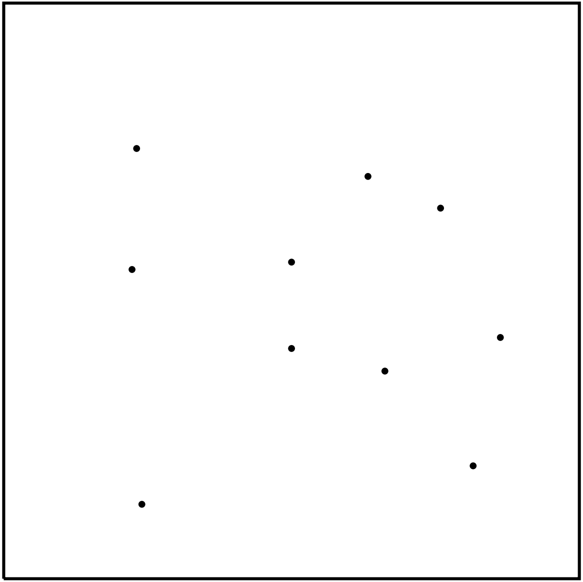
Modeling grains

Input: Spatially indexed individual orientations given as list of tuples $(\mathbf{x}_\ell, p_\ell, q_\ell)$, $\ell = 1, \ldots, n$ with

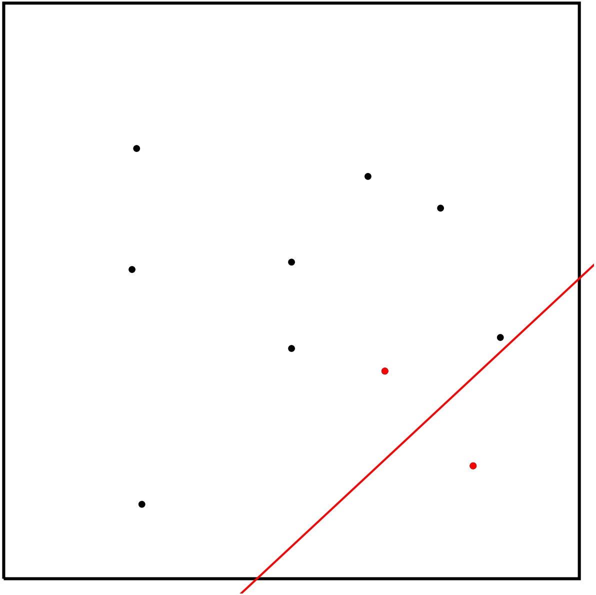
- locations $\mathbf{x}_\ell \in \mathfrak{D}$ bounded to some area $\mathfrak{D} \subset \mathbb{R}^d$,
- phase information $p_\ell \in \mathbb{N}^+$ (includes crystal symmetry),
- orientations $q_\ell \in \mathrm{SO}(3)$.

Output: A graph based data model representing grains and grain boundaries.

