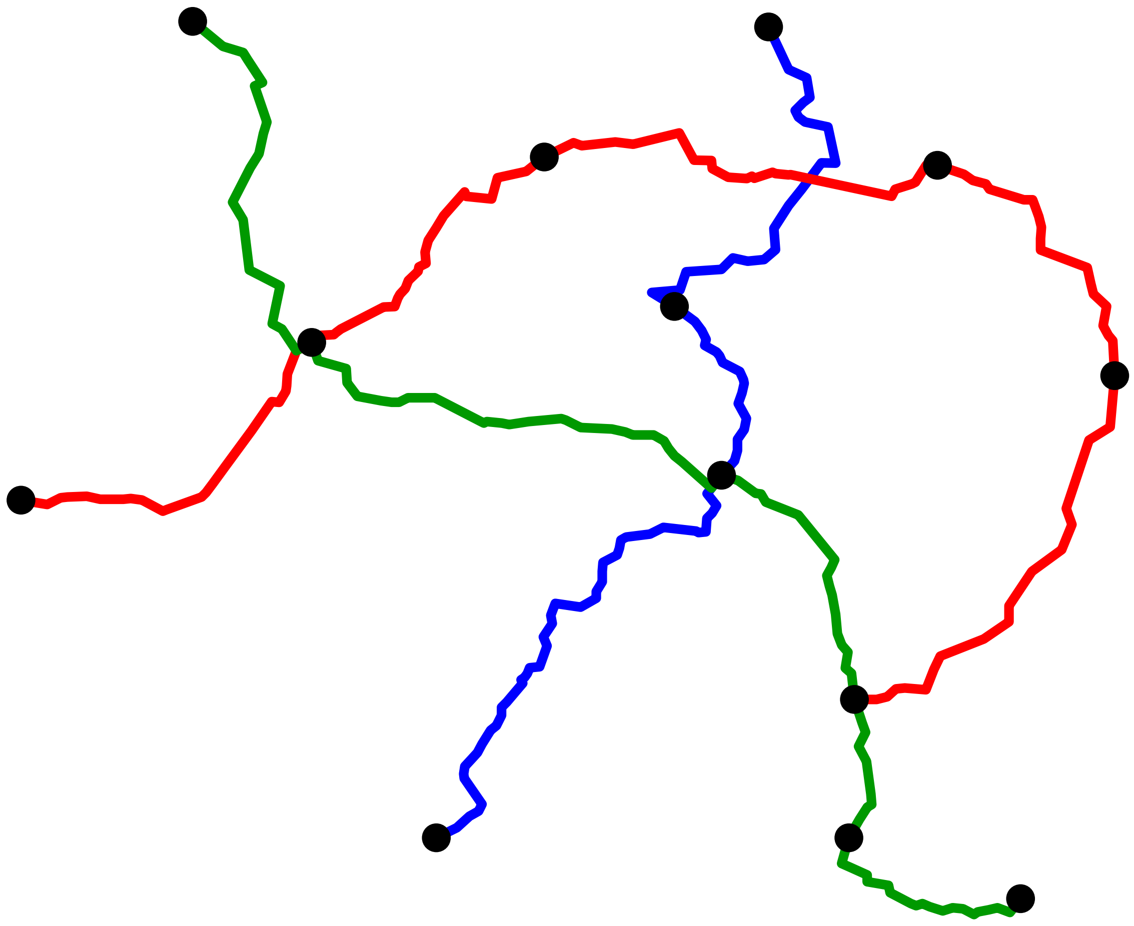
Our Approach

- use Bézier curves for representing edges
- use force-directed approach



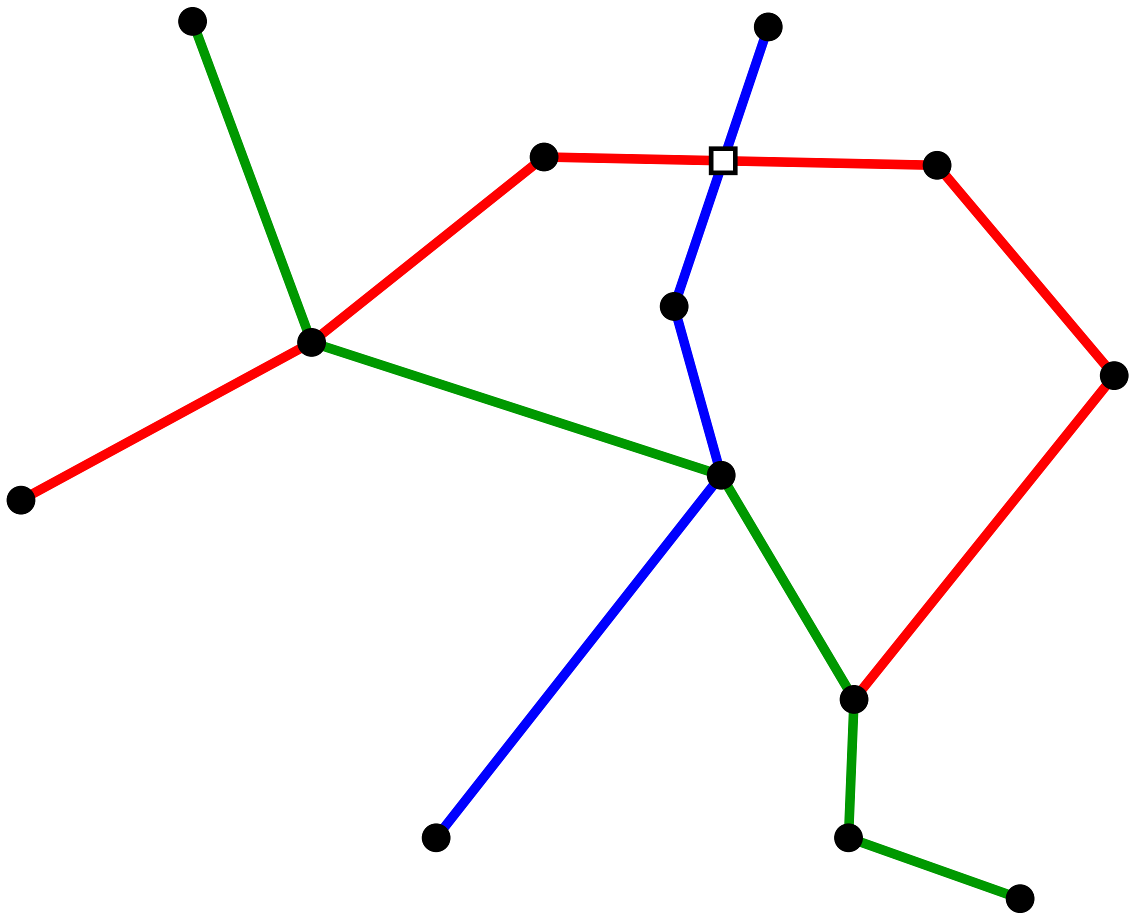
Obtaining an Initial Drawing with Curves

- geographic input



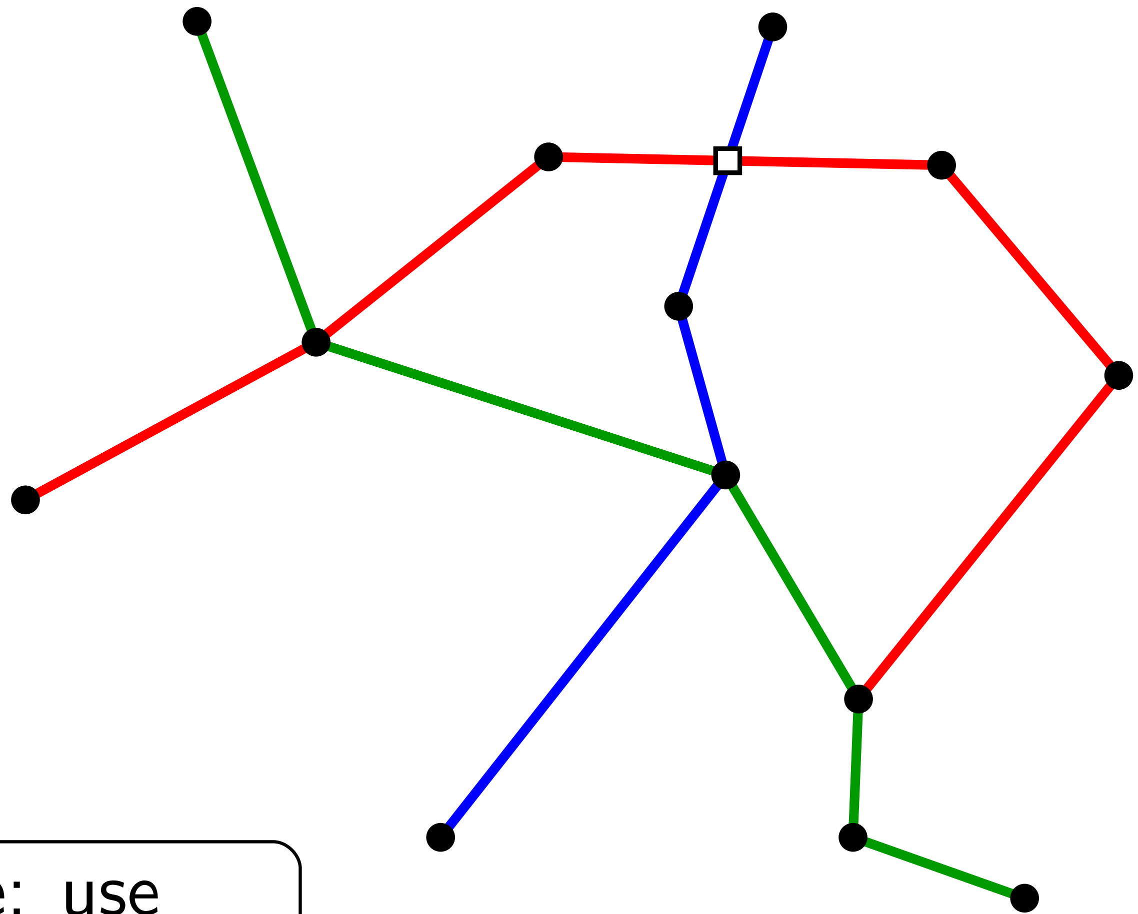
Obtaining an Initial Drawing with Curves

- straight-line drawing



Obtaining an Initial Drawing with Curves

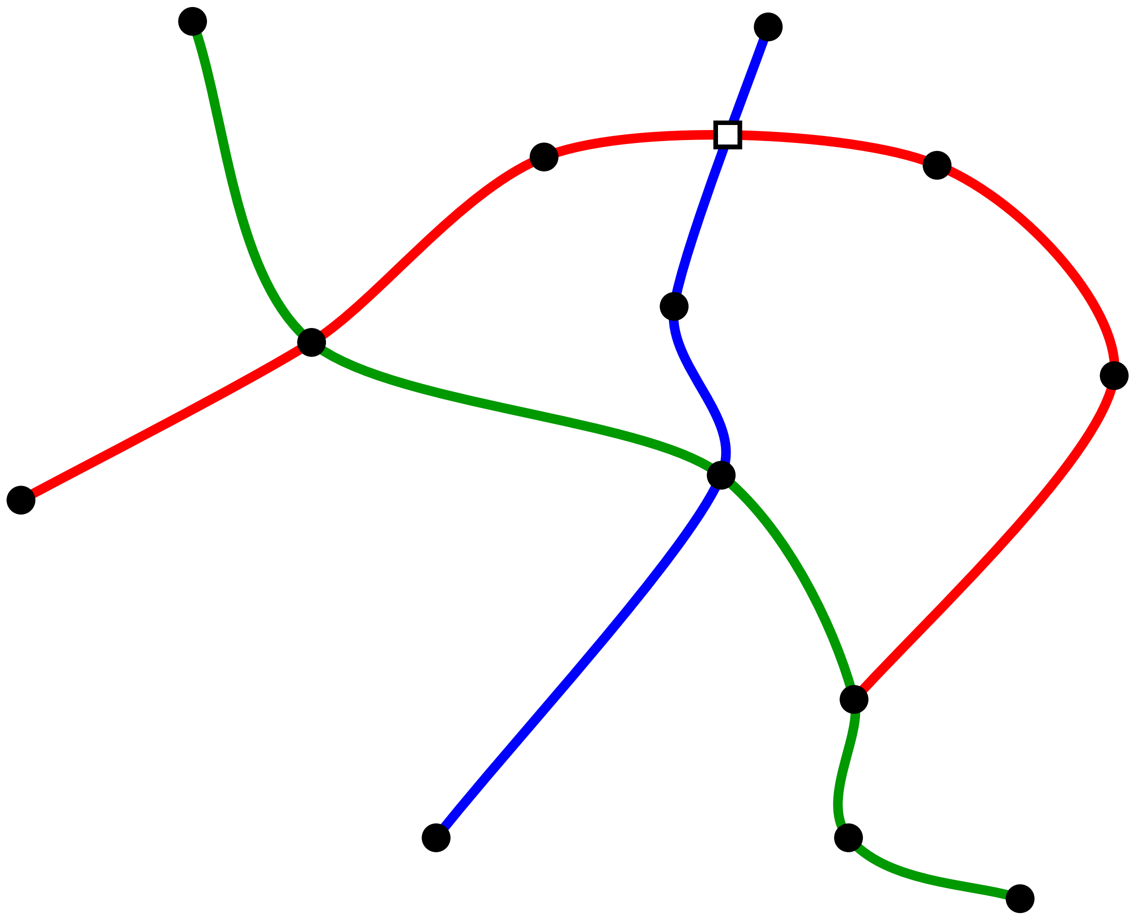
- straight-line drawing



alternative: use octilinear drawing

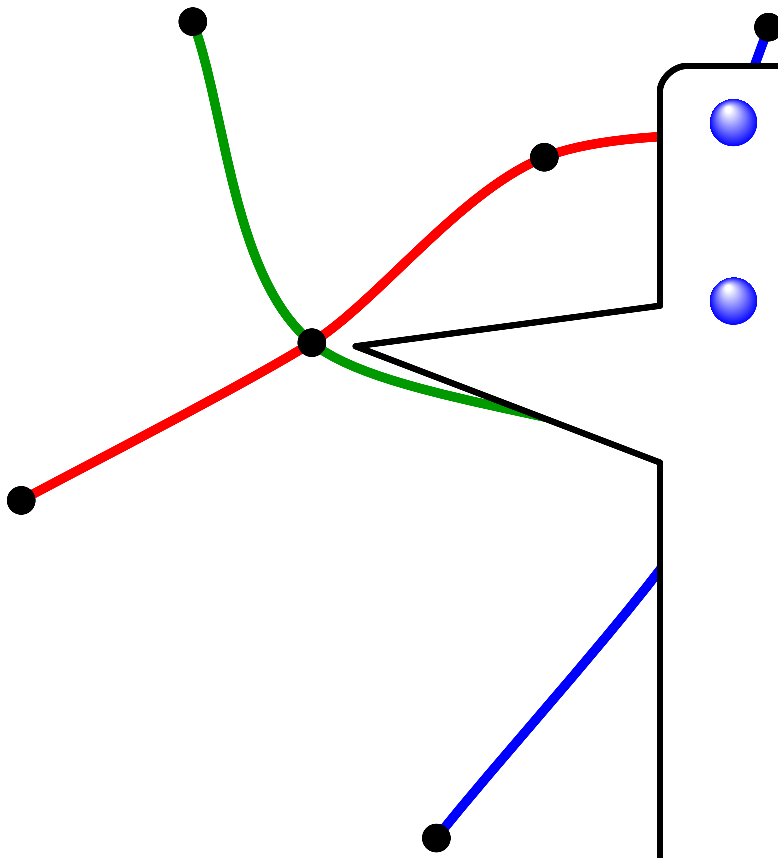
Obtaining an Initial Drawing with Curves

