ETH zürich

From Raster to Vector in OpenStreetMap: Techniques for Automated Stylesheet Transformation

Felix Ludwig

Outline

Raster Maps

Raster Maps

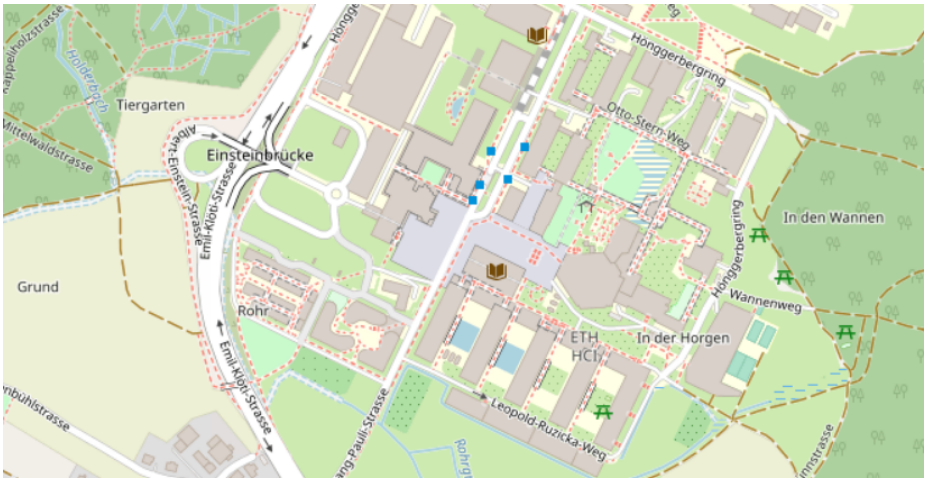
- Starting point

Raster Maps

- Starting point
- Established stylesheet: OSM carto

Raster Maps

- Starting point
- Established stylesheet: OSM carto



Vector Maps

Vector Maps

- Next step of cartography

Vector Maps

- Next step of cartography
- Require more client-side rendering resources

Vector Maps

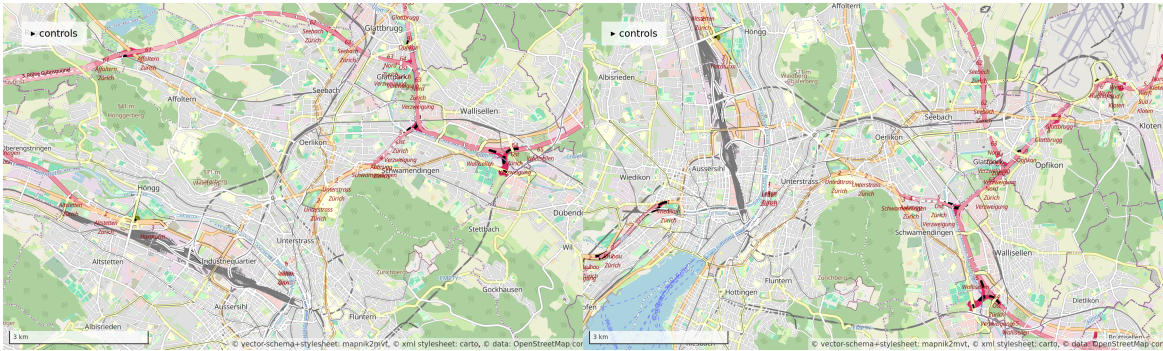
- Next step of cartography
- Require more client-side rendering resources
- Rotation

Vector Maps

- Next step of cartography
- Require more client-side rendering resources
- Rotation
- Multilanguage labels, dynamic styling

Vector Maps

- Next step of cartography
- Require more client-side rendering resources
- Rotation
- Multilanguage labels, dynamic styling



mapnik2mvt

mapnik2mvt

- Automatically convert raster $\rightarrow$ vector

mapnik2mvt

- Automatically convert raster $\rightarrow$ vector
- Stylesheet development continues in MSS

mapnik2mvt

- Automatically convert raster $\rightarrow$ vector
- Stylesheet development continues in MSS
- Add interactive features