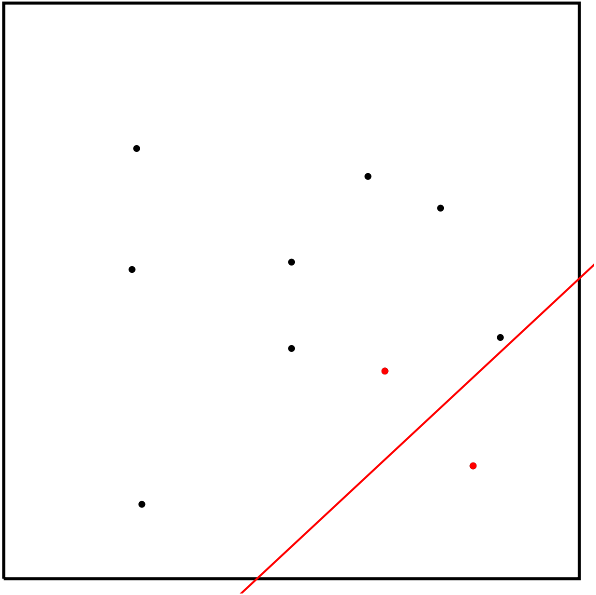
Modeling grains



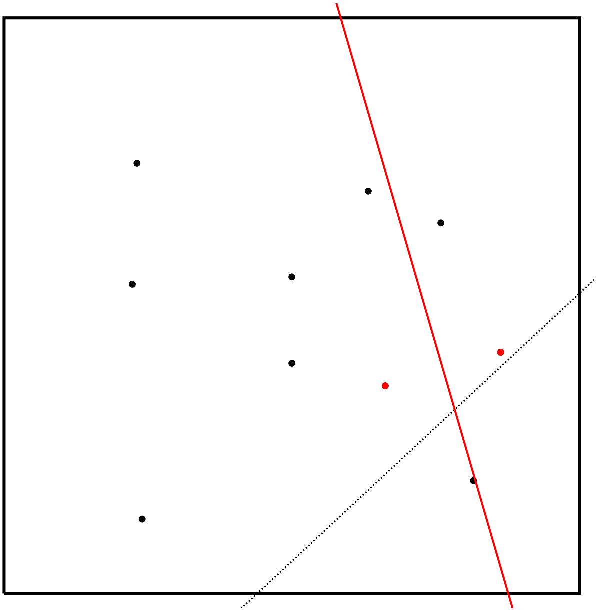
Modeling grains



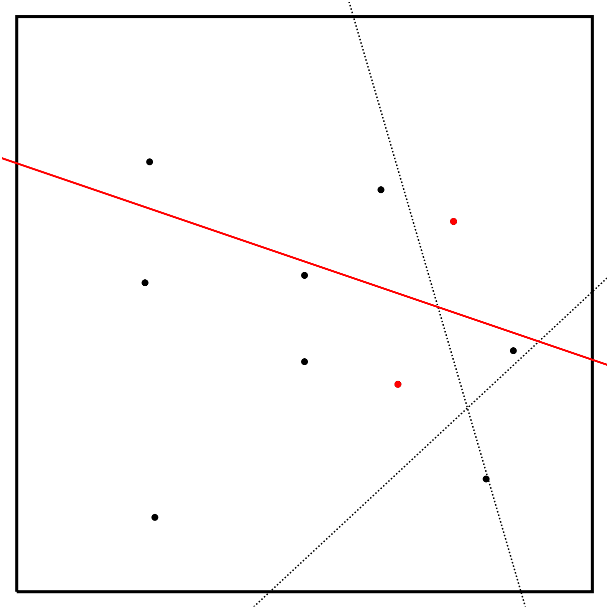
Modeling grains



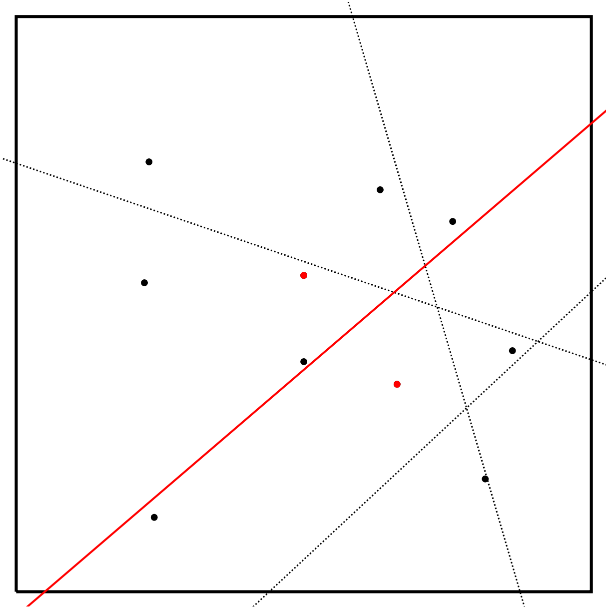
Modeling grains



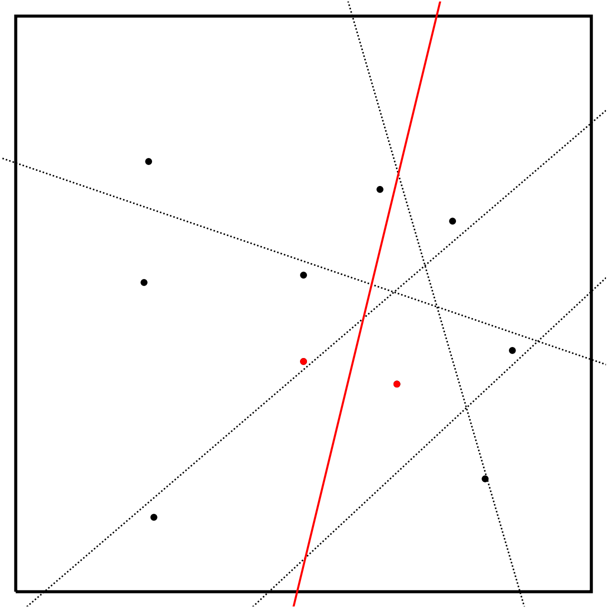
Modeling grains



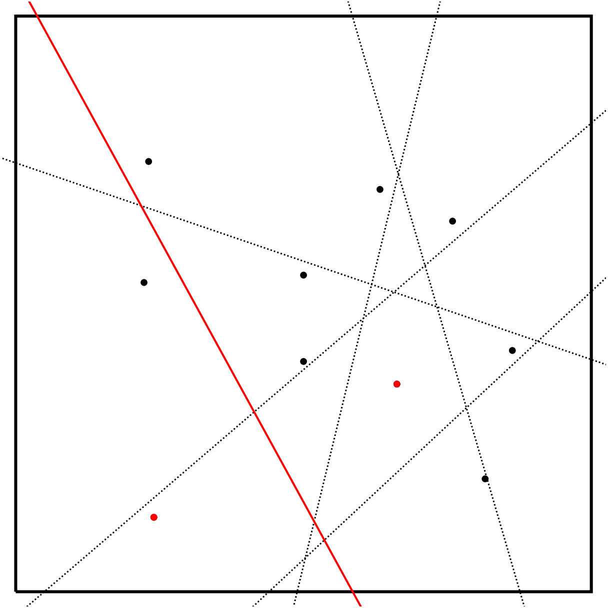
Modeling grains



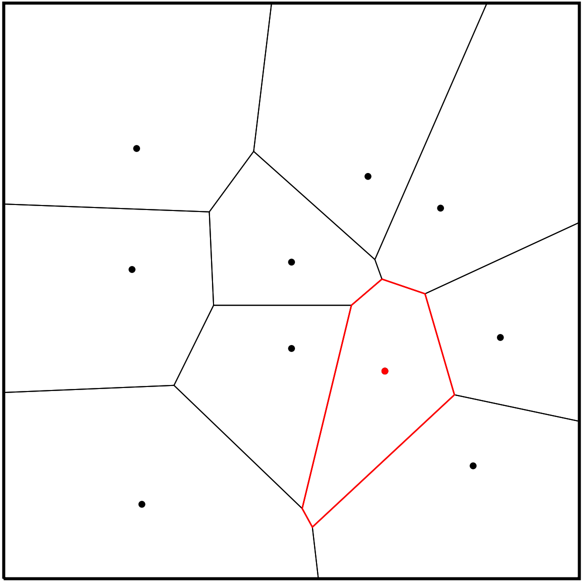
Modeling grains



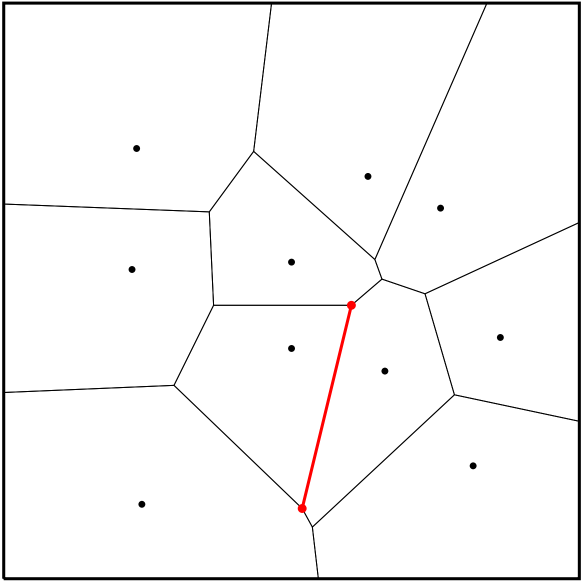
Modeling grains



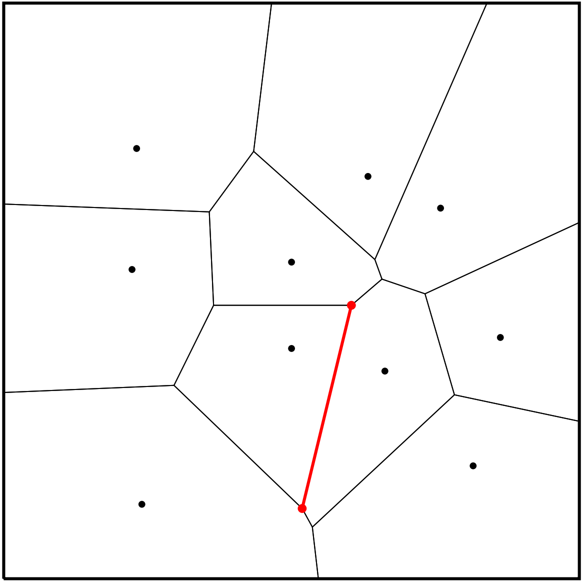
Modeling grains



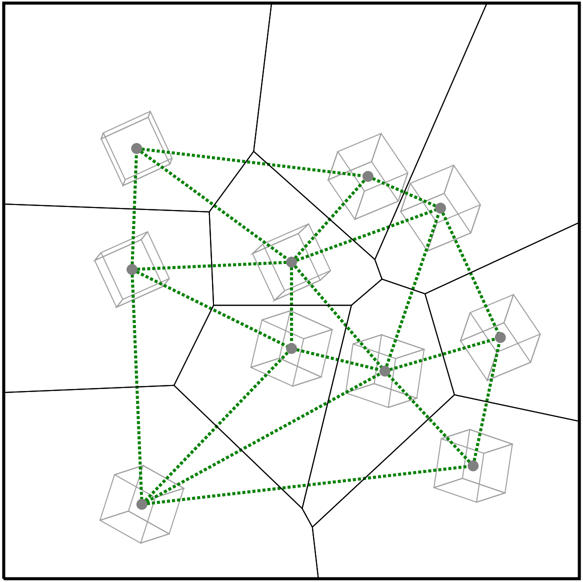
Modeling grains



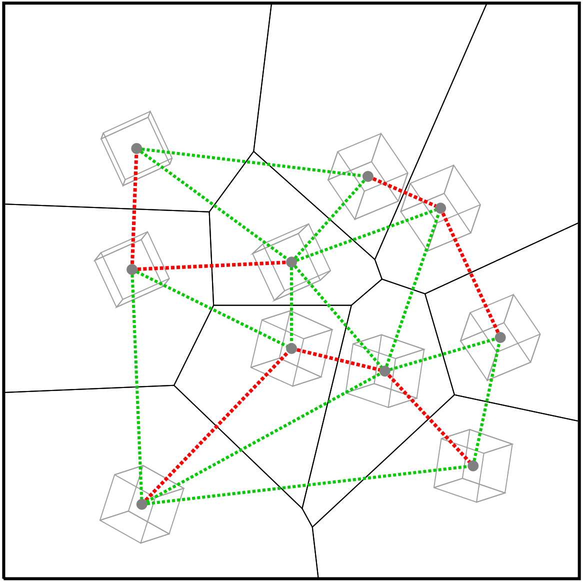
Modeling grains



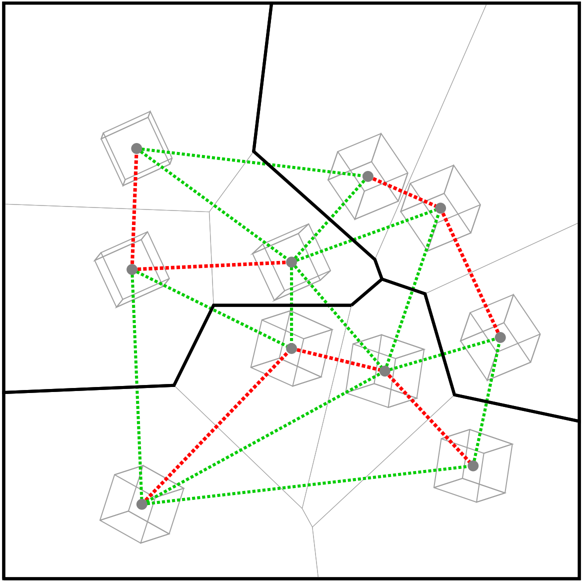
Modeling grains



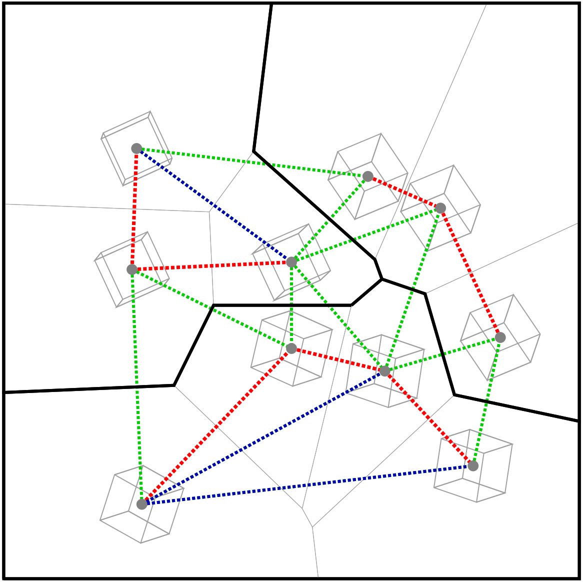
Modeling grains



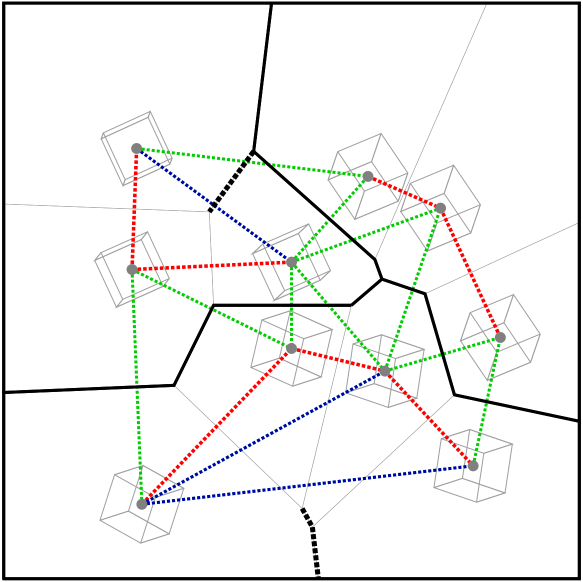
Modeling grains



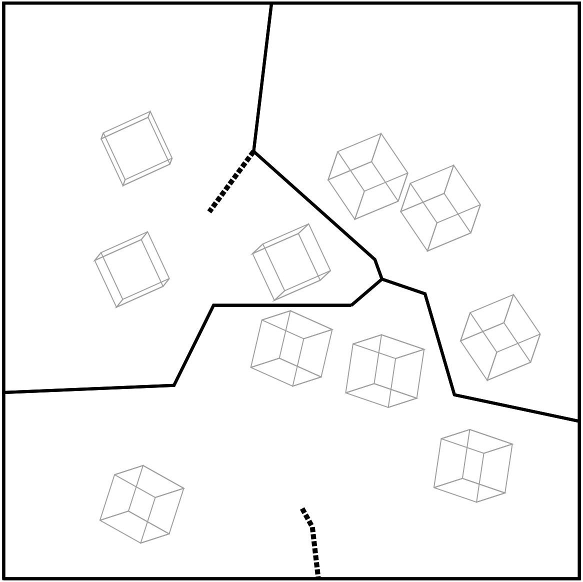
Modeling grains



Modeling grains



Modeling grains



Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial_{\mathrm{sub}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$

$$\left[\mathbf{I}_{VE}\right]^{i,j} = \begin{cases} 1, & \text{if vertex } v_i \text{ incident to edge } e_j, \\ 0, & \text{otherwise.} \end{cases}$$

2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.

$$\left[\mathbf{I}_{ED}\right]^{j,\ell} = \begin{cases} 1, & \text{if edge } e_j \text{ incident to Voronoi-cell } D(\mathbf{x}_\ell), \\ 0, & \text{otherwise.} \end{cases}$$