- approximation by Bézier Curves

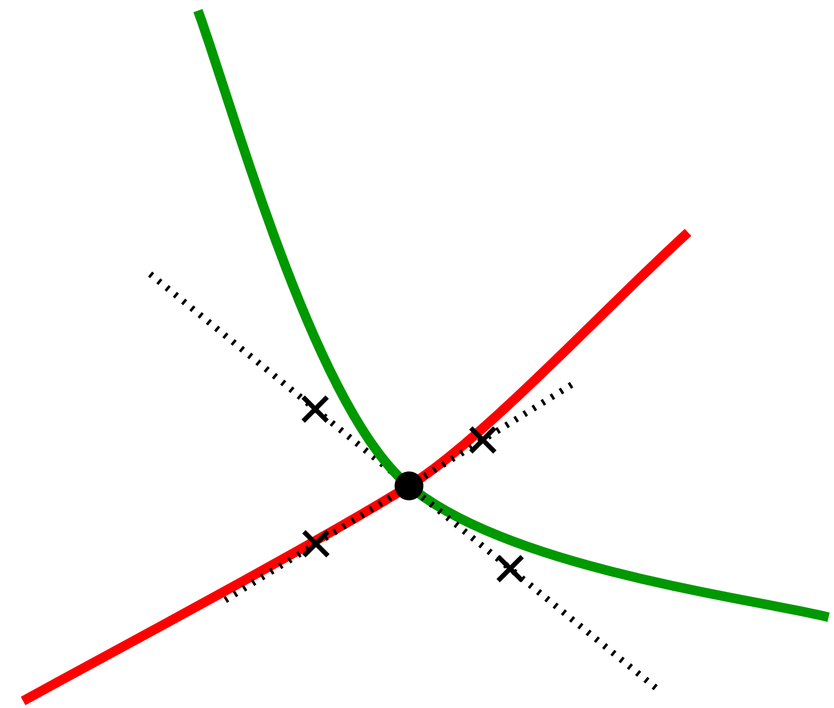


Obtaining an Initial Drawing with Curves

- approximation by Bézier Curves



- put control points close to vertex
- use common tangents



Structure of the Algorithm

while the drawing changes

compute forces on vertices

compute forces on curves

apply forces

simplify the drawing

perform post-processing simplification

Structure of the Algorithm

while the drawing changes

compute forces on vertices

stop if displacement is small

compute forces on curves

apply forces

simplify the drawing

perform post-processing simplification

Structure of the Algorithm

while the drawing changes

    compute forces on vertices

    compute forces on curves

    apply forces

    simplify the drawing

perform post-processing simplification

Structure of the Algorithm

while the drawing changes

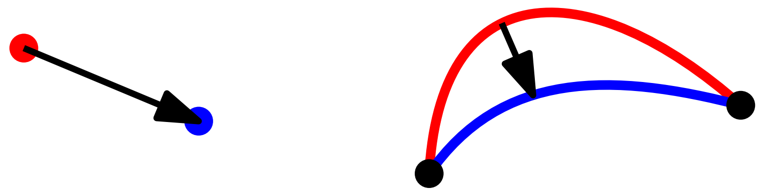
compute forces on vertices

compute forces on curves

apply forces

simplify the drawing

sum up to desired changes



perform post-processing simplification

Structure of the Algorithm

while the drawing changes

compute forces on vertices

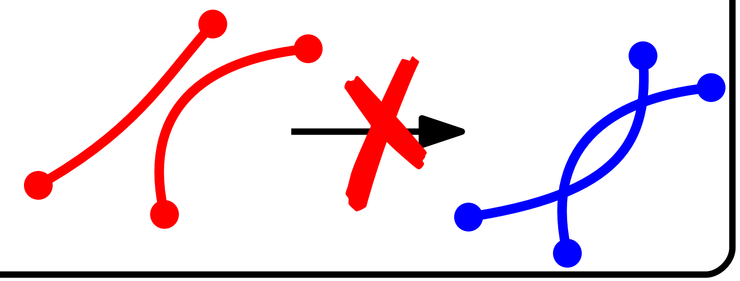
compute forces on curves

apply forces

simplify the drawing

perform post-processing sim

avoid intersections



Structure of the Algorithm

while the drawing changes

compute forces on vertices

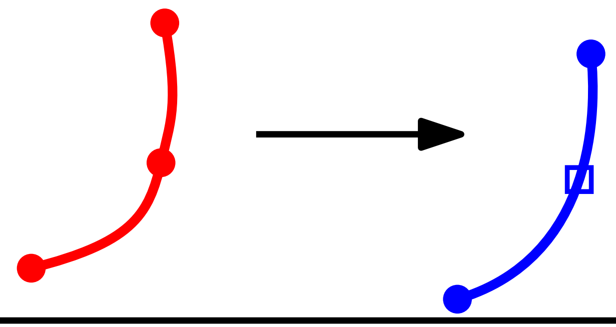
compute forces on curves

apply forces

simplify the drawing

perform post-processing simplification

merge curves around deg-2 vertex



Structure of the Algorithm

while the drawing changes

compute forces on vertices

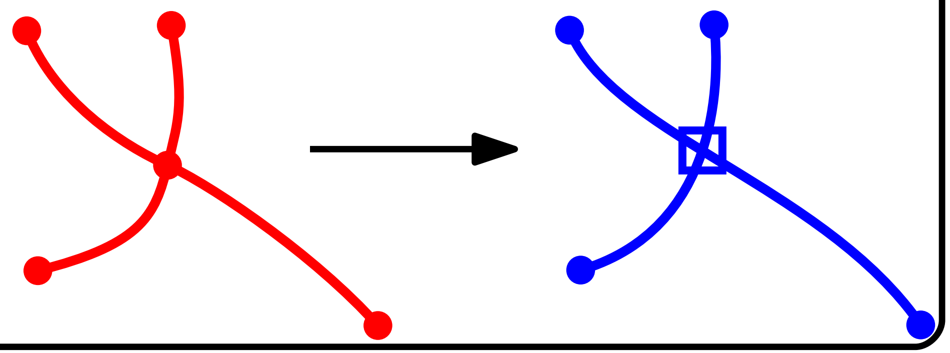
compute forces on curves

apply forces

simplify the drawing

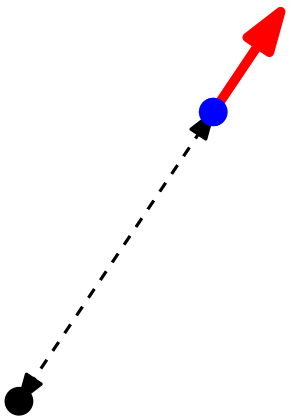
perform post-processing simplification

merge 4 curves to 2



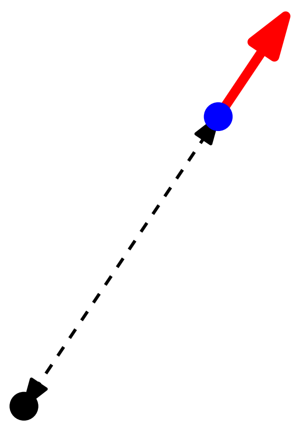
Forces – Vertices

– repulsion

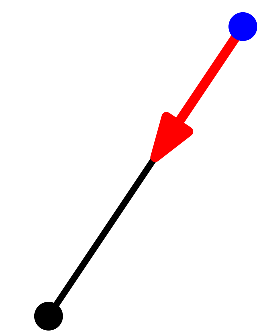


Forces – Vertices

– repulsion

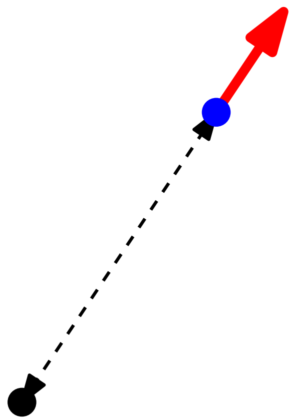


– attraction of adjacent vertices

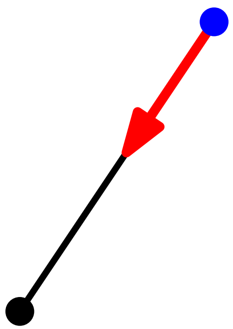


Forces – Vertices

– repulsion



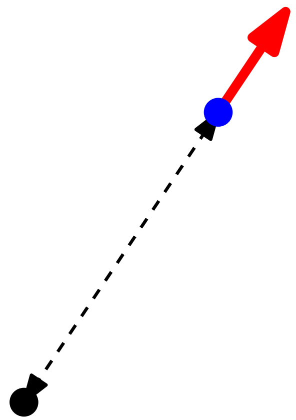
– attraction of adjacent vertices



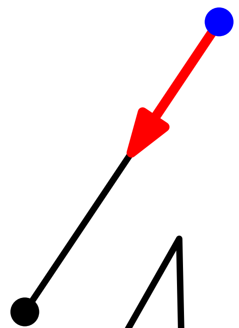
reuse forces from Fruchterman-Rheingold

Forces – Vertices

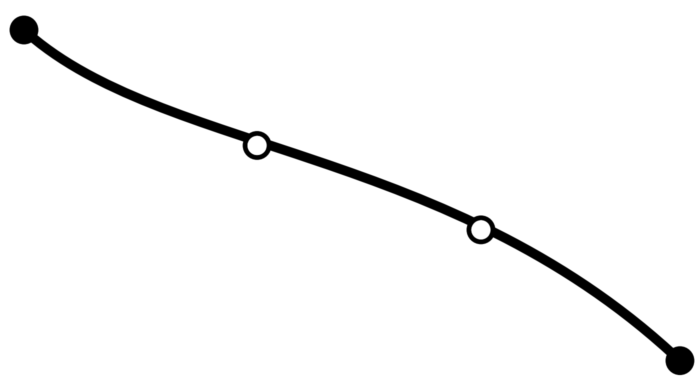
- repulsion



- attraction of adjacent vertices

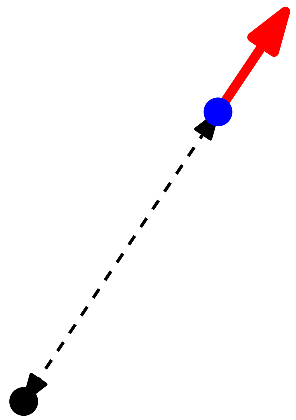


desired edge length =
const $\cdot$ (# intermediate stops + 1)

