From Many User-Contributed Polygons To One Polygon Consensus

Fabian Feitsch

Julius-Maximilians-Universität Würzburg

July 11, 2016
The History of New York
The History of New York
The History of New York

First: Extract the buildings using image processing.

Second: Ask users to fix the extracted polygons.

Third: Have a bunch of similar, yet different polygons.

Fourth: Which footprint was meant by the users?
The History of New York

First: Extract the buildings using image processing.

Second: Ask users to fix the extracted polygons.
The History of New York

First: Extract the buildings using image processing.

Second: Ask users to fix the extracted polygons.

Third: Have a bunch of similar, yet different polygons.
First: Extract the buildings using image processing.
Second: Ask users to fix the extracted polygons.
Third: Have a bunch of similar, yet different polygons.
Fourth: Which footprint was meant by the users?
Table Of Contents

A Geometry-Based Algorithm
   The Pile Algorithm
   Advantages and Disadvantages

Cluster-Based Algorithms
   The Voting Algorithm
   The MMWC Algorithm
   Choice Of Parameters

Evaluation
   Process Of Evaluation
   Semoantical Results
   Brightness Results

Conclusion
The Pile Algorithm

Idea

Return the area that is covered by at least \( k \) polygons.
The Pile Algorithm

Idea
Return the area that is covered by at least $k$ polygons.
The Pile Algorithm

Idea
Return the area that is covered by at least $k$ polygons.
The Pile Algorithm

Idea
Return the area that is covered by at least $k$ polygons.
Implications of the Pile Algorithm
Implications of the Pile Algorithm

- Simple Interpretation
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter $k$
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter $k$
- Scaling does not matter
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter $k$
- Scaling does not matter
- Implementation difficulties
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter $k$
- Scaling does not matter
- Implementation difficulties
- Generates too many corners
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter $k$
- Scaling does not matter
- Implementation difficulties
- Generates too many corners
- Result is not always a simple polygon

The Pile algorithm is not useful for the Building Inspector.
Implications of the Pile Algorithm

- Simple Interpretation
- Takes only one parameter \( k \)
- Scaling does not matter
- Implementation difficulties
- Generates too many corners
- Result is not always a simple polygon

Conclusion
The Pile algorithm is not useful for the Building Inspector.
Strategy of Cluster-Based Algorithms
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal
2. Clustering of Corners
3. Apply actual Algorithm
4. Translate Clusters to Polygon

Claim: Either Step 1 or Step 2 can be omitted. To be shown later...
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters

For example DBSCAN:

- min-eps
- min-pts = 3
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm

For example DBSCAN:

- min-eps
- min-pts = 3
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm

Generate graph from clusters and extract cycle.
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm
4. Translate Clusters to Polygon

Generate graph from clusters and extract cycle.
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm
4. Translate Clusters to Polygon

Use centroids of the clusters in the circle.
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm
4. Translate Clusters to Polygon

Use centroids of the clusters in the circle.

Claim: Either Step 1 or Step 2 can be omitted.
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm
4. Translate Clusters to Polygon

Use centroids of the clusters in the circle.

Claim: Either Step 1 or Step 2 can be omitted.
To be shown later . . .
Strategy of Cluster-Based Algorithms

Four steps derived from Mauricio Arteaga (NYPL):

1. Outlier Removal requires parameters
2. Clustering of Corners requires parameters
3. Apply actual Algorithm \leftarrow \text{Now!}
4. Translate Clusters to Polygon

Claim: Either Step 1 or Step 2 can be omitted. To be shown later . . .
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:

- Orientation of Polygons
- Result can be arbitrarily bad or not existent in the input.
The Voting Algorithm

1. Find cluster \( c \) with most corners.
2. Select neighbor of \( c \) which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:
- Orientation of Polygons
- Result can be arbitrarily bad
- ... or not existent in the input.

- 8 Polygons
- 7 Polygons
- 5 Polygons
The Voting Algorithm

1. Find cluster $c$ with most corners.
The Voting Algorithm

1. Find cluster \( c \) with most corners.
The Voting Algorithm

1. Find cluster \( c \) with most corners.
2. Select neighbor of \( c \) which most users agree with.
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:

- Orientation of Polygons
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:

- Orientation of Polygons
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:
- Orientation of Polygons
- Result can be arbitrarily bad . . .
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:

- Orientation of Polygons
- Result can be arbitrarily bad . . .
- . . . or not existent in the input.
The Voting Algorithm

1. Find cluster $c$ with most corners.
2. Select neighbor of $c$ which most users agree with.
3. Proceed until cycle is found.
4. Return that cycle.