2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^\circ + \mathbf{A}_D^\partial$
3. Search components of $\mathbf{A}_D^\circ$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^\partial = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^\circ + \mathbf{I}_{EG}^\partial$, with $\mathbf{I}_{EG}^\partial = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\boxed{\mathbf{A}_D}$.

$$\left[\boxed{\mathbf{A}_D}\right]^{\ell,\ell'} = \begin{cases} 1, & \text{if Voronoi-cell } D(\mathbf{x}_\ell) \text{ adjacent to Voronoi-cell } D(\mathbf{x}_{\ell'}), \\ 0, & \text{otherwise.} \end{cases}$$

2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^\circ + \mathbf{A}_D^\partial$
3. Search components of $\mathbf{A}_D^\circ$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^\partial = \mathbf{A}_D^{\partial_{\mathrm{sub}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^\circ + \mathbf{I}_{EG}^\partial$, with $\mathbf{I}_{EG}^\partial = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \boxed{\mathbf{A}_D^{\circ}} + \mathbf{A}_D^{\partial}$,

$$\left[\boxed{\mathbf{A}_D^{\circ}}\right]^{\ell,\ell'} = \begin{cases} 1, & \text{if } D(\mathbf{x}_\ell) \text{ and } D(\mathbf{x}_{\ell'}) \text{ adjacent and no grain boundary}, \\ 0, & \text{otherwise.} \end{cases}$$