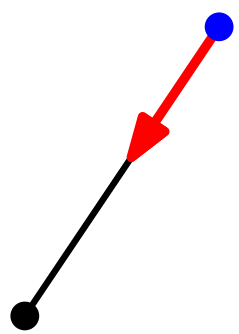


Forces – Vertices

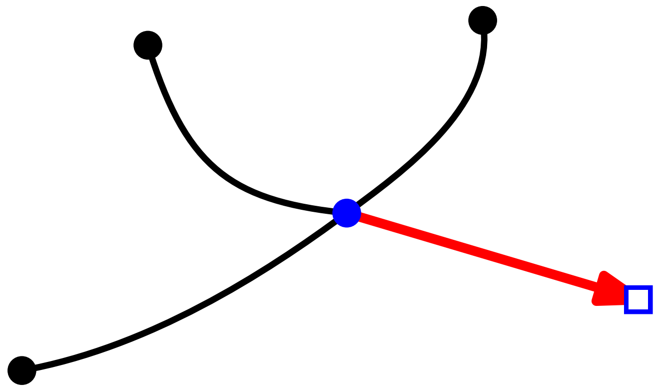
– repulsion



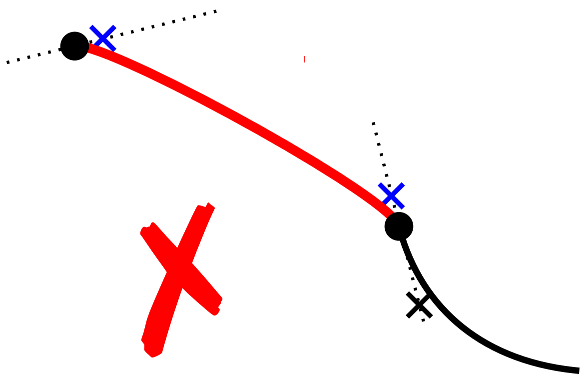
– attraction of adjacent vertices



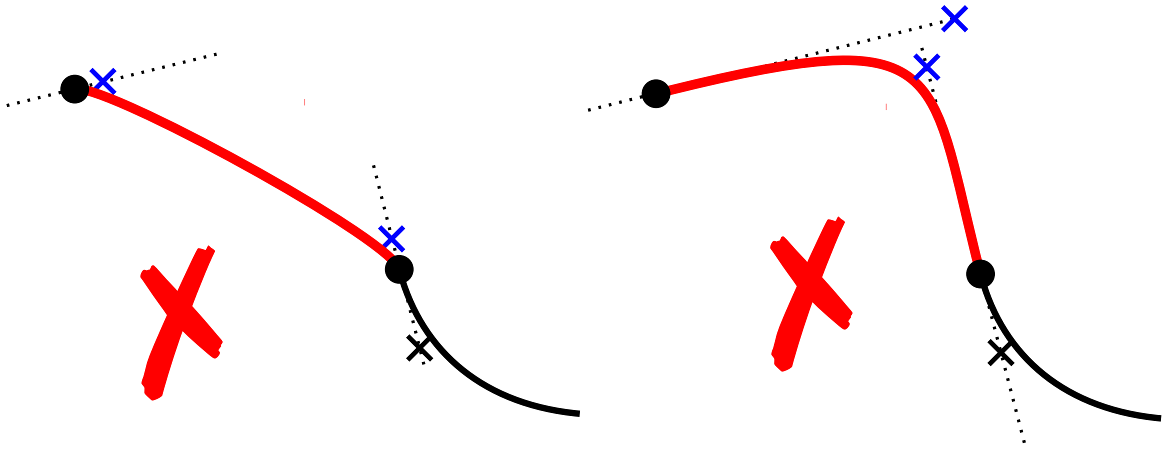
– attraction to geographic position



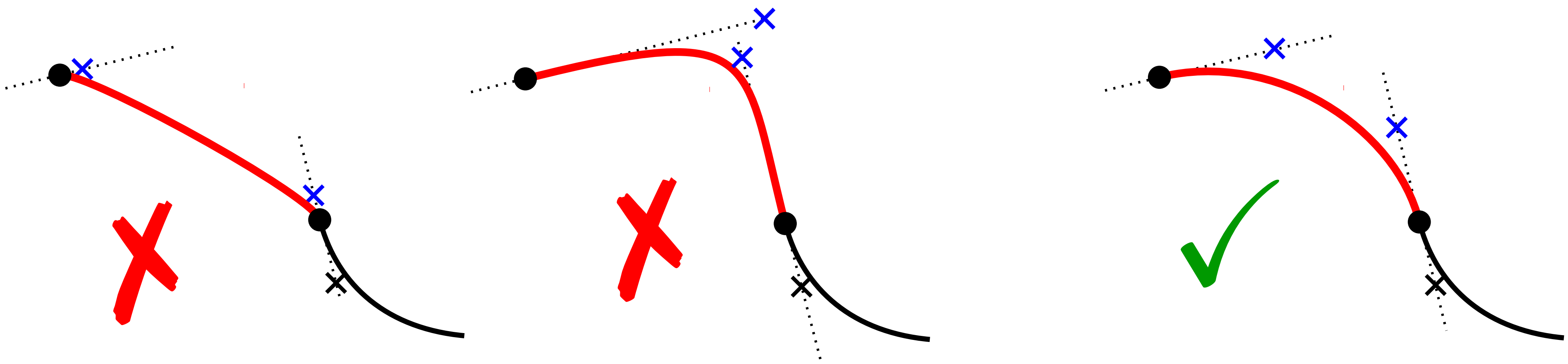
Forces – Shape of Curves



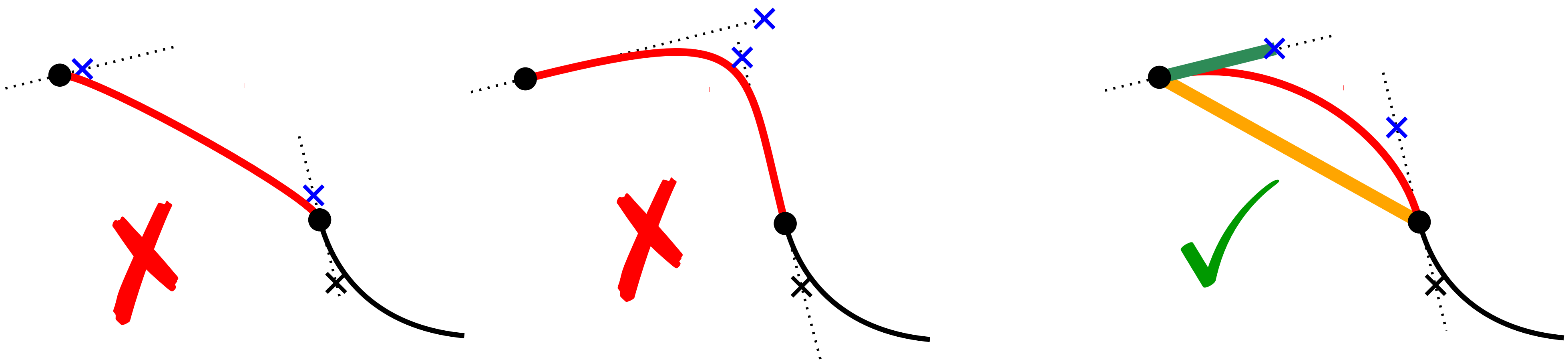
Forces – Shape of Curves



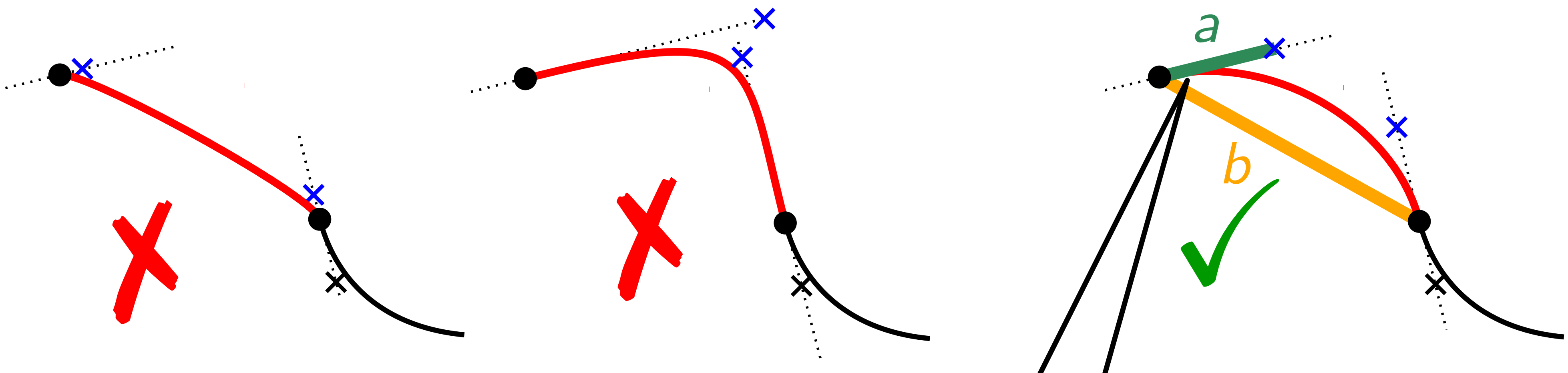
Forces – Shape of Curves



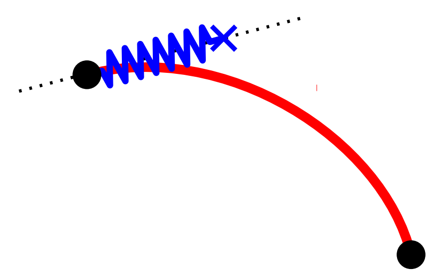
Forces – Shape of Curves



Forces – Shape of Curves

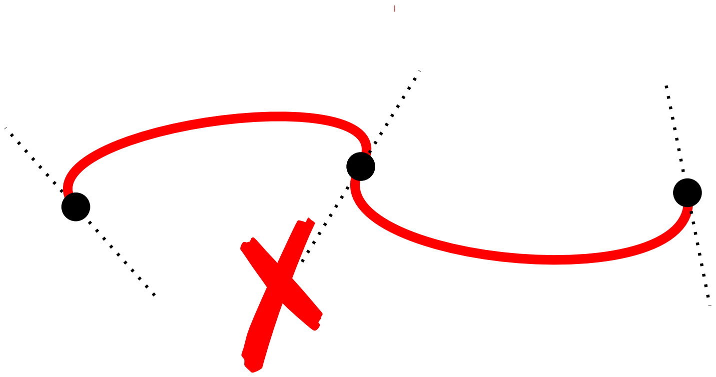


distances: $\frac{a}{b} \approx \frac{1}{3}$

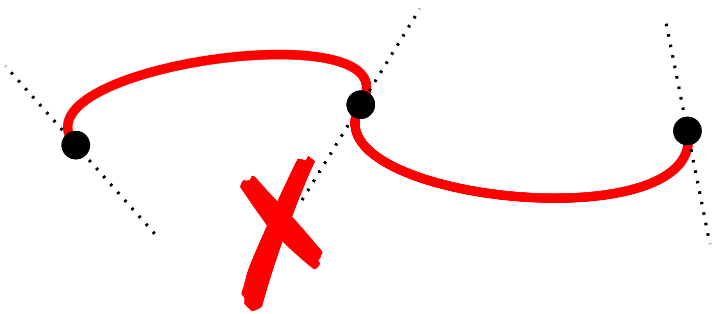


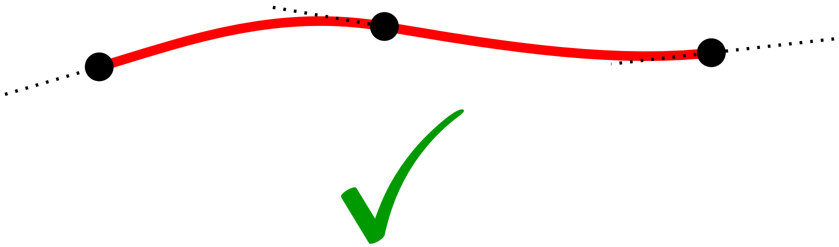
use Fruchterman-Rheingold-like spring force