Unfortunately, there are subtleties:
- Orientation of Polygons
- Result can be arbitrarily bad . . .
- . . . or not existent in the input.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$.
3. Then the edge $(c_1, c_2)$ has the weight $w = -\log(k/n)$.
4. Find cycle with minimum mean weight in the graph.
5. Return that cycle and translate it to a polygon.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let \( k \) be the count of edges between \( c_1 \) and \( c_2 \).
   Then the edge \((c_1, c_2)\) has the weight \( w = -\log\left(\frac{k}{n}\right) \).
3. Find cycle with minimum mean weight in the graph.
4. Return that cycle and translate it to a polygon.

No more orientation-dependent!
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.

![Graph Diagram]

- 8 Polygons
- 7 Polygons
- 5 Polygons
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let \( k \) be the count of edges between \( c_1 \) and \( c_2 \).
   Then the edge \((c_1, c_2)\) has the weight \( w = -\log\left(\frac{k}{n}\right) \).
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$. Then the edge $(c_1, c_2)$ has the weight $w = -\log\left(\frac{k}{n}\right)$. 

![Diagram showing the Minimum Mean Weight Cycle Algorithm](image)
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$. Then the edge $(c_1, c_2)$ has the weight $w = -\log\left(\frac{k}{n}\right)$.
3. Find cycle with minimum mean weight in the graph.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$. Then the edge $(c_1, c_2)$ has the weight $w = -\log\left(\frac{k}{n}\right)$.
3. Find cycle with minimum mean weight in the graph.
4. Return that cycle and translate it to a polygon.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$. Then the edge $(c_1, c_2)$ has the weight $w = -\log\left(\frac{k}{n}\right)$.
3. Find cycle with minimum mean weight in the graph.
4. Return that cycle and translate it to a polygon.
Minimum Mean Weight Cycle Algorithm

1. Consider input as directed, weighted graph.
2. Let $k$ be the count of edges between $c_1$ and $c_2$. Then the edge $(c_1, c_2)$ has the weight $w = -\log\left(\frac{k}{n}\right)$.
3. Find cycle with minimum mean weight in the graph.
4. Return that cycle and translate it to a polygon.
Choice Of Parameters
Choice Of Parameters

**Problem:** How to find a good min-eps for clustering?
Choice Of Parameters

**Problem:** How to find a good min-eps for clustering?

**Solution so far:** Trial and Error
Choice Of Parameters

**Problem**: How to find a good min-eps for clustering?

**Solution so far**: Trial and Error
Choice Of Parameters

**Problem:** How to find a good min-eps for clustering?

**Solution so far:** Trial and Error
Choice Of Parameters

**Problem:** How to find a good min-eps for clustering?

**Solution so far:** Trial and Error

**Hope:** Find min-eps automatically
From an Observation to a Heuristic
From an Observation to a Heuristic
From an Observation to a Heuristic

Number of Clusters

\( \varepsilon \cdot 10^{-4} \)
From an Observation to a Heuristic

- Calculate median.

![Graph showing the number of clusters versus ε with steps at 0.2, 0.4, 0.6, 0.8, 1.0, and 1.2, with ε on the x-axis and the number of clusters on the y-axis, with a step function indicating the decrease in the number of clusters as ε increases.]
From an Observation to a Heuristic

- Calculate median.
- Allow only clusters of size \( n/2 \).
From an Observation to a Heuristic

- Calculate median.
- Allow only clusters of size $n/2$.
- Select longest plateau.
From an Observation to a Heuristic

- Calculate median.
- Allow only clusters of size \( n/2 \).
- Select longest plateau.
- Use Semi-Dynamic-Sets.

![Graph showing number of clusters vs. parameter \( \varepsilon \)]

Number of Clusters

\( \cdot 10^{-4} \)
From an Observation to a Heuristic

- Calculate median.
- Allow only clusters of size $n/2$.
- Select longest plateau.
- Use Semi-Dynamic-Sets.
- Running time $O(n^2 \log n)$.

![Graph showing the number of clusters vs. parameter $\epsilon$.]
Evaluation Process

Semantics
Check if the polygons fit to footprints of buildings.

Accuracy
Extract the average brightness of the pixels under the polygon edges automatically. The lower the better!
Evaluation Process

Sources
Evaluation Process

Sources

detected,
Evaluation Process

Sources

detected, user,
Evaluation Process

Sources

- detected, user,
- mmwc-autoeps,
Evaluation Process

Sources

detected, user,
mmwc-autoeps, mmwc-raw,
Evaluation Process

Sources

detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
Evaluation Process

Sources

detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
voting-autoeps,
Evaluation Process

Sources

detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
voting-autoeps, voting-raw,
Evaluation Process

Sources

detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
投票-autoeps, voting-raw, voting-clean
Evaluation Process

**Sources**
detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
voting-autoeps, voting-raw, voting-clean

**Semantics**
Check if the polygons fit to footprints of buildings.
Evaluation Process

Sources
detected, user, mmwc-autoeps, mmwc-raw, mmwc-clean, voting-autoeps, voting-raw, voting-clean

Semantics
Check if the polygons fit to footprints of buildings.
Evaluation Process

Sources
detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
voting-autoeps, voting-raw, voting-clean

Semantics
Check if the polygons fit to footprints of buildings.