3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \boxed{\mathbf{A}_D^{\partial}}$,
$$\left[\boxed{\mathbf{A}_D^{\partial}}\right]^{\ell,\ell'} = \begin{cases} 1, & \text{if } D(\mathbf{x}_\ell) \text{ and } D(\mathbf{x}_{\ell'}) \text{ adjacent and grain boundary,} \\ 0, & \text{otherwise.} \end{cases}$$
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^\circ + \mathbf{A}_D^\partial$
3. Search components of $\mathbf{A}_D^\circ$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^\partial = \mathbf{A}_D^{\partial_{\mathrm{sub}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^\circ + \mathbf{I}_{EG}^\partial$, with $\mathbf{I}_{EG}^\partial = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\boxed{\mathbf{I}_{DG}} \subseteq D \times G$,

$$\left[\boxed{\mathbf{I}_{DG}}\right]^{\ell,m} = \begin{cases} 1, & \text{if } D(\mathbf{x}_\ell) \text{ is part of grain } g_m, \\ 0, & \text{otherwise.} \end{cases}$$

4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial_{\text{sub}}} + \mathbf{A}_D^{\partial_{\text{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial_{\text{sub}}} + \mathbf{I}_{EG}^{\partial_{\text{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\boxed{\mathbf{A}_G} \subseteq G \times G$,

$$\left[\boxed{\mathbf{A}_G}\right]^{m,m'} = \begin{cases} 1, & \text{if grain } g_m \text{ and } g_{m'} \text{ have a common grain boundary,} \\ 0, & \text{otherwise.} \end{cases}$$

5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^\circ + \mathbf{A}_D^\partial$.
3. Search components of $\mathbf{A}_D^\circ$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$,

$$[\mathbf{A}_G]^{m,m'} = \begin{cases} 1, & \text{if } \left[\mathbf{I}_{DG}^\top \mathbf{A}_D^\partial \mathbf{I}_{DG}\right]^{m,m'} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

5. Decompose adjacencies $\mathbf{A}_D^\partial = \mathbf{A}_D^{\partial \mathrm{sub}} + \mathbf{A}_D^{\partial \mathrm{ext}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^\circ + \mathbf{I}_{EG}^\partial$, with $\mathbf{I}_{EG}^\partial = \mathbf{I}_{EG}^{\partial \mathrm{sub}} + \mathbf{I}_{EG}^{\partial \mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$.
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial_{\mathrm{sub}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$.
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \boxed{\mathbf{A}_D^{\partial_{\mathrm{sub}}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$.,

$$\left[\boxed{\mathbf{A}_D^{\partial_{\mathrm{sub}}}}\right]^{\ell,\ell'} = \begin{cases} 1, & \text{if } D(\mathbf{x}_\ell) \text{ and } D(\mathbf{x}_{\ell'}) \text{ adjacent and sub grain boundary}, \\ 0, & \text{otherwise.} \end{cases}$$

6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$.
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \boxed{\mathbf{A}_D^{\partial\mathrm{ext}}}$,

$$\left[\boxed{\mathbf{A}_D^{\partial\mathrm{ext}}}\right]^{\ell,\ell'} = \begin{cases} 1, & \text{if } D(\mathbf{x}_\ell) \text{ and } D(\mathbf{x}_{\ell'}) \text{ adjacent and grain boundary}, \\ 0, & \text{otherwise.} \end{cases}$$

6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$.
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial\mathrm{sub}} + \mathbf{A}_D^{\partial\mathrm{ext}}$.
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial\mathrm{sub}} + \mathbf{I}_{EG}^{\partial\mathrm{ext}}$.

Modeling grains: algorithm

1. Voronoi-decomposition resulting in incidence matrix $\mathbf{I}_{VE}$, $\mathbf{I}_{ED}$ and adjacency matrix $\mathbf{A}_D$.
2. Decompose adjacencies $\mathbf{A}_D = \mathbf{A}_D^{\circ} + \mathbf{A}_D^{\partial}$.
3. Search components of $\mathbf{A}_D^{\circ}$ via depth-first search and determine incidence matrix $\mathbf{I}_{DG} \subseteq D \times G$.
4. Represent adjacencies of grains in a matrix $\mathbf{A}_G \subseteq G \times G$.
5. Decompose adjacencies $\mathbf{A}_D^{\partial} = \mathbf{A}_D^{\partial_{\mathrm{sub}}} + \mathbf{A}_D^{\partial_{\mathrm{ext}}}$
6. Compute incidence matrix $\mathbf{I}_{EG} = \mathbf{I}_{ED}\mathbf{I}_{DG}$ and decompose $\mathbf{I}_{EG} = \mathbf{I}_{EG}^{\circ} + \mathbf{I}_{EG}^{\partial}$, with $\mathbf{I}_{EG}^{\partial} = \mathbf{I}_{EG}^{\partial_{\mathrm{sub}}} + \mathbf{I}_{EG}^{\partial_{\mathrm{ext}}}$.

Output:
$\mathbf{A}_D^{\circ}, \mathbf{A}_D^{\partial_{\mathrm{sub}}}, \mathbf{A}_D^{\partial_{\mathrm{ext}}}, \mathbf{A}_G,$
$\mathbf{I}_{DG}, \mathbf{I}_{ED}^{\partial_{\mathrm{sub}}}, \mathbf{I}_{ED}^{\partial_{\mathrm{ext}}},$

Modeling grains: algorithm

Benefits ...

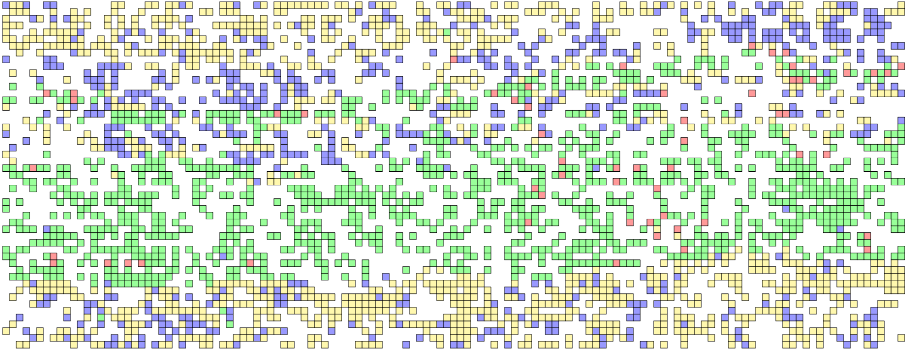
- Works in 2d and 3d
- Voronoi-decomposition flexible against not indexed data
- No interpolation of orientation
- Explicit data model ensures fast access to adjacencies and incidences.
- Almost every spatial relationship of EBSD data and its grains can be represented.

... Disadvantages

- Voronoi-decomposition is expensive.
- Large memory costs to store all adjacencies and incidences.

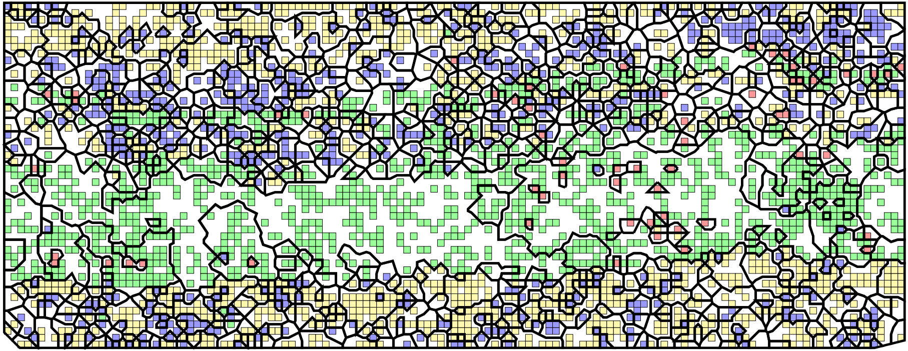
Modeling grains

```
mtexdata mylonite
plot(ebsd,'property','phase');
```