Outline

1. Motivation
2. Concepts and architecture
3. Methodology
4. Results and Discussion
5. Questions

XML (first input)

```
<Map><Layer><Datasource><Parameter name="type">postgis
          </Parameter>
          <Parameter name="table">SELECT feature,way
              FROM planet_osm_line WHERE highway='track'
          </Parameter>
        </Datasource>
        <StyleName>firststyle</StyleName>
    </Layer>
    <Style name="firststyle">
    ...
    </Style>
</Map>
```

XML (first input)

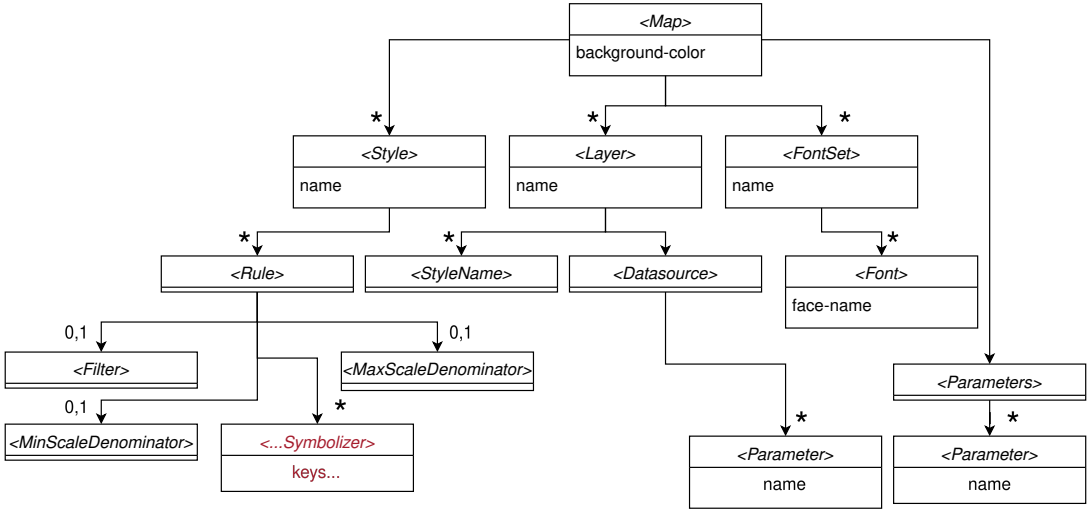
```
<Style name="firststyle"><Rule>
        <Filter>[tracktype]='grade2'</Filter>
        <LinePatternSymbolizer file="symbols/embankment.svg"/>
        <PolygonSymbolizer fill-color="red"/>
    </Rule><Rule>
        <Filter>[tracktype]='grade3'</Filter>
        <PolygonSymbolizer fill-color="#549820"/>
    </Rule><Rule>
        <Filter>([tracktype]!='grade2')
            and ([tracktype]!='grade3')</Filter>
        <LineSymbolizer clip="false" stroke="#ffffff"
            stroke-dasharray="4, 3" stroke-width="3.5" />
    </Rule>
</Style>
```

XML (first input)

```
<Style name="firststyle" filter-mode="first"><Rule>
        <Filter>[tracktype]='grade2'</Filter>
        <LinePatternSymbolizer file="symbols/embankment.svg"/>
        <PolygonSymbolizer fill-color="red"/>
    </Rule><Rule>
        <Filter>[tracktype]='grade3'</Filter>
        <PolygonSymbolizer fill-color="#549820"/>
    </Rule><Rule>


        <LineSymbolizer clip="false" stroke="#ffffff"
            stroke-dasharray="4, 3" stroke-width="3.5" />
    </Rule>
</Style>
```

XML tree


<Map>
background-color
<Style>
name
<Layer>
name
<FontSet>
name
<Rule>
<StyleName>
<Datasource>
<Font>
face-name
0,1
<Filter>
0,1
<MaxScaleDenominator>
0,1
<MinScaleDenominator>
<...Symbolizer>
keys...
<Parameters>
<Parameter>
name
<Parameter>
name
*