Forces – Straightening Curves

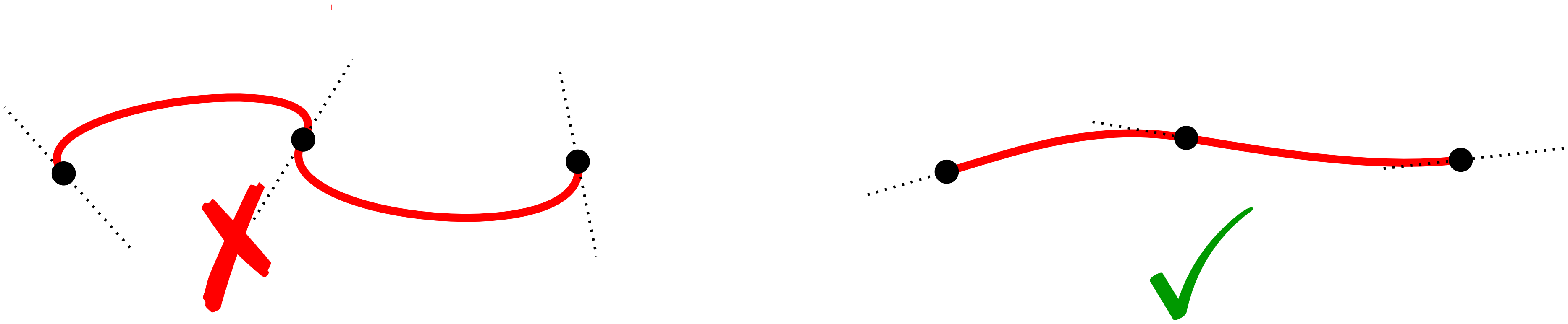


Forces – Straightening Curves



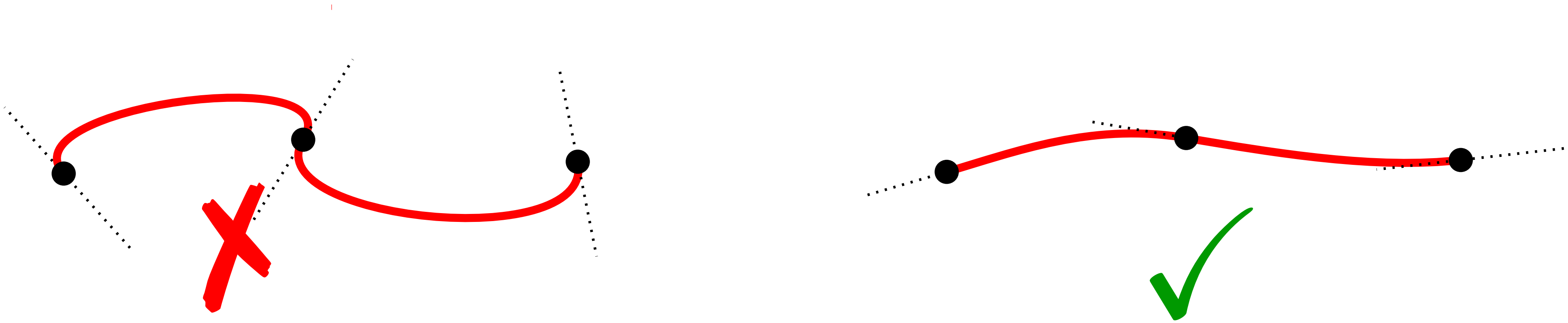


Forces – Straightening Curves

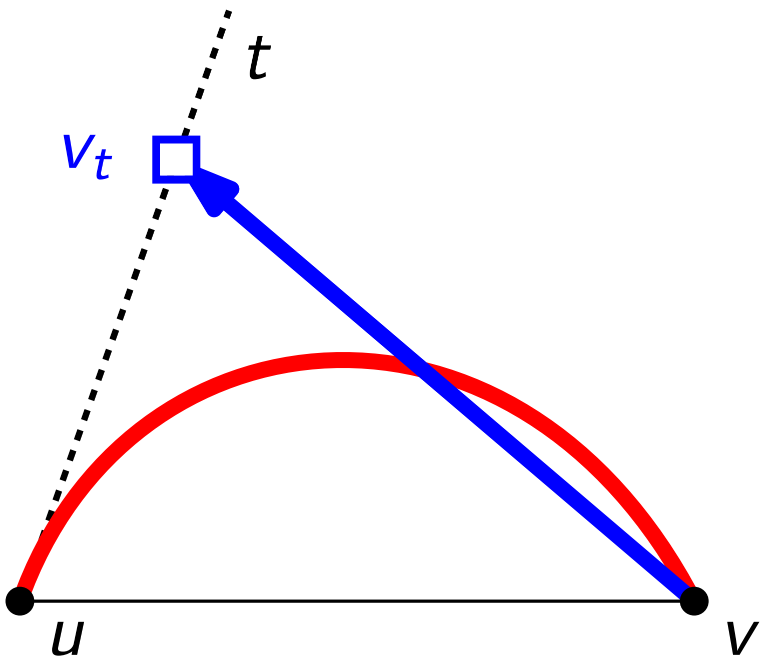


straight-line segment = simplest curve

Forces – Straightening Curves

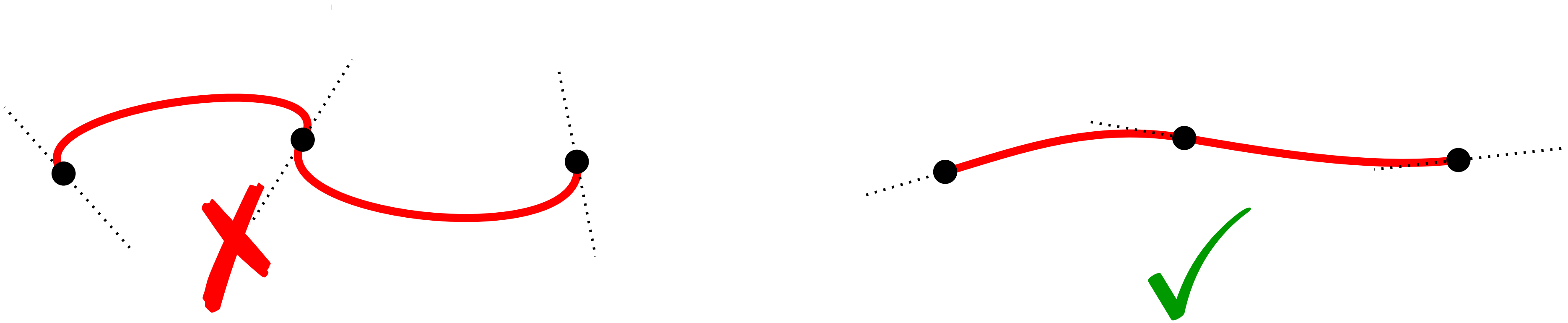


straight-line segment = simplest curve

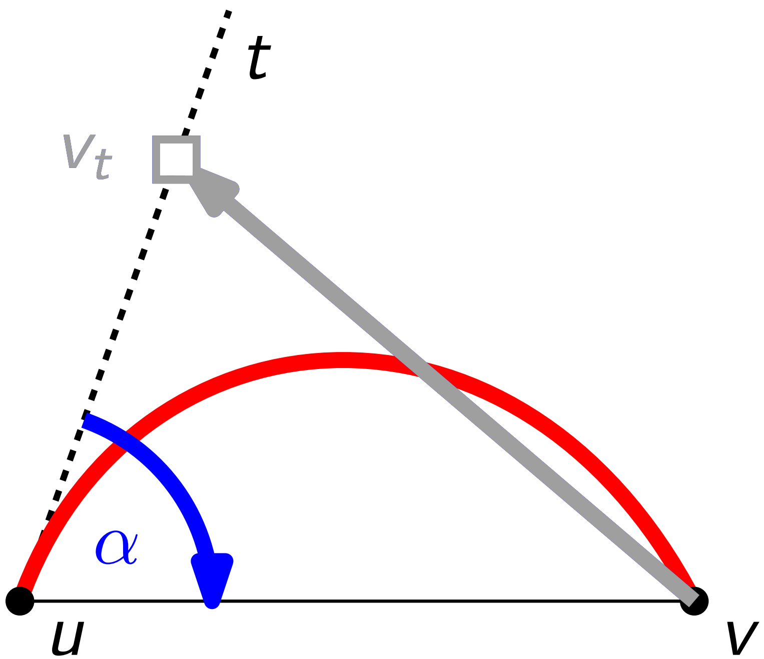


- move vertex v towards tangent

Forces – Straightening Curves

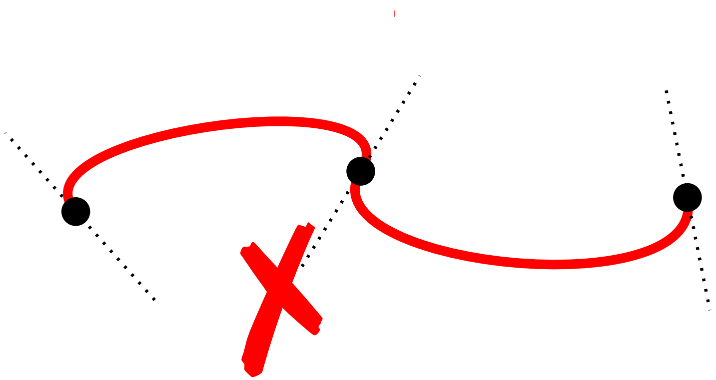


straight-line segment = simplest curve

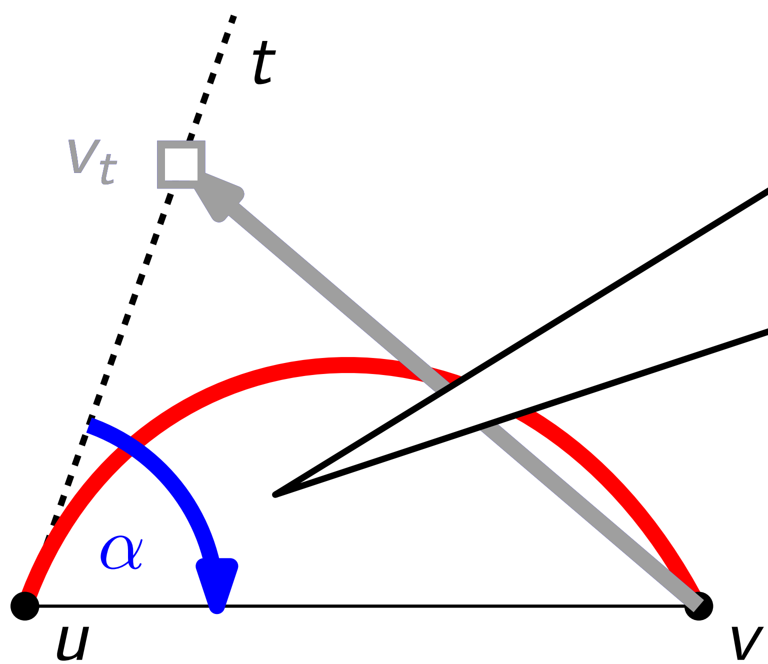


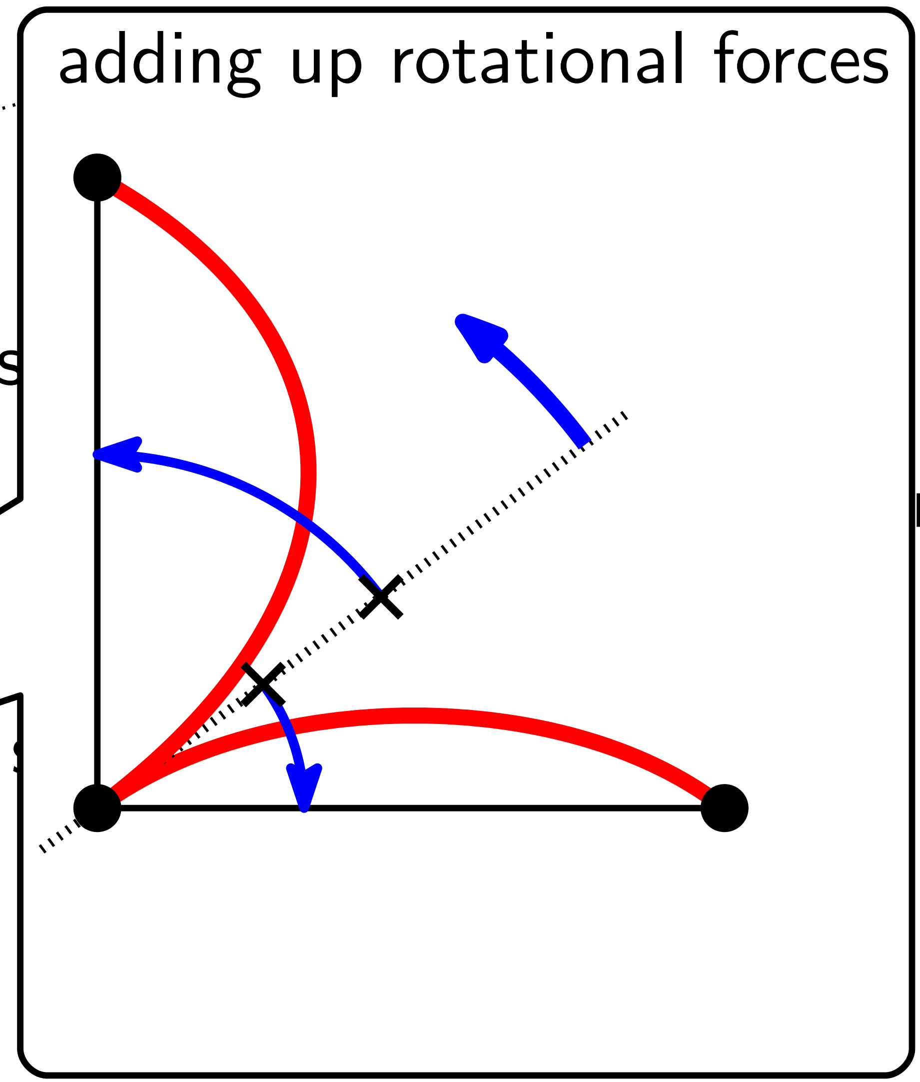
- move vertex v towards tangent
- rotate tangent towards straight-line