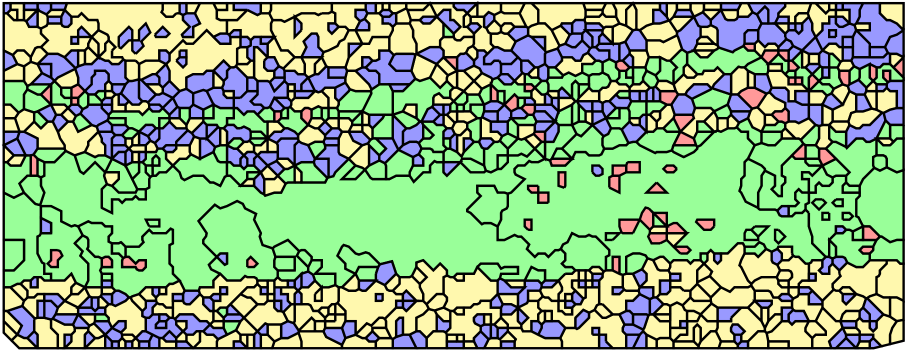
Modeling grains

```
grains = calcGrains(ebsd,'angle',5*degree);
hold on, plotBoundary(grains);
```



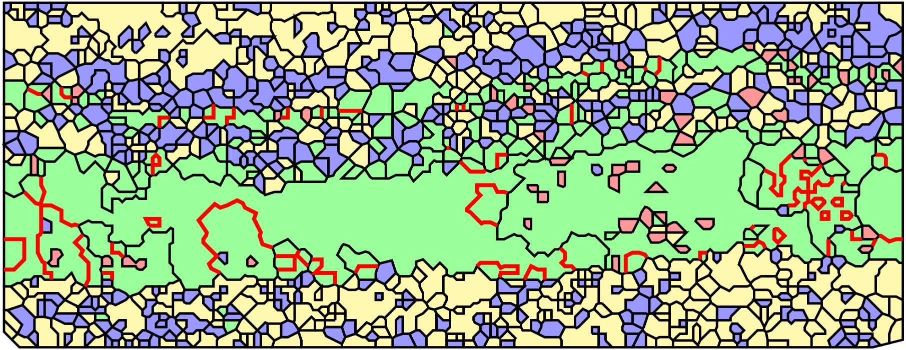
Modeling grains

```
plot(grains,'property','phase')
```



Modeling grains

```
hold on, plotBoundary(grains,'property',...
   orientation(...));
```



Properties of grains

Incidencies and Adjacencies

```
get(grains,'I_VF')
get(grains,'I_DG')
get(grains,'I_FG')
get(grains,'I_FDext')
get(grains,'I_FDint')
get(grains,'A_D')
get(grains,'A_G')
```

Properties

```
get(grains,'EBSD')
get(grains,'orientation')
get(grains,'orientations')
get(grains,'...')
```

Properties of grains

Geometric properties of grains

```
area(grains)
perimeter(grains)
diameter(grains)
centroid(grains)
shapefactor(grains)
aspectratio(grains)
equivalentradius(grains)
equivalentperimeter(grains)
principalcomponents(grains)
...
```

Other properties

```
grainSize(grains)
hasHole(grains)
isNotIndexed(grains)
```

Accessing grains

Accessing by phase

```
grains('mineral␣name')
grains({'fe','mg'})
```

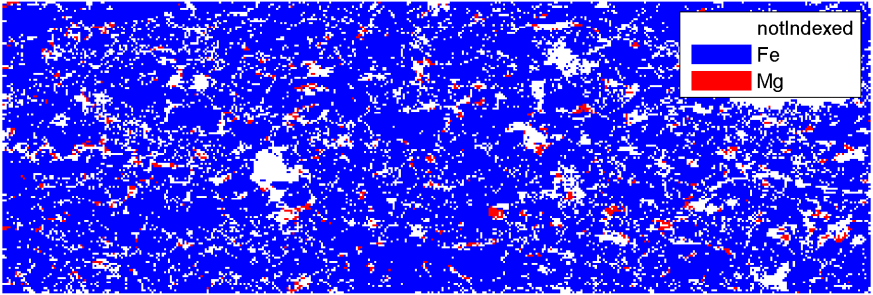
Accessing grains by indexing or logical expression

```
grains(1)
grains(1:10)
grains( area(grains) > 100 )
grains(diameter(grains)>10 | perimeter(grains)>10)
grains(diameter(grains)>10 & grains('fe'))
grains(grainSize(grains)>1 & ~grains('notindexed'))
```

Concatenation of grain

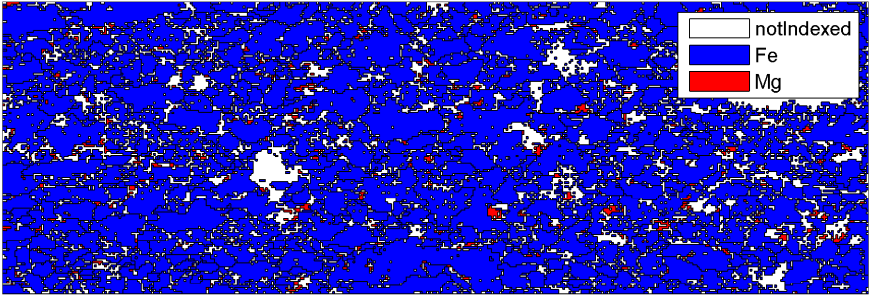
```
[grains('fe') grains('mg')]
```

Correcting EBSD and grains



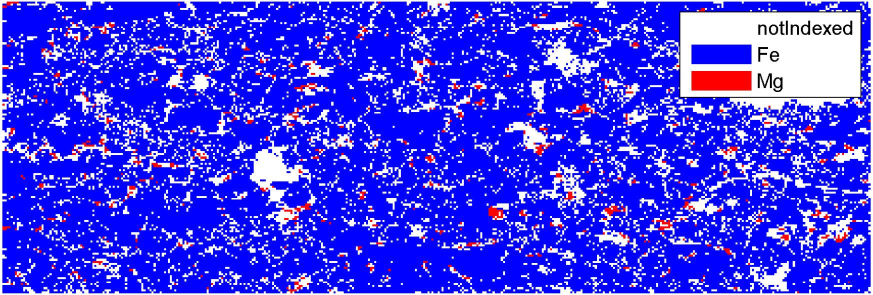
1. Import EBSD.
2. Reconstruct grains.

Correcting EBSD and grains



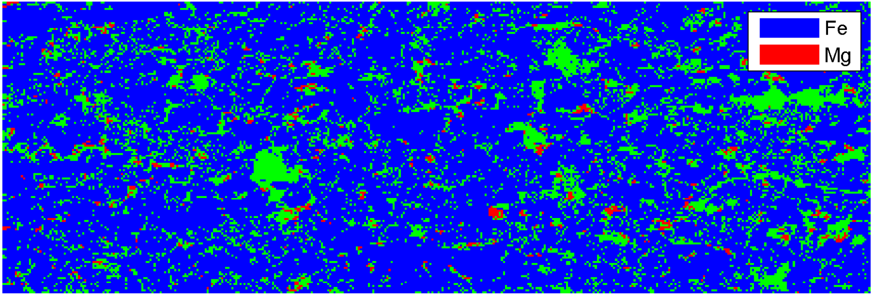
1. Import EBSD.
2. Reconstruct grains.