MSS (second input)

```
#water-areas {
  [natural = 'glacier']::natural {
    [zoom >= 5] {
      line-width: 1.0;
      line-color: @glacier-line;
      polygon-fill: @glacier;
      [zoom >= 10] {
        line-dasharray: 4,2;
        line-width: 1.5;
      }
    }
  }
}
```

SQL (input+output)

in Layer `water-lines-low-zoom`

```
SELECT way, waterway,
  CASE WHEN tags->'intermittent' IN ('yes')
    OR tags->'seasonal' IN ('yes', 'wet_season', 'dry_season')
    THEN 'yes' ELSE 'no' END AS int_intermittent
FROM planet_osm_line
WHERE waterway = 'river'
```

SQL (input+output)

- Parse into AST, with sqlglot library

in Layer `water-lines-low-zoom`

```
SELECT way, waterway,
  CASE WHEN tags->'intermittent' IN ('yes')
    OR tags->'seasonal' IN ('yes', 'wet_season', 'dry_season')
    THEN 'yes' ELSE 'no' END AS int_intermittent
FROM planet_osm_line
WHERE waterway = 'river'
```

SQL (input+output)

- Parse into AST, with sqlglot library
- Instantiate all zoom integers

in Layer `water-lines-low-zoom`

```
SELECT way, waterway,
  CASE WHEN tags->'intermittent' IN ('yes')
    OR tags->'seasonal' IN ('yes', 'wet_season', 'dry_season')
    THEN 'yes' ELSE 'no' END AS int_intermittent
FROM planet_osm_line
WHERE waterway = 'river'
```

SQL (input+output)

- Parse into AST, with sqlglot library
- Instantiate all zoom integers
- Simplify features away

in Layer `water-lines-low-zoom`

```
SELECT way, waterway,
  CASE WHEN tags->'intermittent' IN ('yes')
    OR tags->'seasonal' IN ('yes', 'wet_season', 'dry_season')
    THEN 'yes' ELSE 'no' END AS int_intermittent
FROM planet_osm_line
WHERE waterway = 'river'
```

SQL (input+output)

- Parse into AST, with sqlglot library
- Instantiate all zoom integers
- Simplify features away
- Add: osm_id, multiple languages

in Layer `water-lines-low-zoom`

```
SELECT way, waterway,
  CASE WHEN tags->'intermittent' IN ('yes')
    OR tags->'seasonal' IN ('yes', 'wet_season', 'dry_season')
    THEN 'yes' ELSE 'no' END AS int_intermittent
FROM planet_osm_line
WHERE waterway = 'river'
```

Outline

From XML

From XML

- Every Symbolizer -> one JSONLayer

From XML

- Every Symbolizer -> one JSONLayer
- Flickering, slow performance

From XML

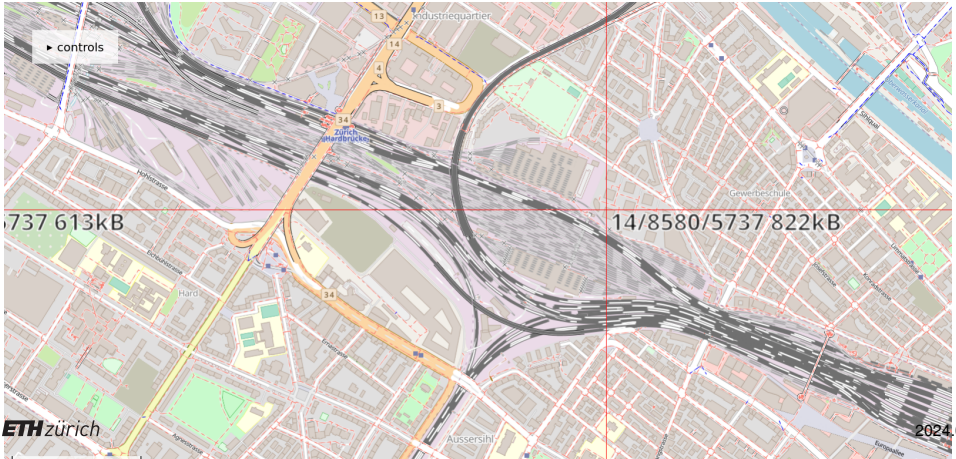
- Every Symbolizer -> one JSONLayer
- Flickering, slow performance
- $\Longrightarrow$ Merging