Forces – Straightening Curves



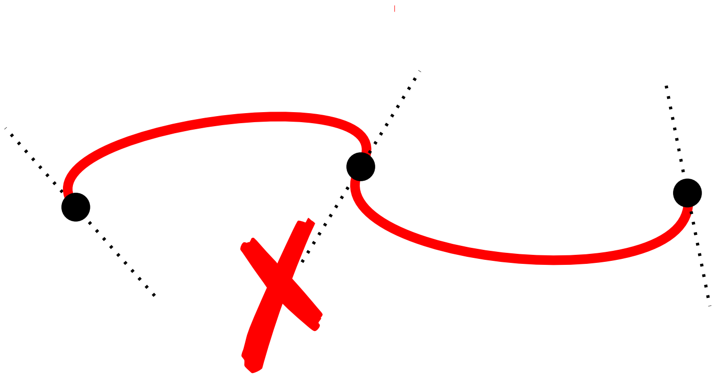
straight-line segment = simples



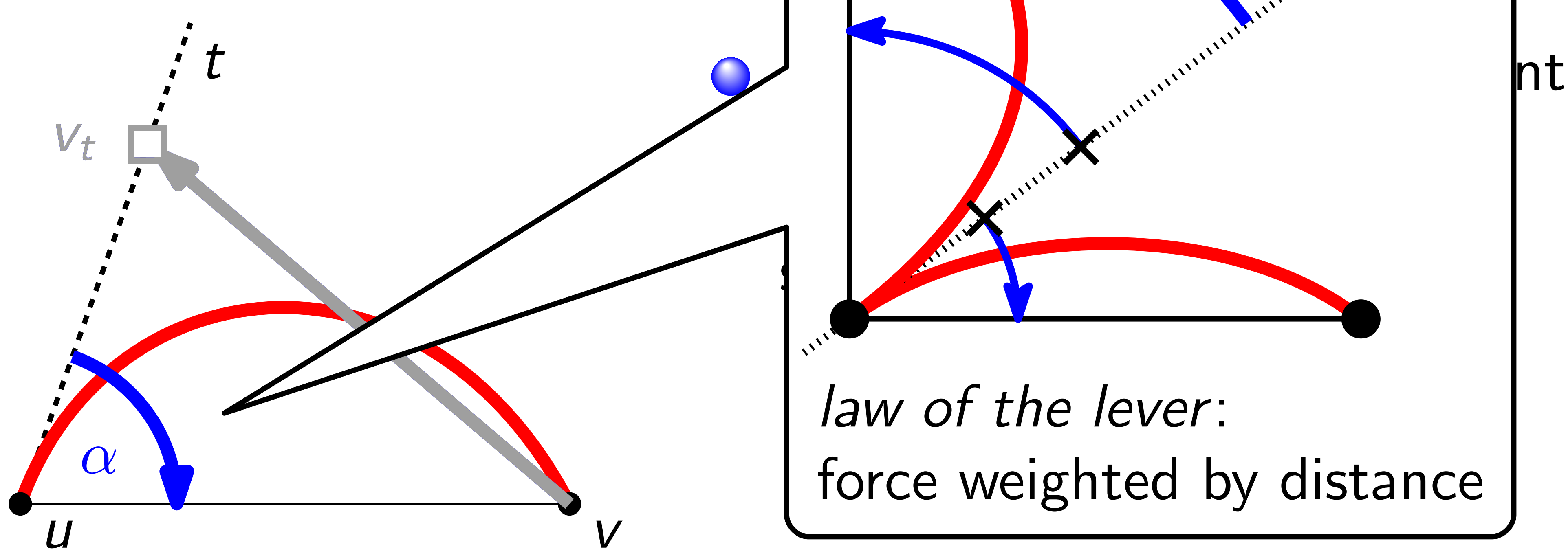


nt

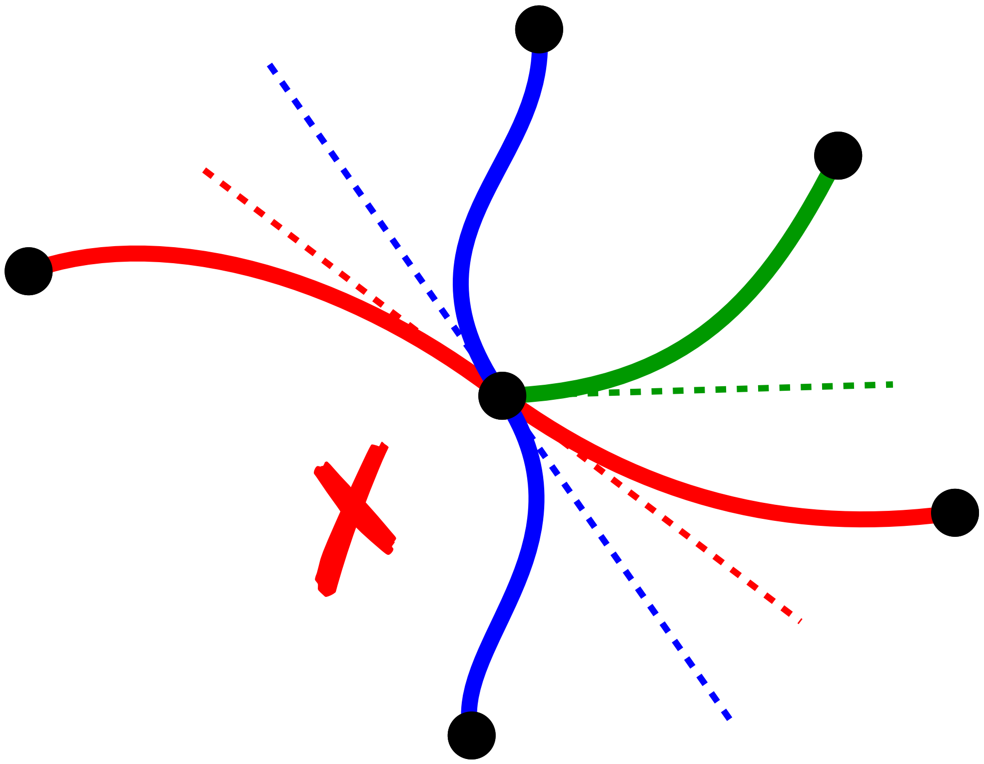
Forces – Straightening Curves



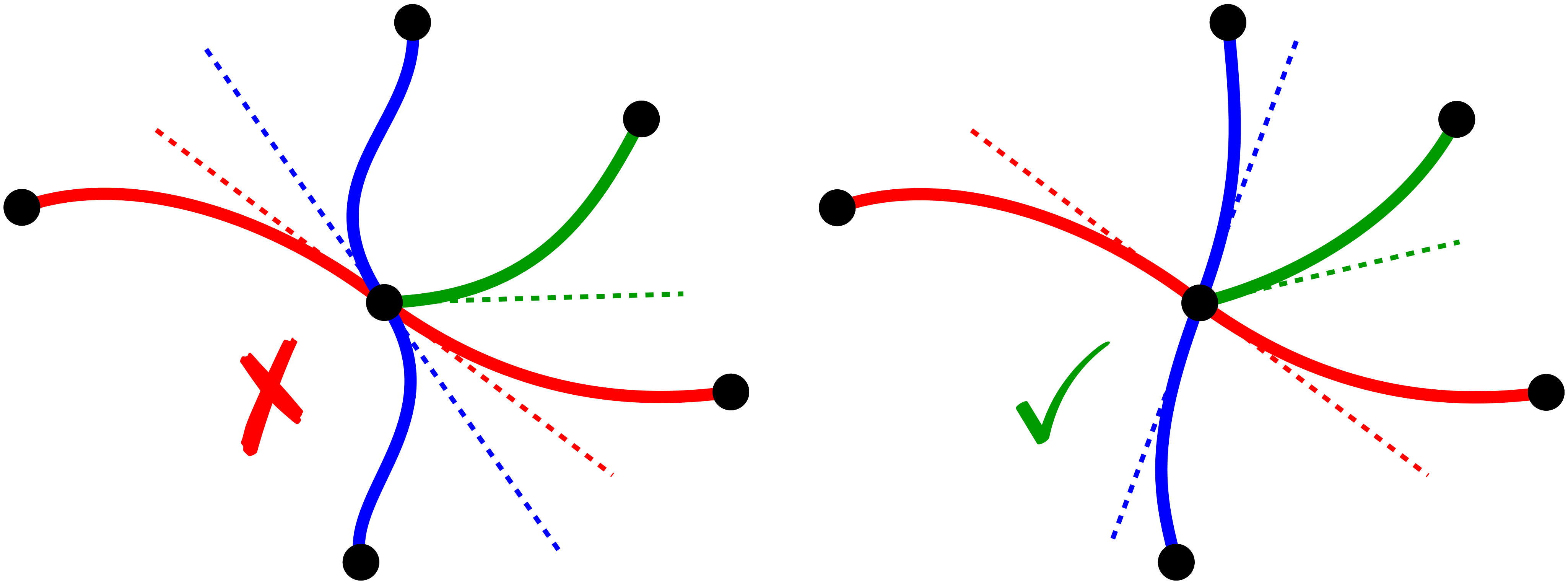
straight-line segment = simples



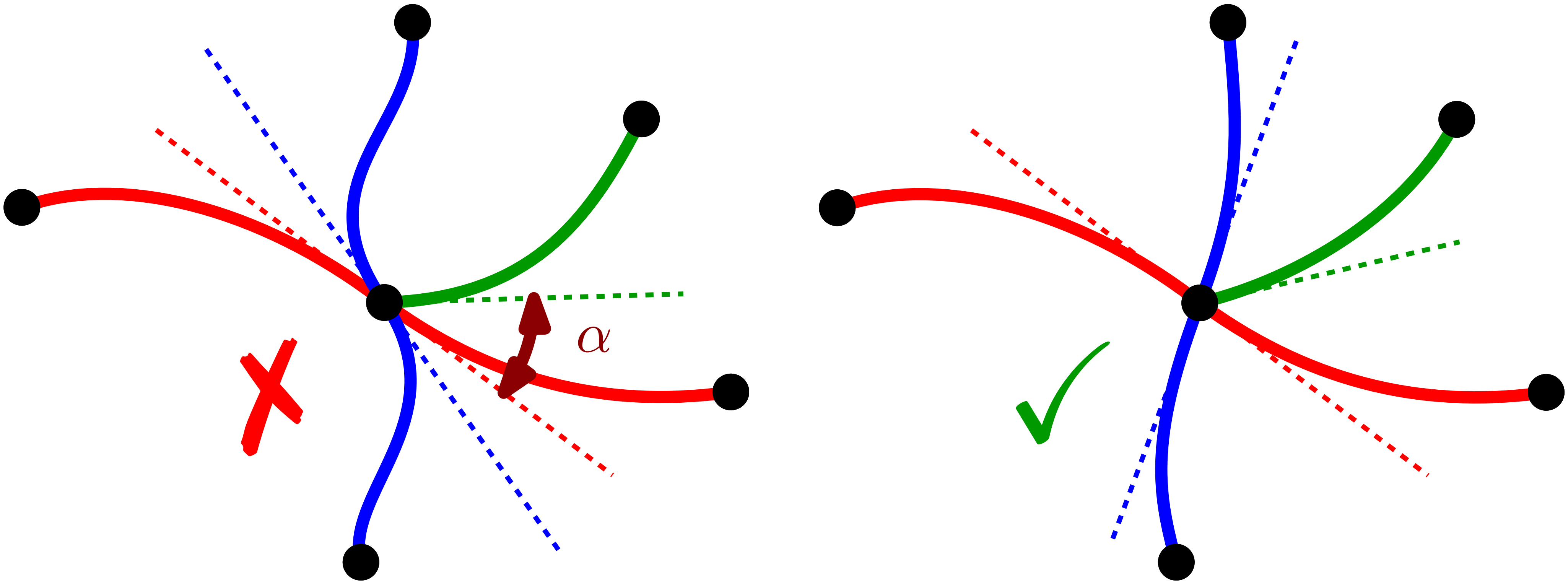
Forces – Angular Resolution



Forces – Angular Resolution

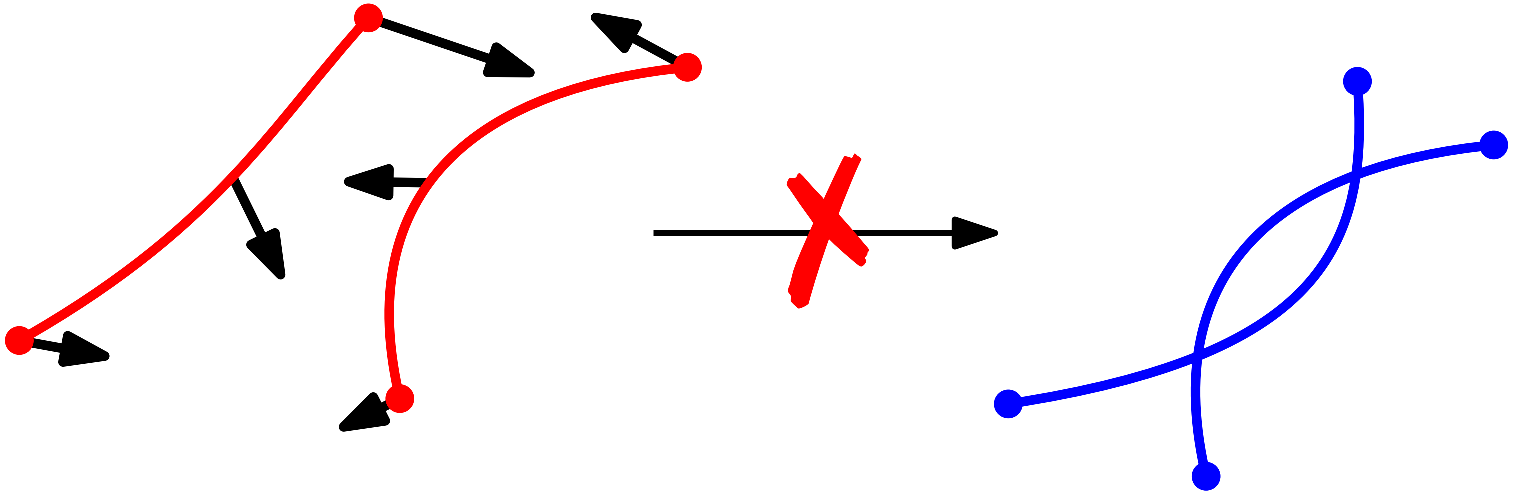


Forces – Angular Resolution

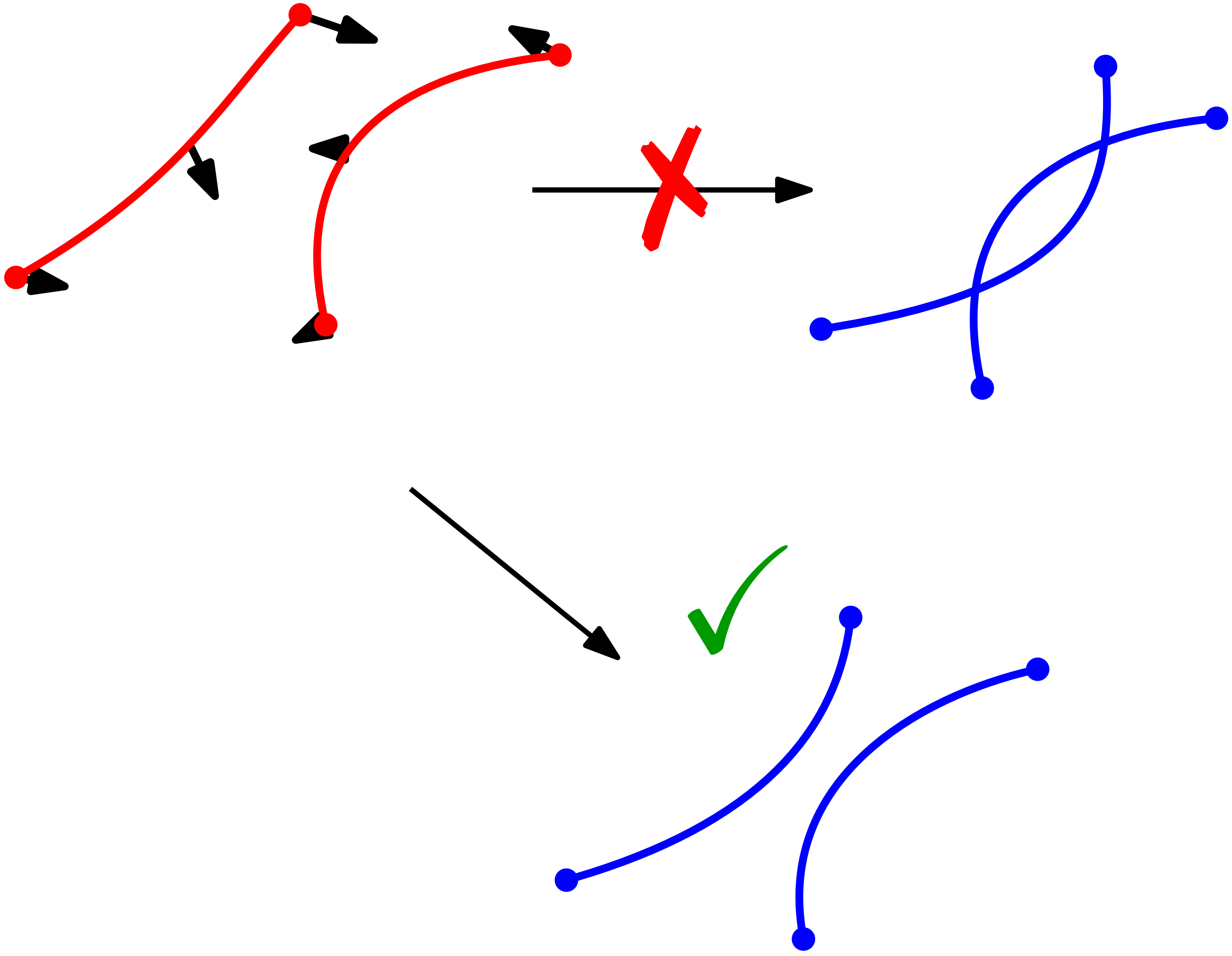


- tangents repelling each other
- force $=$ const. $\cdot \frac{1}{\alpha}$

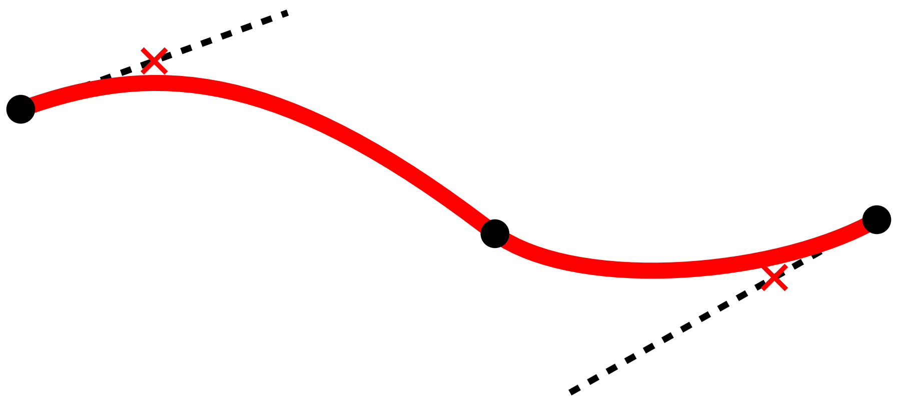
Avoiding Intersections



Avoiding Intersections

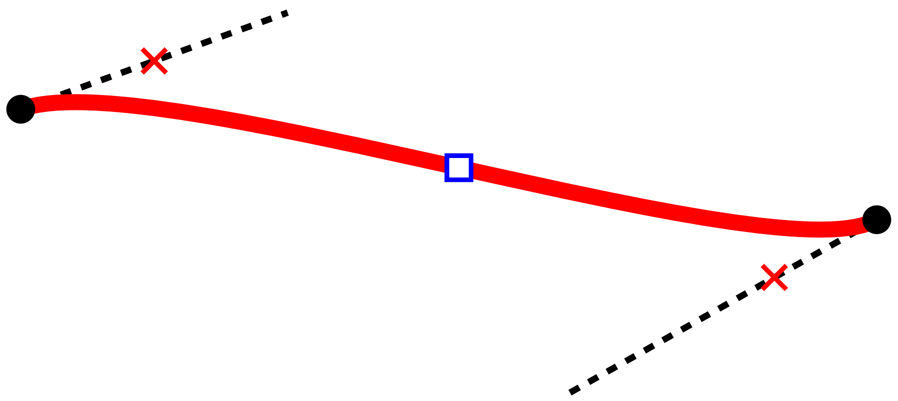


Merging Curves



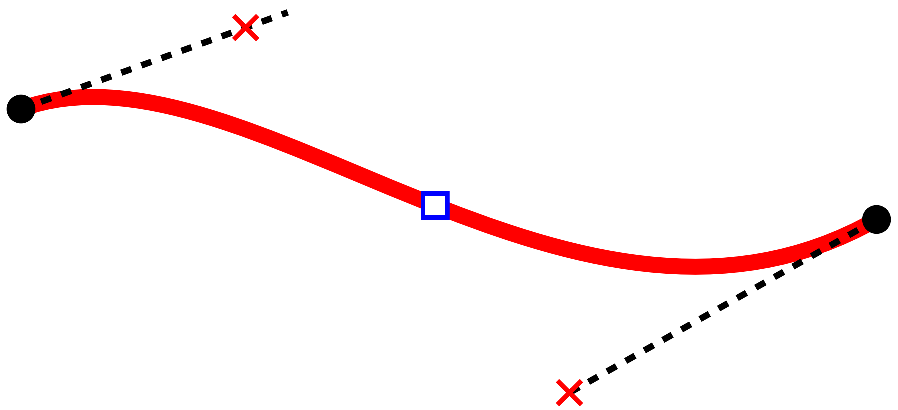
- intermediate stop

Merging Curves



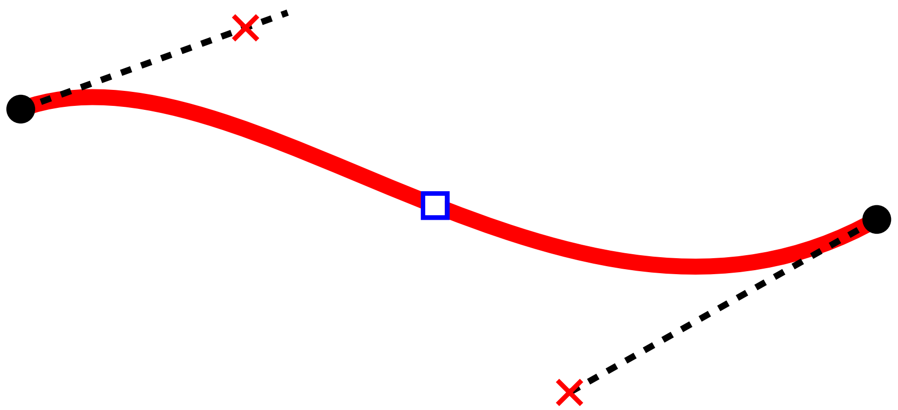
- intermediate stop
- keep control points

Merging Curves



