# Description of MTEX twin analysis code

## Introduction

MTEX has become a powerful, widely used texture microstructure characterization tool with the latest releases having over 4000 downloads from github. In addition to its extensive functionality and documentation, MTEX has an advantage over many other codes for microstructure characterization in that its code is ready to use in Matlab without special environment setup or compiling, and it is easily understood and debugged in the Matlab environment. Many students are already familiar with Matlab due to extensive use in Academic and research institutions. In parallel to the rise of MTEX there have been significant advances in automated twin characterization presented in recent papers such as (Pradalier, Juan et al. 2018) and (Marshall, Proust et al. 2010). Incorporating the functionality described in these papers into the MTEX framework could be major boon to researchers who have already invested in learning MTEX and would provide a platform to quickly prototype advances in twin analysis. The final motivation for developing a twin analysis code in MTEX is that the code will benefit from community support – i.e. in finding bugs, community explanation on using the code, and extending functionality. The community framework will likely result in the largest impact on the microstructure characterization field.

Two distinct problems are solved in the twin analysis code: 1) Reconstruction of the original grains from twin and parent fragments and 2) classification of fragments by reconstruction of a twin graph. These two outputs allow for quantitative statistical analysis of twin behavior and the study of analysis of twin formation and propagation from EBSD datasets.

## Cleaning and grain fragment reconstruction

Segmentation of EBSD should be performed in MTEX using the smallest angle that does not result in significant subgrain artifacts. This choice stems from the observation that it relatively simple to handle over segmentation compared to under segmentation in downstream analysis. Under segmentation is particularly a problem for sharply textured samples where it is statistically probably to have initial grains with low misorientation (less than 5 degrees). Grain cleanup is also an important part of grain reconstruction. For microstructures where it is unlikely to have multiple generations of twins or completely twinned grains, it is preferable to cleanup over segmentation artifacts and small fragments since this will simplify the classification of fragments and improve the speed of the code; however, if some grains in the microstructure are potentially fully twinned or have more than one generation of twinning, the grain cleanup should be limited to minimize the chance of removing an important twin relationship and compromising the classification procedure. One also has to consider what type of analysis will be performed after reconstruction and classification. For instance, EBSD resolution artifacts can cause a twin to appear as several grain fragments. If the goal is to analyze the variants or shape of the twins, the removal of some of the twin fragments might compromise the analysis. The cleaned EBSD and reconstructed grain file should be performed separate from the twin code and saved as .mat files to prevent repeating the cleaning procedure (wasting time), but also to fix the cleanup data for downstream analysis.

## Reconstruction of the original grains

### Clustering approach

Historically the collection of fragments that came from a single grain (grain cluster) have been reconstructed solely based on the mean orientation of a fragment or the boundary misorientation. If two grains are misoriented within some tolerance of a twin misorientation relationship, then the fragments are merged and classified as having the relationship of the twin relationship that merged them. In this work the classification and merging procedures are partially decoupled.

Fragment merging is performed using boundary misorientation while classification is performed using the boundary misorientation in conjunction with a loose tolerance and the mean orientation. There are several motivations for these choices: 1) the boundary misorientation is less sensitive to large misorientation in grains and is better than the misorientation of mean fragment at correctly clustering fragments. 2) If a loose tolerance is used for clustering in general, false boundaries will be merged and if a tight tolerance is used then twins sometimes are not identified. The severity of this will depend on the sharpness of the initial texture, level of deformation, and the number of twin types in grains. Ideally, the identification of twin type and clustering is decoupled. For full control, the clustering tolerances are specified per twin type for the boundary and a single mean orientation misorientation tolerance is used in conjunction with the boundary identified twins to assign twin type. 3) Fragments that are not merged by the boundary misorientation can still be merged automatically if they share a high fraction their boundary with a cluster or are fully internal to a cluster, if they have a similar orientation to other fragment in the cluster, or if they are added to a cluster manually by the user. This feature is especially useful in combination with clustering tolerances for rare twin types which are not essential in the clustering case procedure, but have a negative impact on clustering and for highly twinned microstructures. 4) A secondary, relaxed tolerance can be used to identify the twin relationship between mean orientations and families. While, the strongest relationship is between fragments that share boundary, for highly twinned microstructures it often happens that a twin no longer shares boundary with its parent, but only with a fellow twin. To address this case, the twin type is identified based on the family (fragments with the same orientation) of orientations in the cluster using a relaxed tolerance and care is taken to keep track of whether a relationship is defined for fragments touching or not. This information is later used during classification. 5) Since the clustering is relatively independent of the classification, looser tolerances can be used with impunity; the only issue being during the clustering of the fragment that matches a family in a cluster, false identification of twin type for missing twin types, or for twins that have similar misorientation e.g. double twins.