From XML

- Every Symbolizer -> one JSONLayer
- Flickering, slow performance
- $\Longrightarrow$ Merging
- $\Longrightarrow$ Order is disturbed

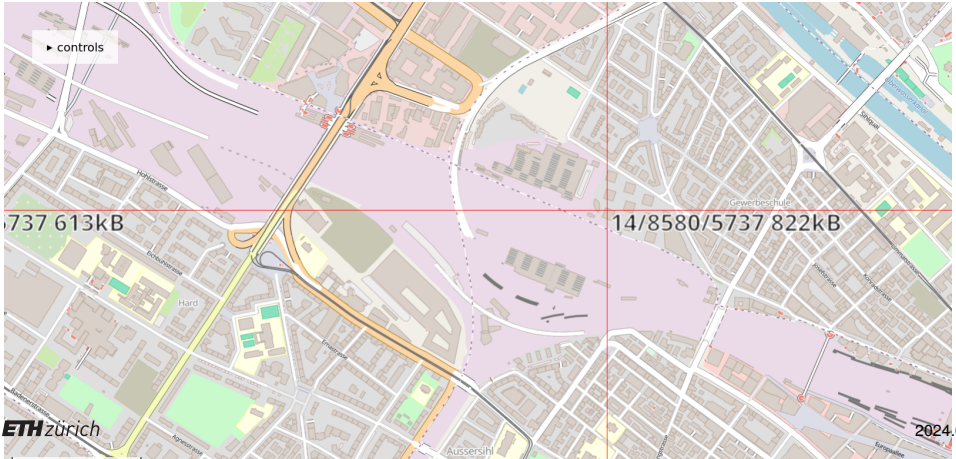
From XML

- Every Symbolizer -> one JSONLayer
- Flickering, slow performance
- $\Longrightarrow$ Merging
- $\Longrightarrow$ Order is disturbed



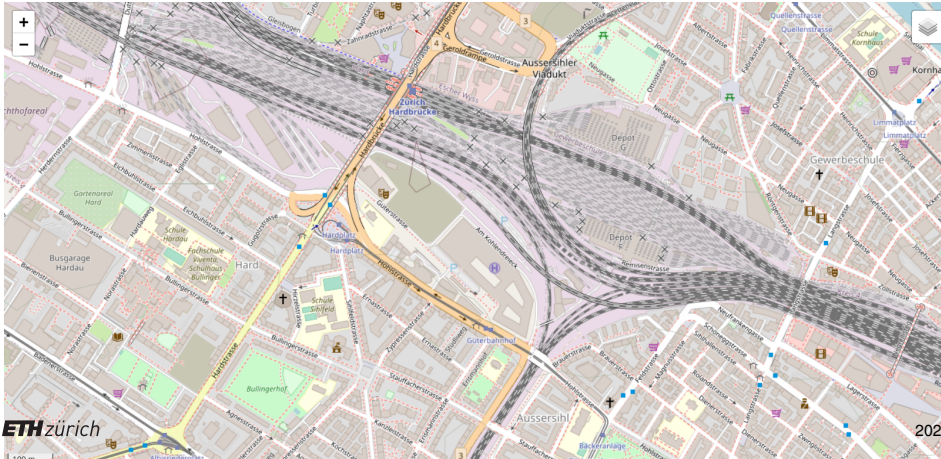
From XML (compare to vector from mss)

- Every Symbolizer -> one JSONLayer
- Flickering, slow performance
- $\Longrightarrow$ Merging
- $\Longrightarrow$ Order is disturbed



From XML (compare to raster)

- Every Symbolizer -> one JSONLayer
- Flickering, slow performance
- $\Longrightarrow$ Merging
- $\Longrightarrow$ Order is disturbed



Converting from MSS

Converting from MSS

- Choose starting points of JSONLayers dynamically

Converting from MSS

- Choose starting points of JSONLayers dynamically
- Seed keys

Converting from MSS

- Choose starting points of JSONLayers dynamically
- Seed keys
- Order is preserved