- intermediate stop
- keep control points
- different distances for keeping the drawing crossing-free

Merging Curves

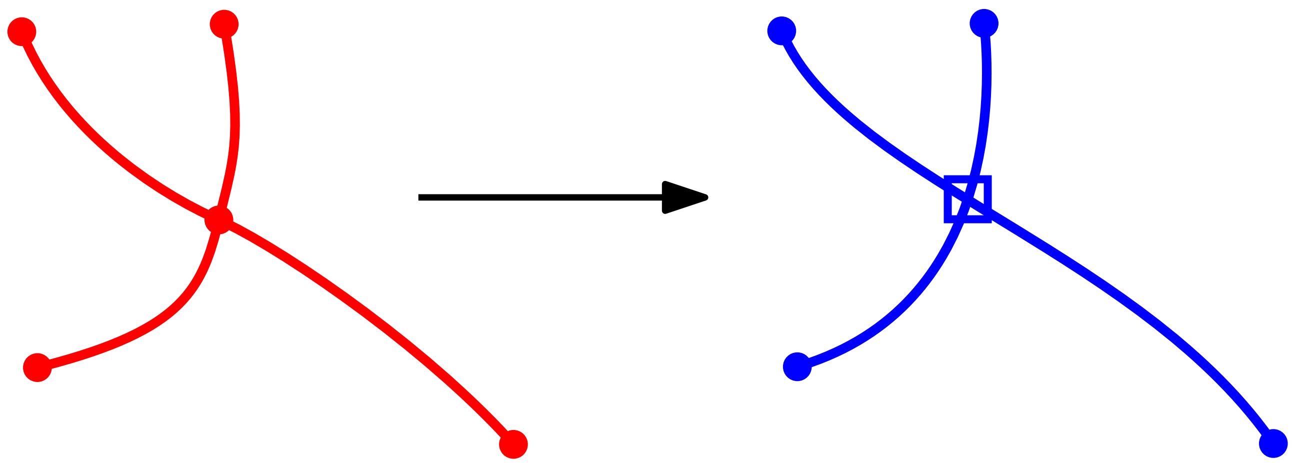


- intermediate stop
- keep control points
- different distances for keeping the drawing crossing-free

Tests: not necessary

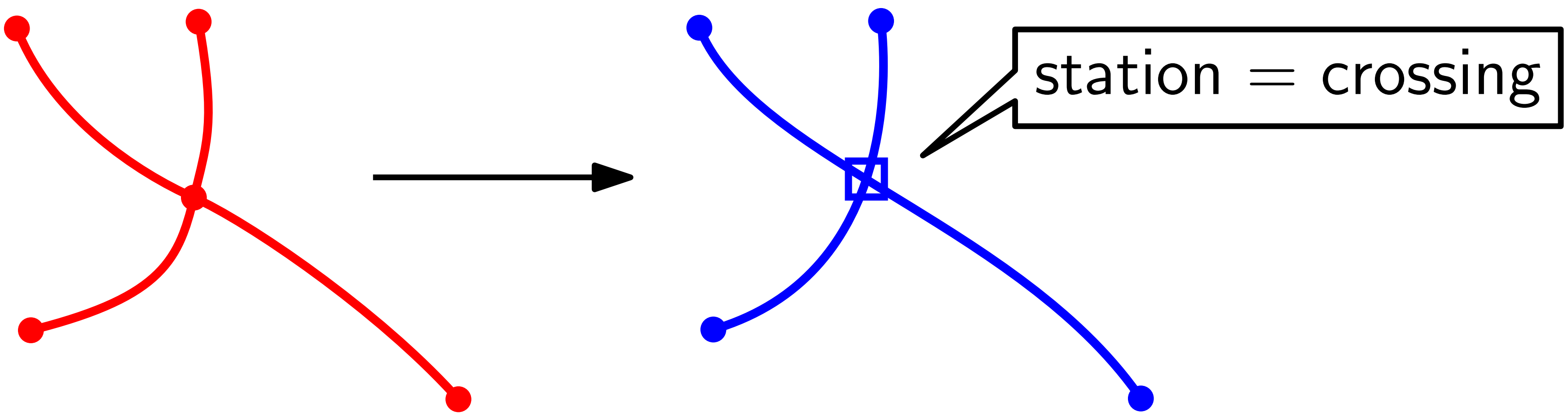
Merging Curves in Interchange Stations

merge 4 curves to 2



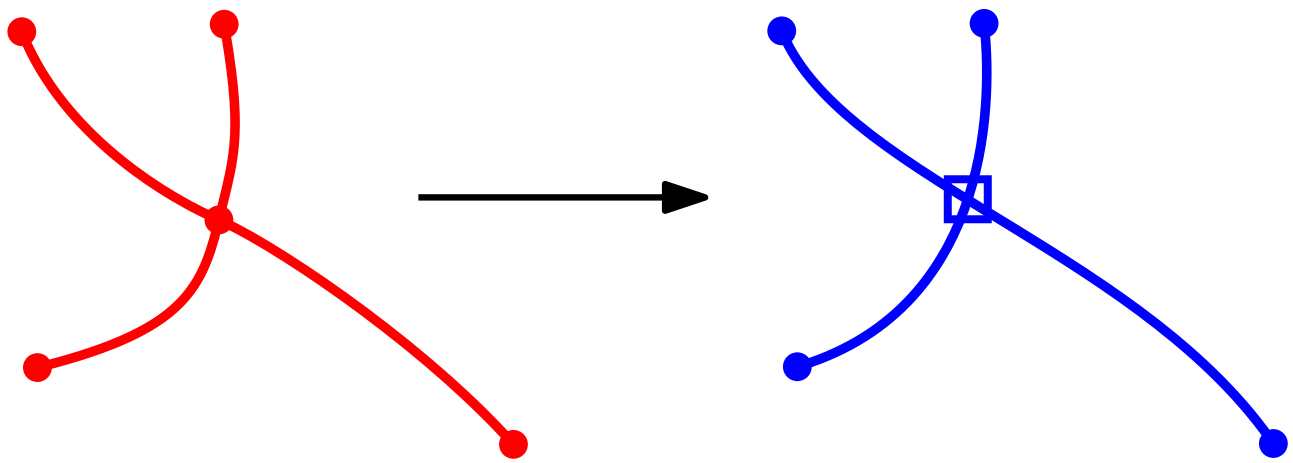
Merging Curves in Interchange Stations

merge 4 curves to 2



Merging Curves in Interchange Stations

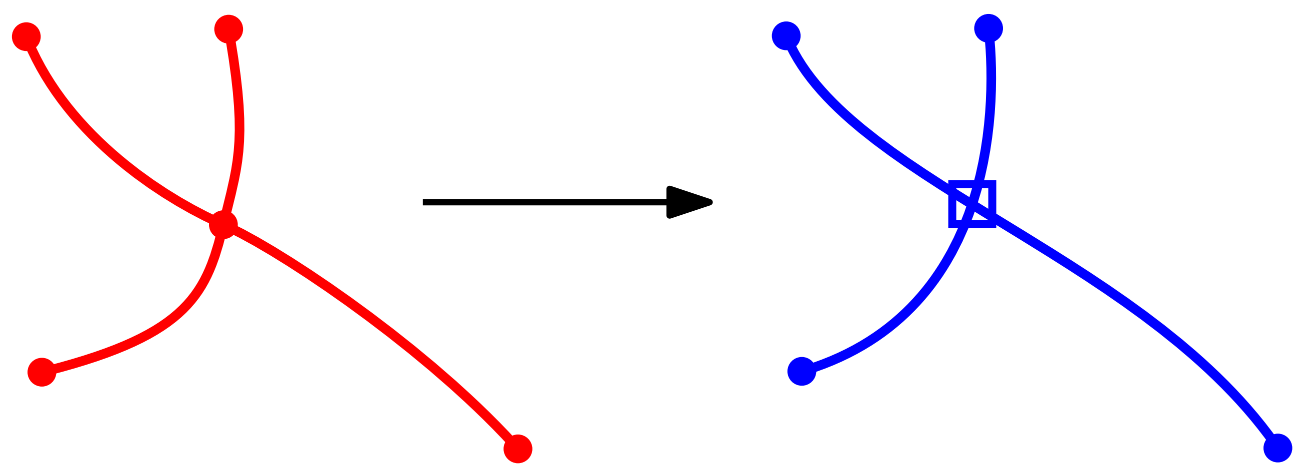
merge 4 curves to 2



- edge proportions should be kept approximately
- testing different control point distances necessary

Merging Curves in Interchange Stations

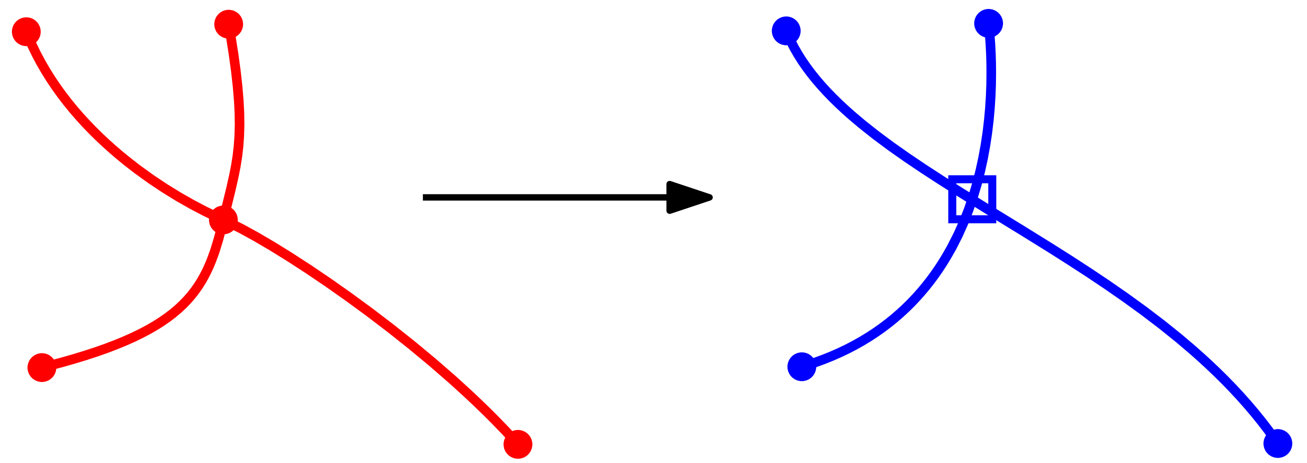
merge 4 curves to 2



- edge proportions should be kept approximately
- testing different control point distances necessary
- adds many additional constraints