Correcting EBSD and grains



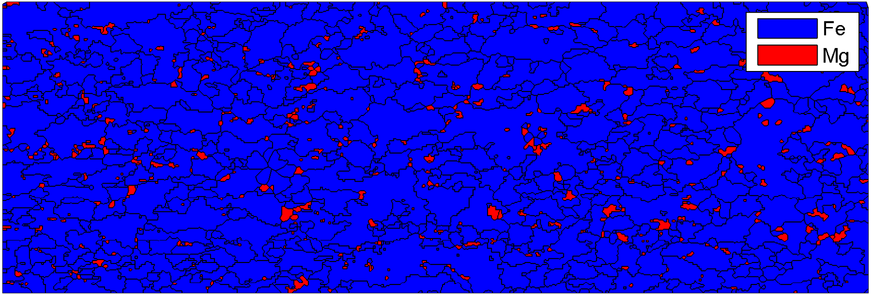
1. Import EBSD.
2. Remove badly indexed data and remove non-plausible grains, e.g. '1-pixel' grains.
3. Reconstruct grains.

Correcting EBSD and grains



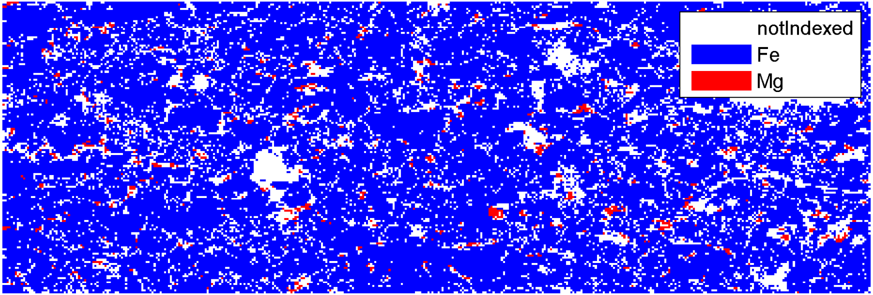
1. Import EBSD.
2. Remove badly indexed data and remove non-plausible grains, e.g. '1-pixel' grains.
3. Reconstruct grains.

Correcting EBSD and grains



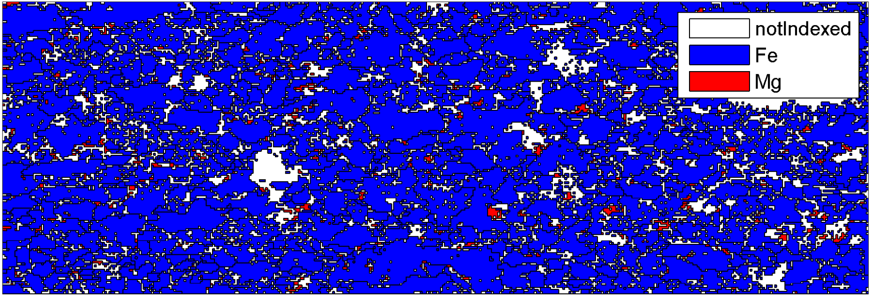
1. Import EBSD.
2. Remove badly indexed data and remove non-plausible grains, e.g. '1-pixel' grains.
3. Reconstruct grains.

Correcting EBSD and grains



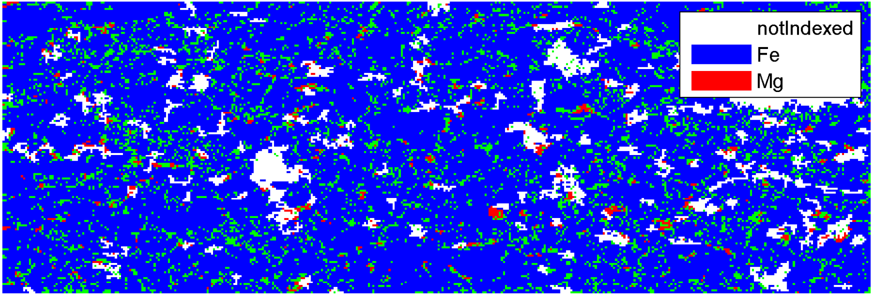
1. Import EBSD.
2. Reconstruct grains.
3. Remove non-plausible data but keep some dummy data.
4. Reconstruct grains again.

Correcting EBSD and grains



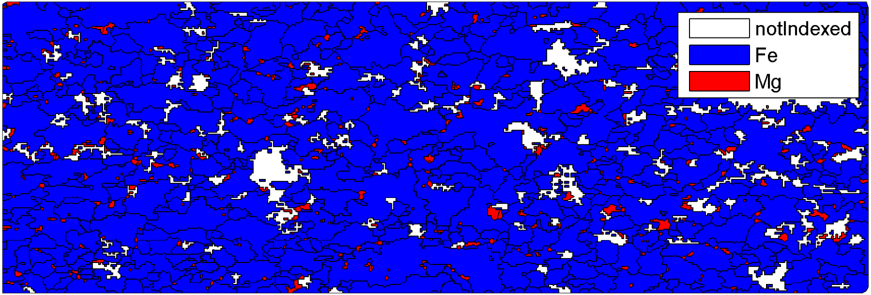
1. Import EBSD.
2. Reconstruct grains.
3. Remove non-plausible data but keep some dummy data.
4. Reconstruct grains again.

Correcting EBSD and grains



1. Import EBSD.
2. Reconstruct grains.
3. Remove non-plausible data but keep some dummy data.
4. Reconstruct grains again.

Correcting EBSD and grains

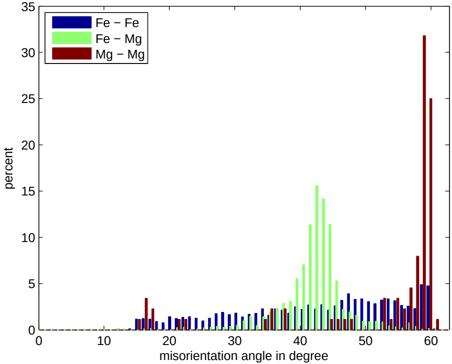


1. Import EBSD.
2. Reconstruct grains.
3. Remove non-plausible data but keep some dummy data.
4. Reconstruct grains again.

Grain boundaries and misorienation

via boundary misorientation angular distribution

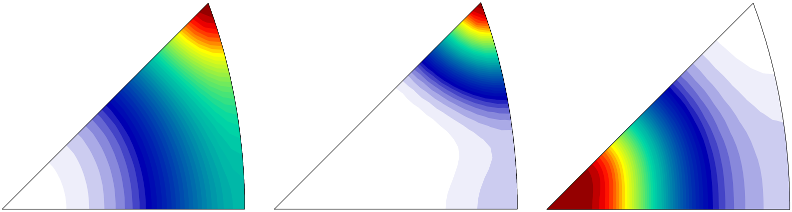
```
plotAngleDistribution(grains)
plotAngleDistribution(grains('fe'),63)
plotAngleDistribution(grains('fe'),grains('mg'),63)
```



Grain boundaries and misorienation

via boundary misorientation axis distribution

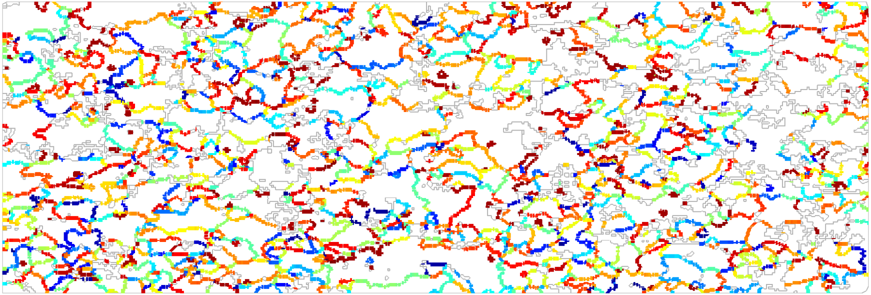
```
plotAxisDistribution(grains('fe'))
plotAxisDistribution(grains('mg'),...
                          'smooth','antipodal')
plotAxisDistribution(grains('mg'),grains('fe'))
```