Outline

Moved approach XML $\rightarrow$ MSS

Moved approach XML $\rightarrow$ MSS

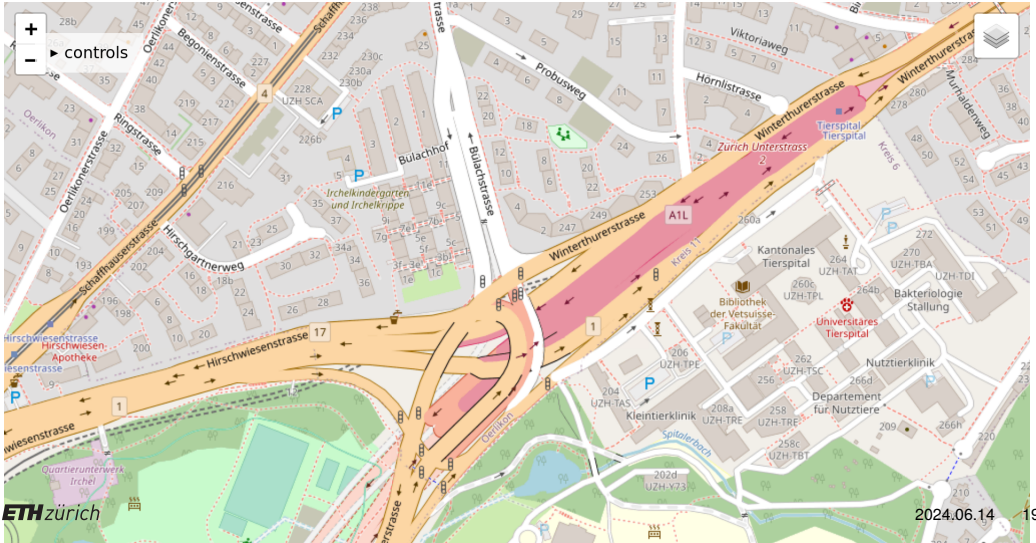
- XML simpler and flat $\rightarrow$ mergers, and order is disturbed

Moved approach XML $\rightarrow$ MSS

- XML simpler and flat $\rightarrow$ mergers, and order is disturbed
- MSS choice of starting points of JSONLayers