Merging Curves in Interchange Stations

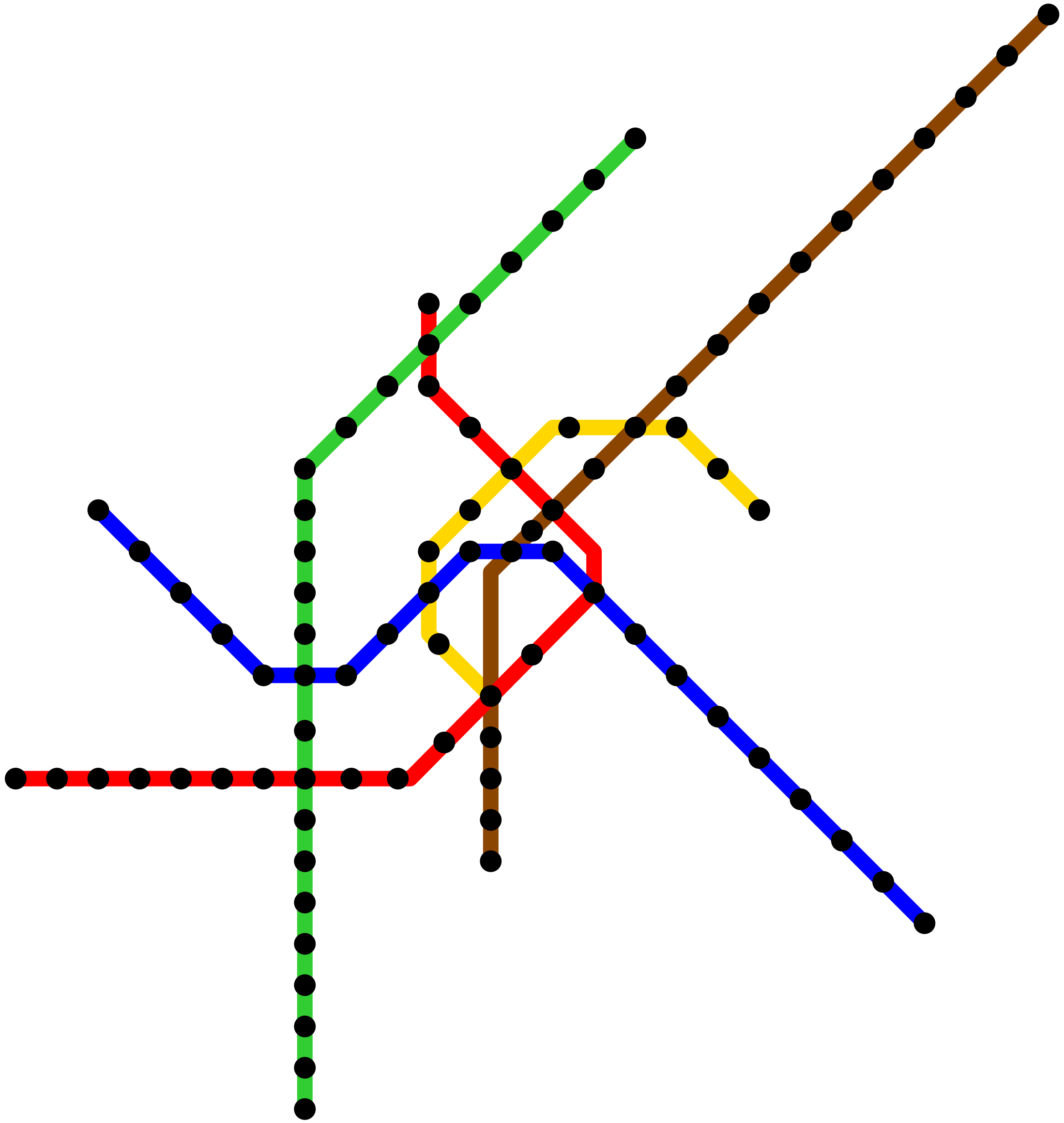
merge 4 curves to 2



- edge proportions should be kept approximately
- testing different control point distances necessary
- adds many additional constraints