Grain boundaries and misorienation

via spatial plots

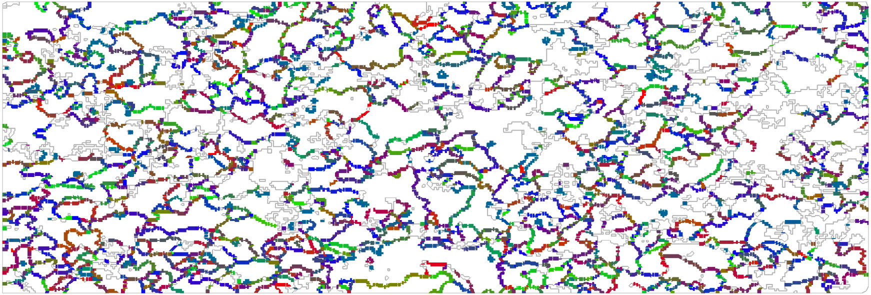
```
plotBoundary(grains,'property','angle')
plotBoundary(grains,'property','misorientation')
plotBoundary(grains,'property',...
   orientation(...),'delta',2*degree)
plotBoundary(grains,'property',[2 10]*degree)
```



Grain boundaries and misorienation

via spatial plots

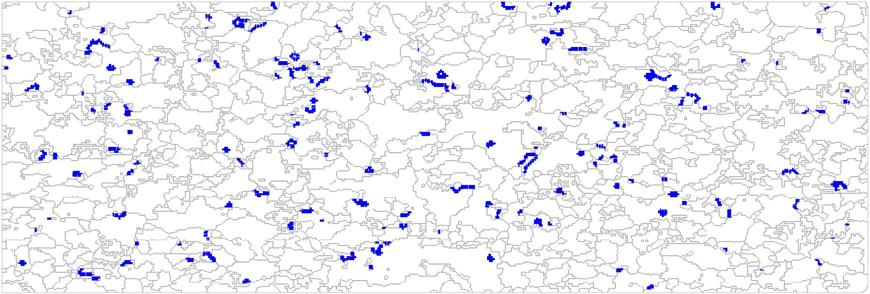
```
plotBoundary(grains,'property','angle')
plotBoundary(grains,'property','misorientation')
plotBoundary(grains,'property',...
   orientation(...),'delta',2*degree)
plotBoundary(grains,'property',[2 10]*degree)
```



Grain boundaries and misorienation

via spatial plots

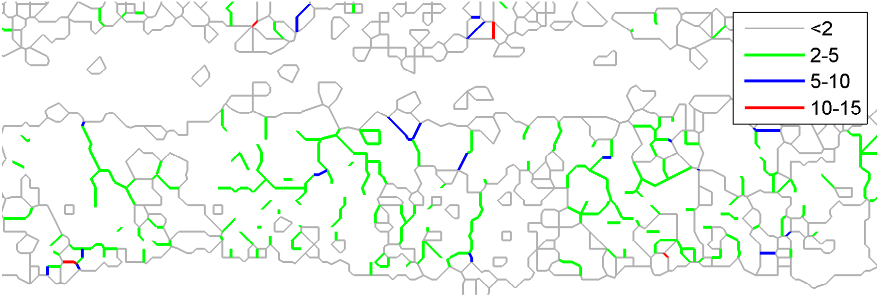
```
plotBoundary(grains,'property','angle')
plotBoundary(grains,'property','misorientation')
plotBoundary(grains,'property',...
   orientation(...),'delta',2*degree)
plotBoundary(grains,'property',[2 10]*degree)
```



Grain boundaries and misorienation

via spatial plots

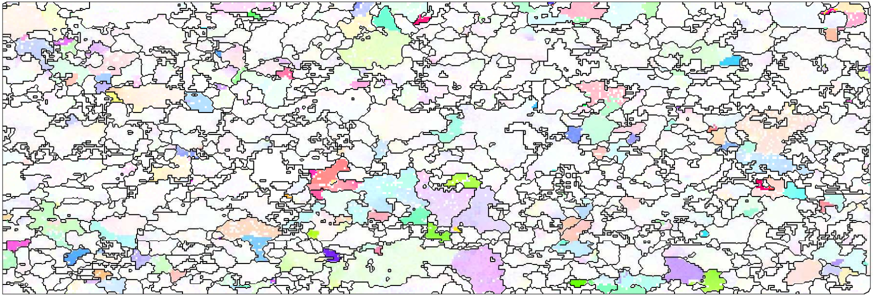
```
plotBoundary(grains,'property','angle')
plotBoundary(grains,'property','misorientation')
plotBoundary(grains,'property',...
   orientation(...),'delta',2*degree)
plotBoundary(grains,'property',[2 10]*degree)
```



Grains and internal misorienation

Observe misorientation within a grain

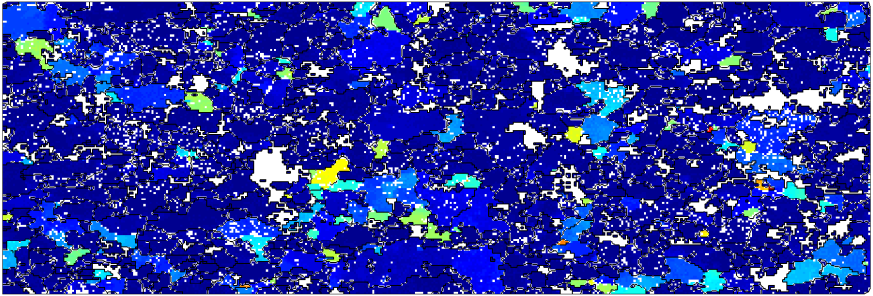
```
plot(grains,'property','mis2mean')
plot(grains,'property','mis2mean',...
                    'colorcoding','angle')
```



Grains and internal misorienation

Observe misorientation within a grain

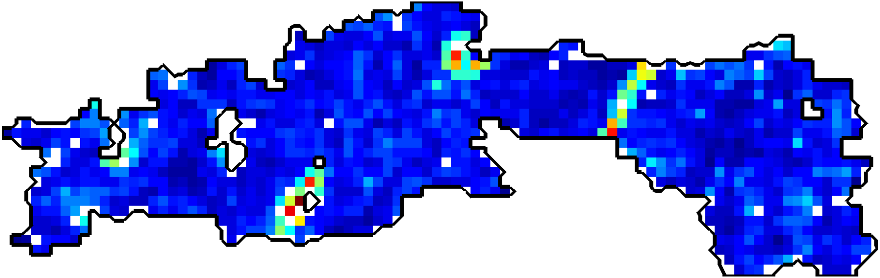
```
plot(grains,'property','mis2mean')
plot(grains,'property','mis2mean',...
                    'colorcoding','angle')
```



Grains and internal misorienation

Observe misorientation of grains individually

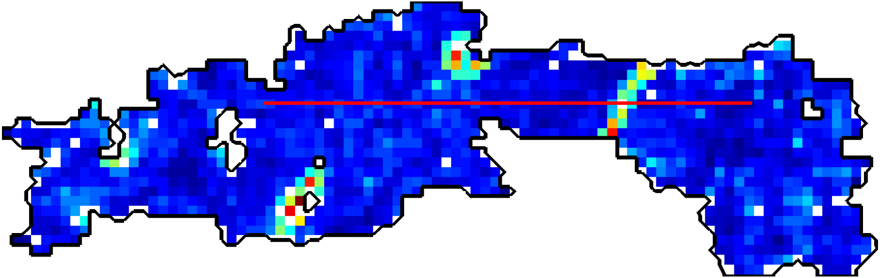
```
singlegrain = findByLocation(grains,[x y])
plotKAM(singlegrain)
spatialProfile(singlegrain,[x1 y1; x2 y2])
```