Accuracy
Extract the average brightness of the pixels *under* the polygon edges automatically. The lower the better!
Evaluation Process

Sources
detected, user, mmwc-autoeps, mmwc-raw, mmwc-clean, voting-autoeps, voting-raw, voting-clean

Semantics
Check if the polygons fit to footprints of buildings.

Accuracy
Extract the average brightness of the pixels under the polygon edges automatically. The lower the better!

Value: 0.48
Evaluation Process

Sources
detected, user,
mmwc-autoeps, mmwc-raw, mmwc-clean,
voting-autoeps, voting-raw, voting-clean

Semantics
Check if the polygons fit to footprints of buildings.

Accuracy
Extract the average brightness of the pixels under the polygon edges automatically. The lower the better!

Value: 0.38
Evaluation Process

Sources
detected, user, mmwc-autoeps, mmwc-raw, mmwc-clean, voting-autoeps, voting-raw, voting-clean

Semantics
Check if the polygons fit to footprints of buildings.

Accuracy
Extract the average brightness of the pixels under the polygon edges automatically. The lower the better! Value: 0.36
Sematical Results
Semantical Results

Yes  No

Percent Of Votes

detected  user  mmwc-autoeps  mmwc-raw  mmwc-clean  voting-autoeps  voting-raw  voting-clean

0  0.5  1
Semantical Results

MMWC was not better than Voting!

<table>
<thead>
<tr>
<th>Method</th>
<th>Percent Of Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>detected</td>
<td>1</td>
</tr>
<tr>
<td>user</td>
<td>1</td>
</tr>
<tr>
<td>mmwc-autoeps</td>
<td>1</td>
</tr>
<tr>
<td>mmwc-raw</td>
<td>1</td>
</tr>
<tr>
<td>mmwc-clean</td>
<td>1</td>
</tr>
<tr>
<td>voting-autoeps</td>
<td>1</td>
</tr>
<tr>
<td>voting-raw</td>
<td>1</td>
</tr>
<tr>
<td>voting-clean</td>
<td>1</td>
</tr>
</tbody>
</table>

Yes ■ No ■

Mmwc was not better than Voting!
Semantical Results

- Yes
- No

Removal of outliers does not affect Voting much.

Graph showing the percent of votes for different conditions:
- detected
- user
- mmwc-autoeps
- mmwc-raw
- mmwc-clean
- voting-autoeps
- voting-raw
- voting-clean
Semantical Results

What happened with the autoeps-variants?

Percent Of Votes

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>detected</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>user</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>mmwc-autoeps</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>mmwc-raw</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>mmwc-clean</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>voting-autoeps</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>voting-raw</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>voting-clean</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Analysis of the auto-eps Variants
Analysis of the auto-eps Variants

There were 45 groups where both autoeps-variants failed.
Analysis of the auto-eps Variants

There were 45 groups where both autoeps-variants failed.

Example:
Analysis of the auto-eps Variants

There were 45 groups where both autoeps-variants failed.

Example:

They got confused!
Analysis of the auto-eps Variants

There were 45 groups where both autoeps-variants failed. There were two groups only the autoeps-variants could solve.

Example:

They got confused!
Analysis of the auto-eps Variants

There were 45 groups where both autoeps-variants failed.

Example:

They got confused!

There were two groups only the autoeps-variants could solve.

Example:
Brightness Results
Brightness Results

![Brightness Results Graph]

- **Average Brightness**
  - user: 0.492
  - mmwc-autoeps: 0.432
  - mmwc-raw: 0.443
  - mmwc-clean: 0.442
  - voting-autoeps: 0.436
  - voting-raw: 0.443
  - voting-clean: 0.440

14 / 15
Algorithmically, we proposed three algorithms to solve the problem, using pure geometry or cluster-based approaches. Implications from the results: the cluster-based algorithms need two pre-processing steps. One of them can be spared without loss of quality. This choice can be made with regard to the data. The Voting Algorithm performs better than the Mwwc Algorithm.
Conclusion

Algorithmically
Conclusion

Algorithmically

We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.
Conclusion

Algorithmically
We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.

Implications from the Results
Conclusion

Algorithmically
We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.

Implications from the Results
The cluster-based algorithms need two pre-processing steps.
Conclusion

Algorithmically
We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.

Implications from the Results
The cluster-based algorithms need two pre-processing steps. One of them can be spared without loss of quality.
Conclusion

Algorithmically
We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.

Implications from the Results
The cluster-based algorithms need two pre-processing steps. One of them can be spared without loss of quality. This choice can be made with regard to the data.
Conclusion

Algorithmically
We proposed three Algorithms to solve the problem, using pure geometry or cluster-based approaches.

Implications from the Results
The cluster-based algorithms need two pre-processing steps. One of them can be spared without loss of quality. This choice can be made with regard to the data.

The Voting Algorithm performs better than the Mwwc Algorithm.