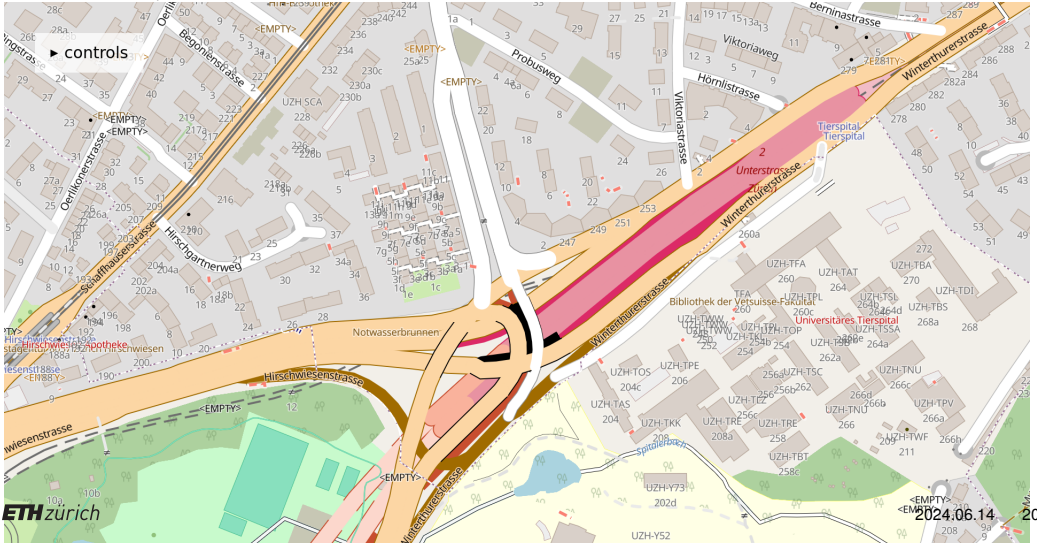
Visual comparisons, 1

Intersection, raster



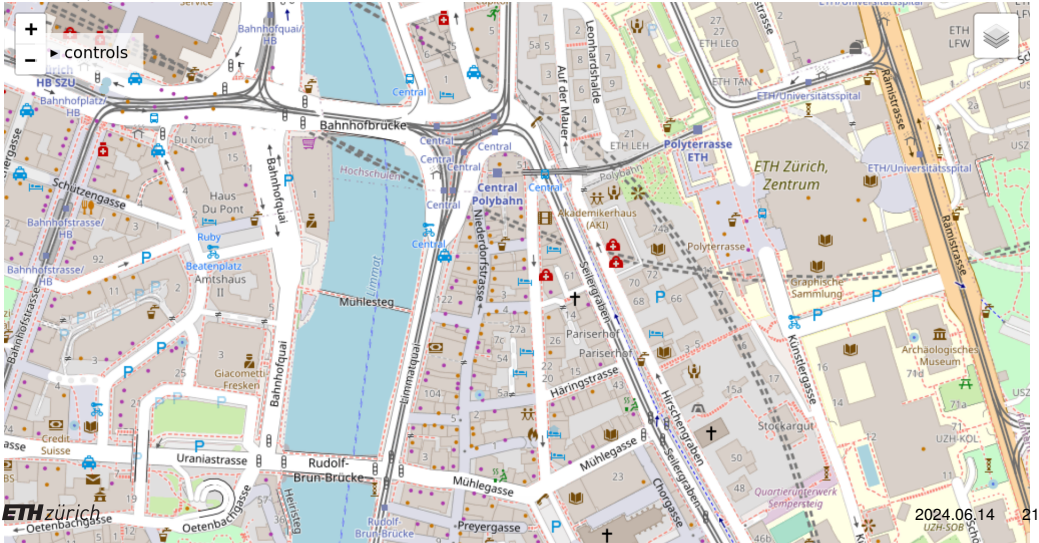
Visual comparisons, 1

Intersection, vector



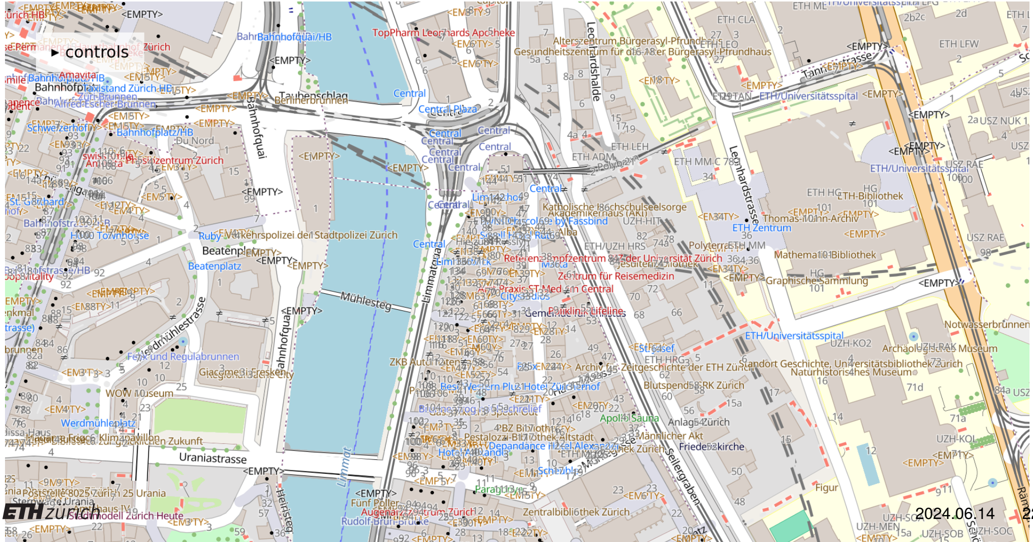
Visual comparisons, 2

Zurich, raster



Visual comparisons, 2

Zurich, vector



Outline

1. Motivation
2. Concepts and architecture
3. Methodology
4. Results and Discussion
5. Questions