Grains and internal misorienation

Observe misorientation of grains individually

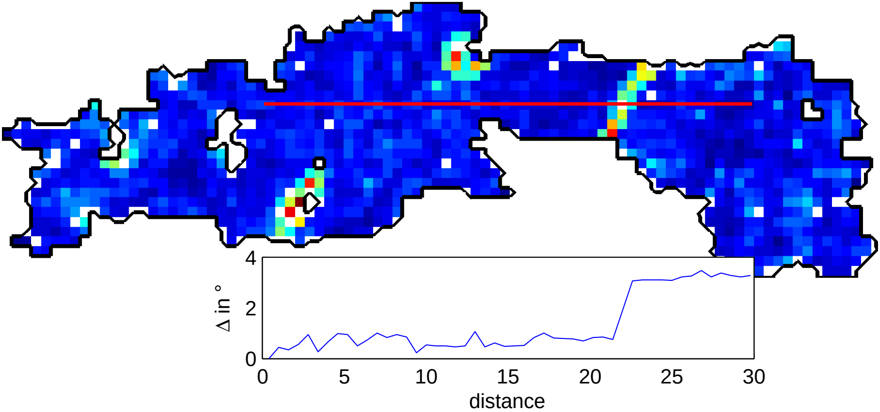
```
singlegrain = findByLocation(grains,[x y])
plotKAM(singlegrain)
spatialProfile(singlegrain,[x1 y1; x2 y2])
```



Grains and internal misorienation

Observe misorientation of grains individually

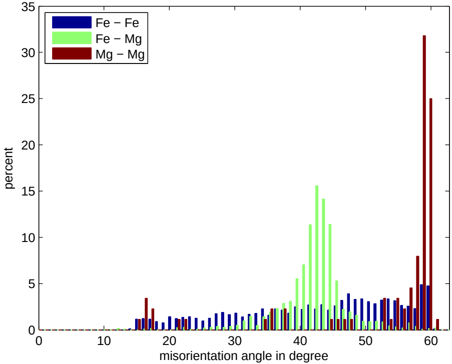
```
singlegrain = findByLocation(grains,[x y])
plotKAM(singlegrain)
spatialProfile(singlegrain,[x1 y1; x2 y2])
```



Grains and misorienation

via boundary misorientation angular distribution

```
plotAngleDistribution(grains)
plotAngleDistribution(grains('fe'),63)
plotAngleDistribution(grains('fe'),grains('mg'),63)
```



Organizing grains hierarchically

merging grains with certain misorientation angle

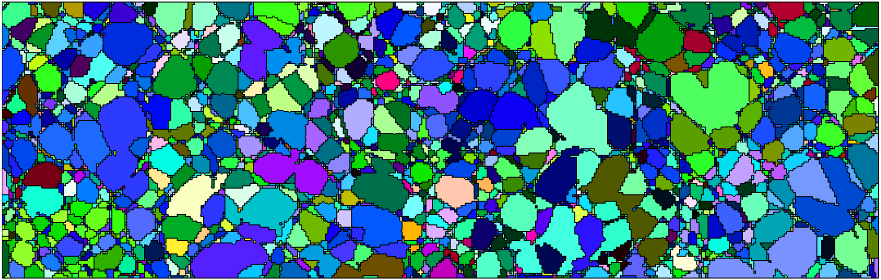
```
[grains5  I_5]  = merge(grains,5*degree);
[grains10 I_10] = merge(grains5,10*degree);

sum(I_5,2); sum(I_10,2);
sum(I_10*I_5,2);
```

Organizing grains hierarchically

merging grains with special boundary

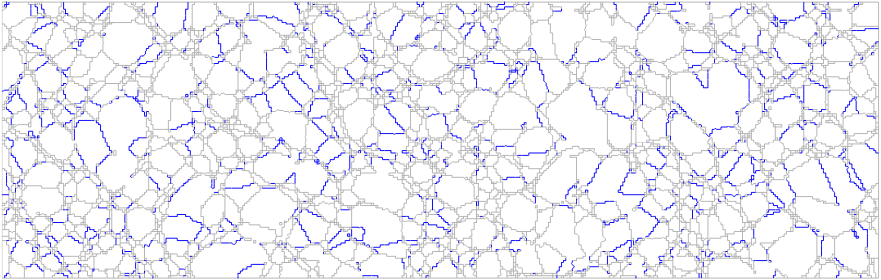
```
[grains_csl  I_csl] = merge(grains,CSL(3));
[grains_o    I_o  ] = merge(grains,...
                 orientation(...),'delta',2*degree);
```



Organizing grains hierarchically

merging grains with special boundary

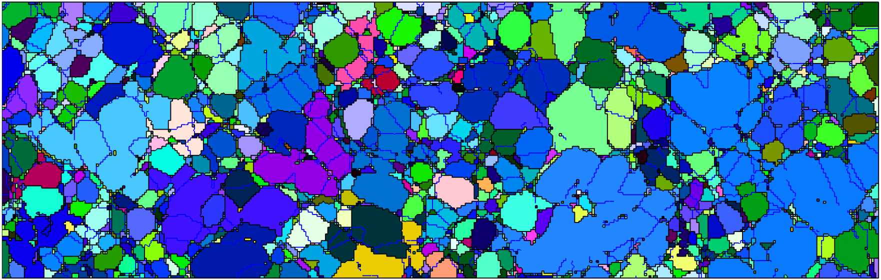
```
[grains_csl I_csl] = merge(grains,CSL(3));
[grains_o   I_o  ] = merge(grains,...
                      orientation(...),'delta',2*degree);
```



Organizing grains hierarchically

merging grains with special boundary

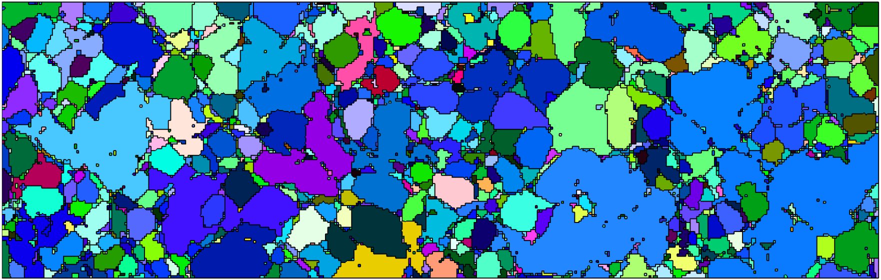
```
[grains_csl  I_csl] = merge(grains,CSL(3));
[grains_o    I_o  ] = merge(grains,...
                orientation(...),'delta',2*degree);
```



Organizing grains hierarchically

merging grains with special boundary

```
[grains_csl  I_csl] = merge(grains,CSL(3));
[grains_o    I_o  ] = merge(grains,...
              orientation(...),'delta',2*degree);
```



Exercises

Examine the EBSD data `mtexdata aachen`:

- Correct the EBSD data / grains.
- How does the correction influence the data analysis?

Examine the EBSD data `mtexdata mylonite`:

- Characterize special grain boundaries for all phases.
- Visualize your results.

Examine the EBSD data `ebsd_mergeCSL3.txt`:

- Which texture is present?
- Characterize special grain boundaries.
- Merge grains and investigate merged regions.
- What is wrong with the data?