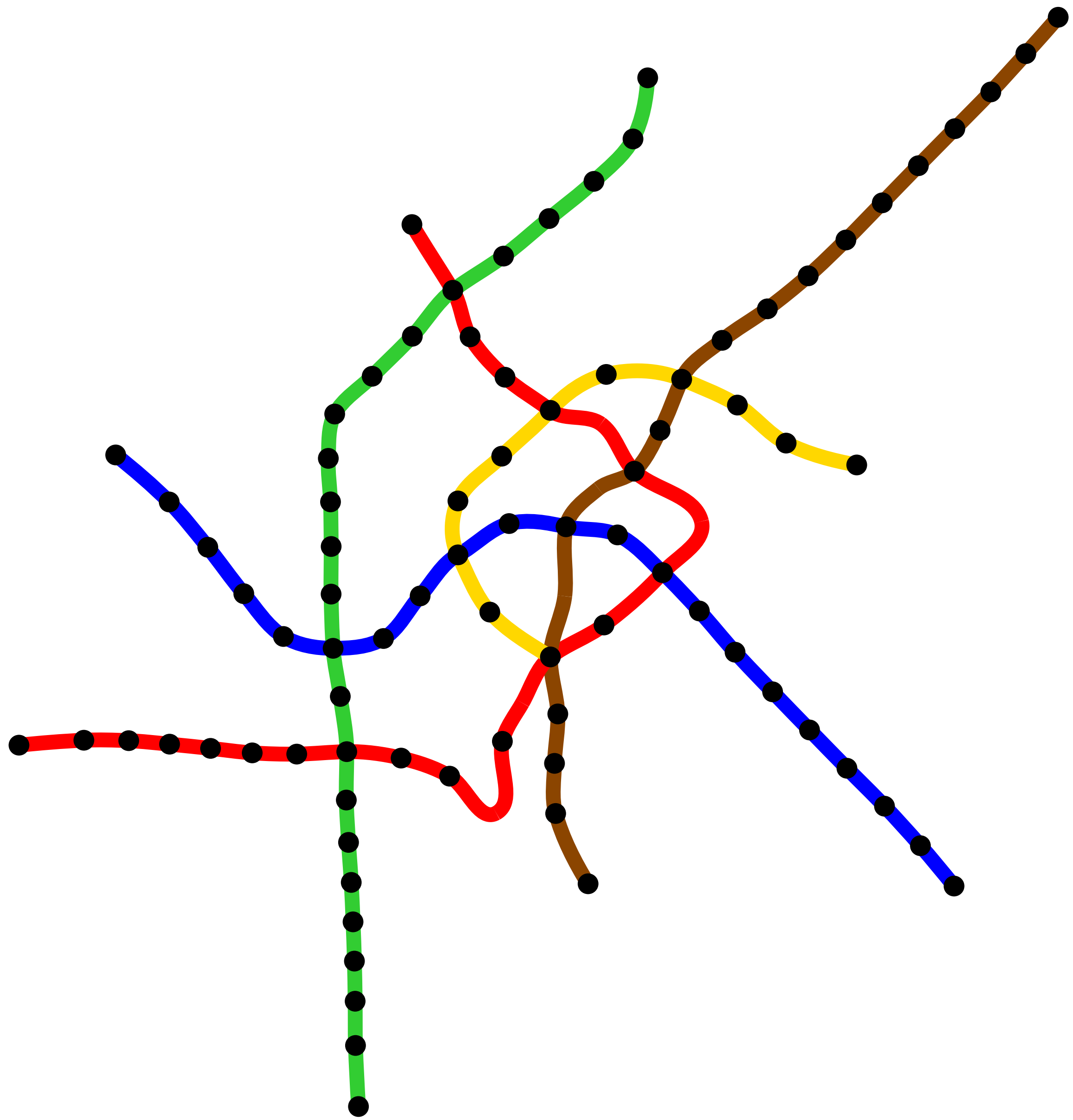
Tests: performed only once at the end

Test Case – Vienna



octilinear map

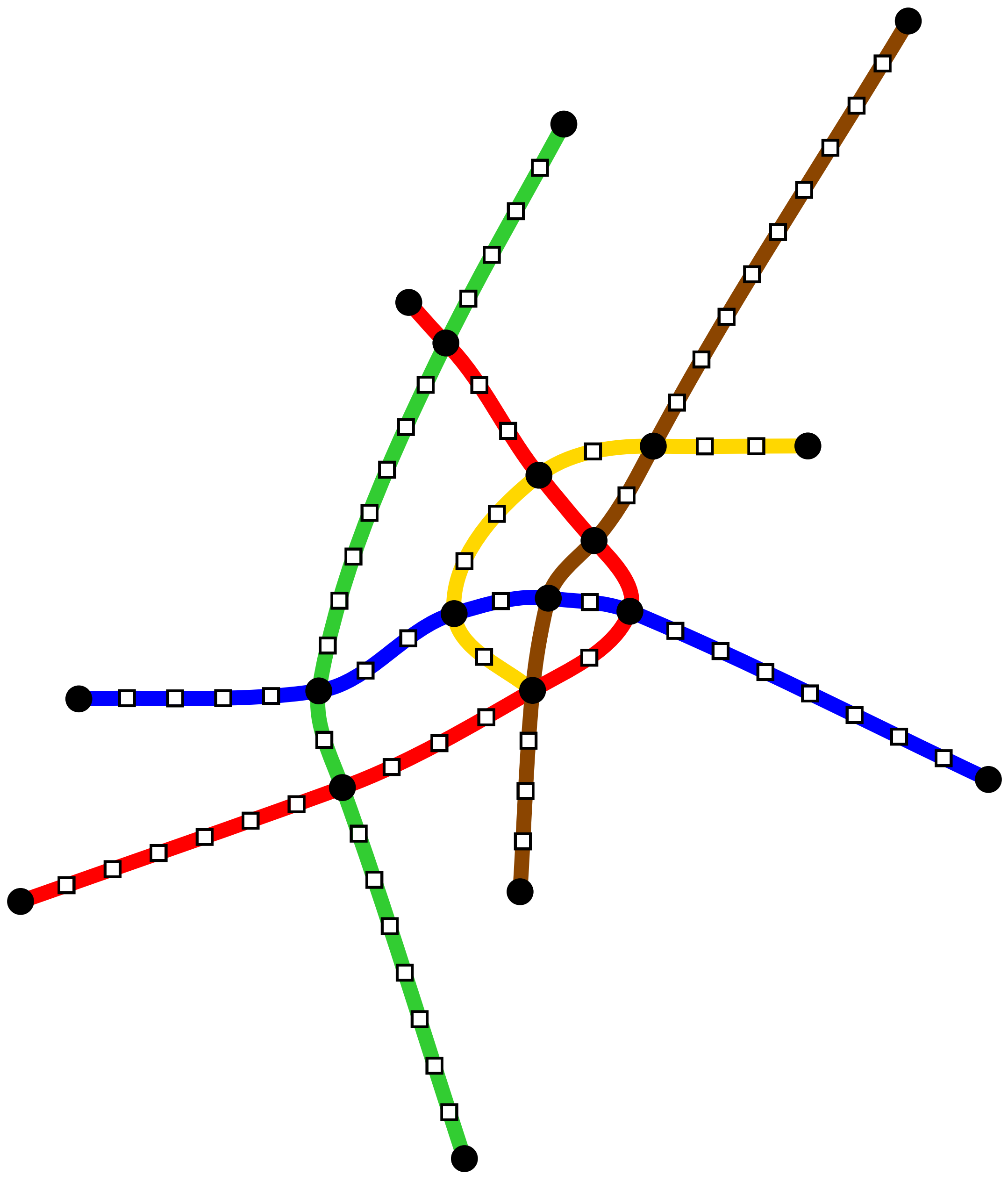
Test Case – Vienna



without merging edges

90 curves

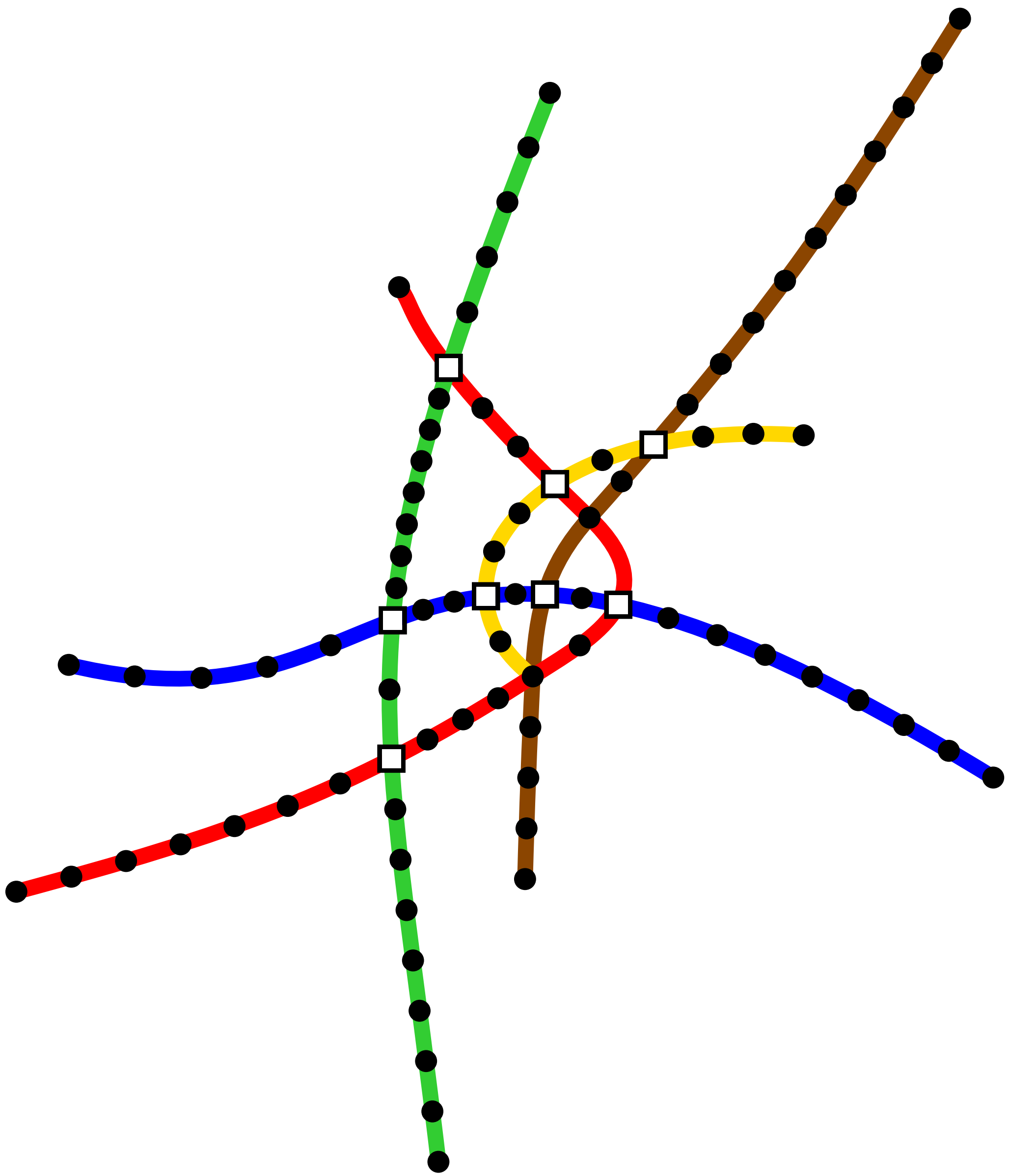
Test Case – Vienna



with merging edges at intermediate stations

25 curves

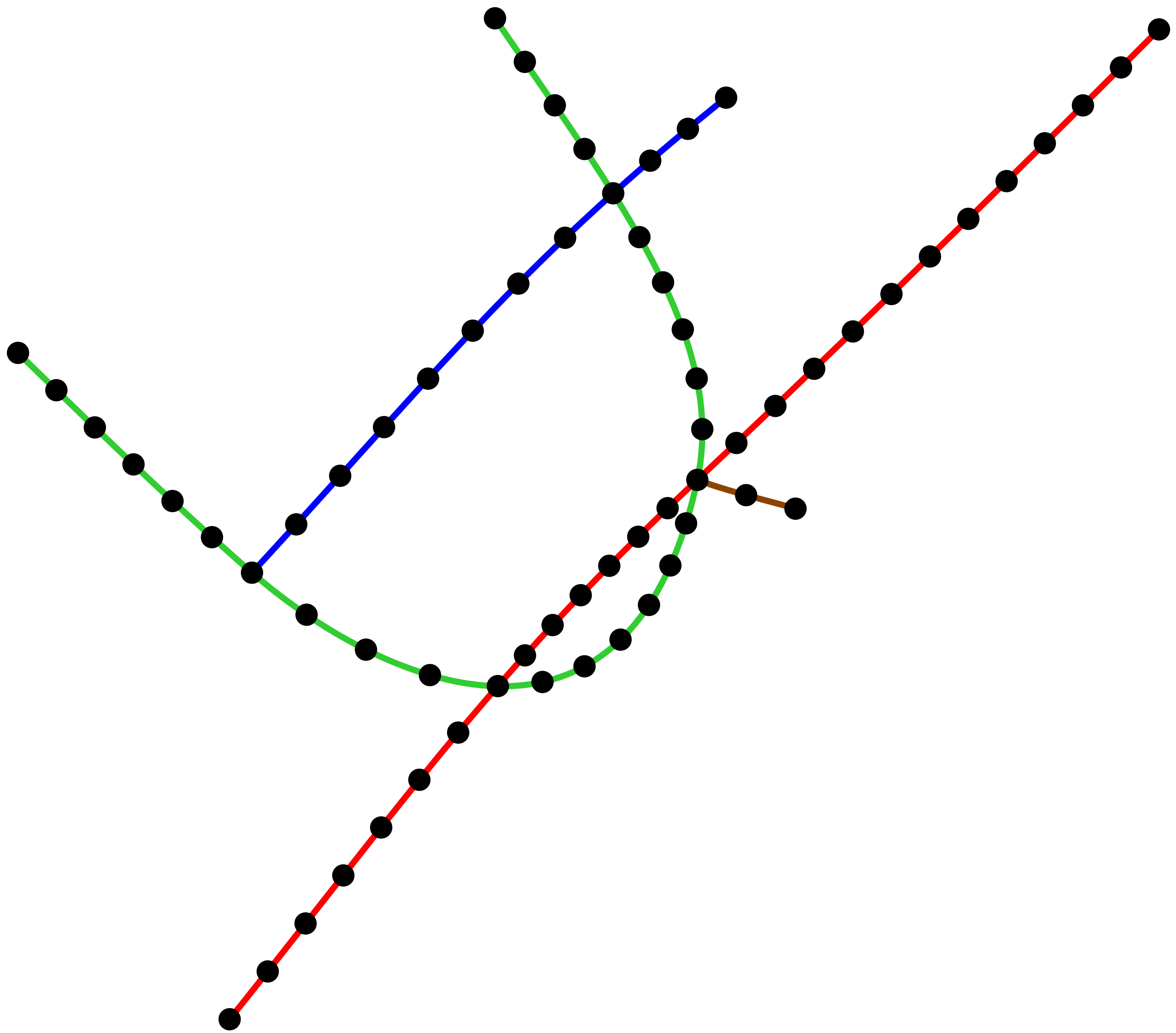
Test Case – Vienna



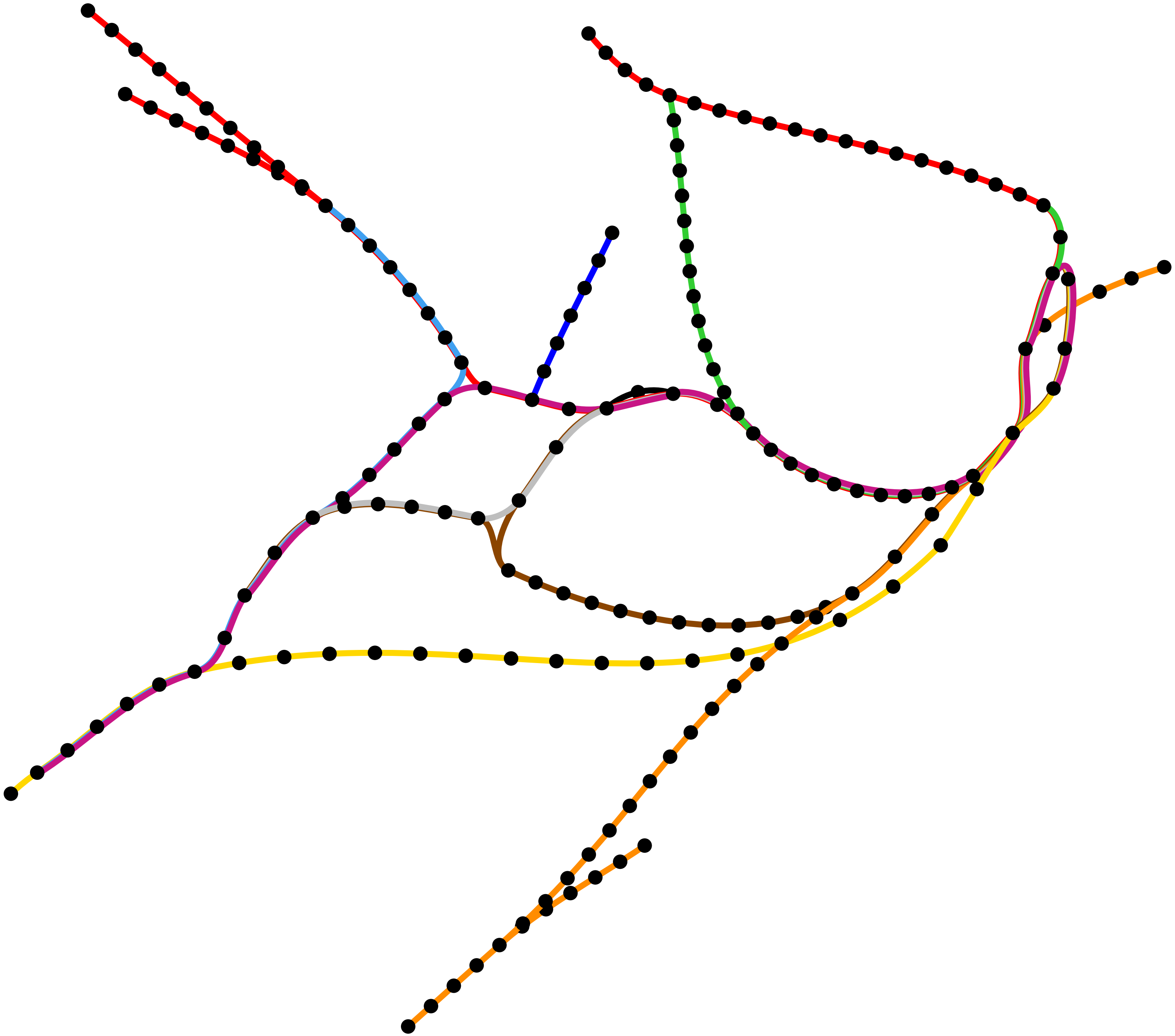
additionally merging edges at interchange stations (degree 4)

9 curves

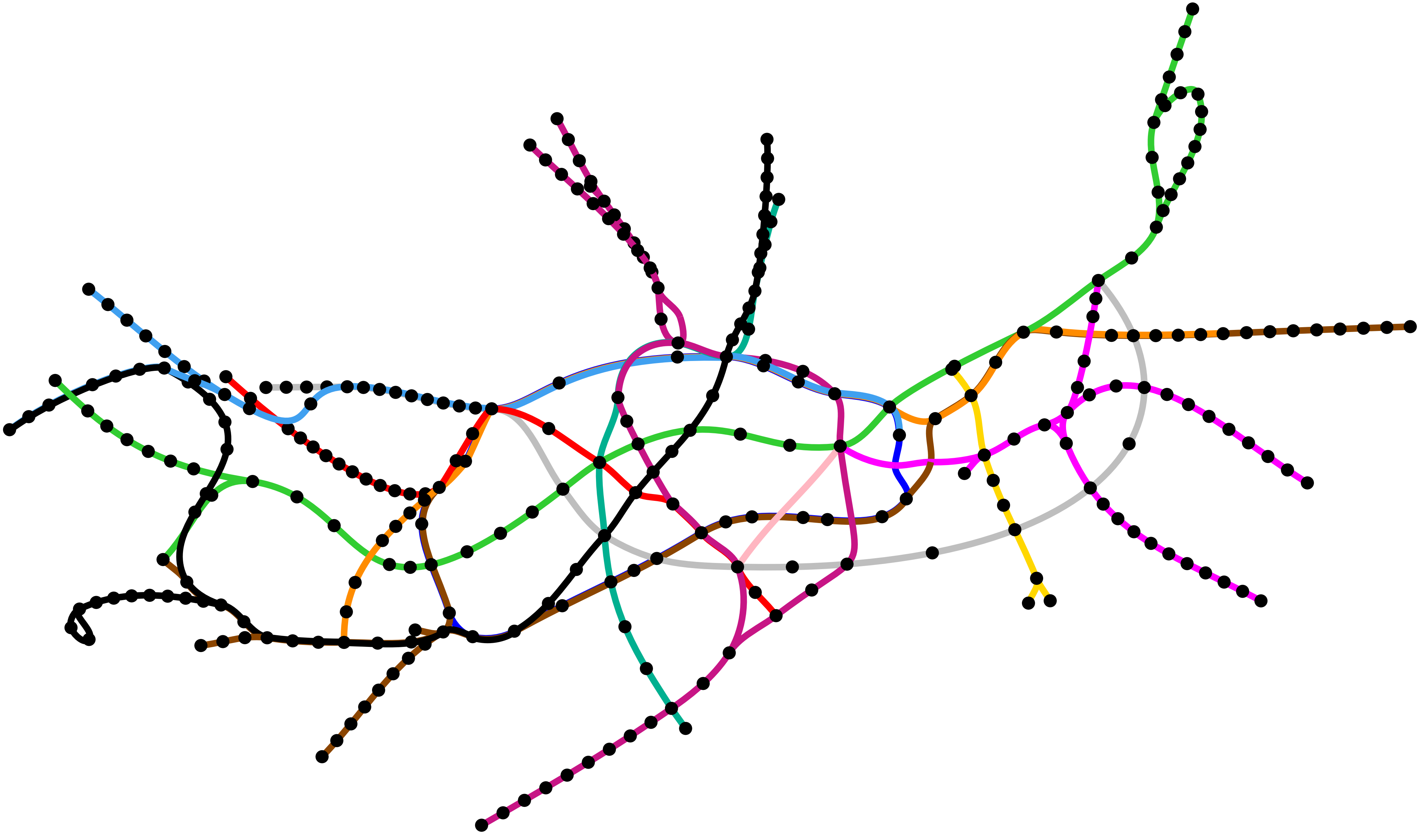
Montréal



Sydney



London



Observations and Conclusion

- works well on smaller networks/sparse regions

Observations and Conclusion

- works well on smaller networks/sparse regions
- dense region like city centers are more problematic
- minimizing the number of curves is important

Observations and Conclusion

- works well on smaller networks/sparse regions
- dense region like city centers are more problematic
- minimizing the number of curves is important
- further minimization with local changes very hard

Observations and Conclusion

- works well on smaller networks/sparse regions
- dense region like city centers are more problematic
- minimizing the number of curves is important
- further minimization with local changes very hard
- Idea:
  Use global approximation of lines by continuos curves – subject to constraints