As mentioned, the misorientation between mean grains is used for classification while the boundary misorientation is used to create a starting cluster graph. In cases where an edge relationship is added manually, a relaxed boundary misorientation tolerance is utilized to determine if the edge a twin. If it is not, then an unknown twin type is assigned to the edge. It is important to realize that the cluster graph can be made up purposefully of non-twin relationships that correctly group grains. This is desirable for instance when the twin/parent relationships is missing and when performing operations such as merging two clusters together. In summary there is a tight tolerance specified for boundary twin misorientation that is specified per twin mode. A relaxed boundary twin misorientation is specified for finding twin relationships for edges added by the user. A relaxed mean grain twin misorientation is utilized during classification.

### Clustering data structure

To handle the combined boundary and mean orientation approach to clustering and fragment relationship type, it is important to adopt a data structure that can represent clusters, relationships between fragments, and relate this information to the reconstructed grains from MTEX. This is achieved by implementing the graph mentality presented in [Cap ref] using Matlab’s graph toolbox. In the fragment graph each grain fragment id is a node and edges relate fragment sharing boundary (). Thus, the grain clustering procedure is simplified to removing edge from the fragment graph, producing a sub graph which we call the cluster graph (). The cluster graph has the same nodes as the fragment graph and as a result, any nodal quantities defined down-stream such generation and twin type can be related to the MTEX grains and MTEX procedures such as merging grains by twin boundary can be performed instead by constructing a boundary list based on edge relationships. Thus, the plotting capabilities of MTEX for grains and merged grains intact, allowing data visualization.

### User interaction with data structure

## Family determination.

Once a cluster has been defined, sorting the various twin relationships is the primary task. Relationships are assumed to be nonlocal (i.e. a twins do not have to share a boundary for a relationship to exist) however local boundary does play a role during the voting procedure for the family graph. In [ref Rods and Cap] the algorithms presented are useful when there is a clear parent twin and the complexity of the circular relationships are limited. Here we utilize some of the many graph based tools in Matlab to develop a general procedure for sorting family relationships based on texture, spatial distribution, shared boundary, and centrality.

Algorithm summary

1. The starting graph is constructed from family twin relationships where each node represents a family and each edge is a twin relationship between the family. If a node is included in the cluster that does not have a twin relationship, the node is included in the family graph without an edge
2. Naturally, with large twin fractions and a predominately Schmid based voting, secondary and tertiary twinning will often create an acyclic graph with multiple family roots. No attempt is made to determine if the clustering is reasonable and it is left to the user to determine based on generation or what have you. To deal with these complicated graphs, the root of the Family tree is determined using the following procedure:
   1. The initial family graph of a cluster is constructed based on a relaxed mean misorientation tolerance and twin type definitions. The direction of relationships are constructed based on the Schmid, area, boundary ratio, and initial texture votes.
   2. The degree out and in of each node is evaluated. If only one family has degree in of 0, it is considered the root of the family tree. If more than one parent exists or a circle relationship results in no parents being available, the outcloseness centrality metric is used to determine how much of the graph can be access by a single node. The node with the highest value best satisfies the voting criterion is used to establish the edges. Family shared boundary segment number is utilized as a weight when computing the outcloseness to ensure that when several families traverse the graph equally well, the one that best describes the available twin boundaries should be chosen.
3. To describe the twin relationships, a minimum spanning tree must be constructed. To do this, the edge lengths are defined initially using the inverse of shared boundary segment. The minimum spanning tree follows Kruskal’s algorithm which finds the subset of edges that form a tree including every vertex while minimizing the additive weight from each edge included in the tree. By using the inverse of the shared boundary between families, the algorithm prioritizes families that share grain boundary. Given a minimum spanning tree and the root node, the directional graph representing the relations can be constructed and the voting method of [Rod] can be utilized to change the edge weights.

* The edge

1. Since the spanning tree is unaware of the edge direction, a binary operation is performed to increase The mimumum spanning tree can then be recalculated and the process repeated until a single graph emerges. The benefit of solving the minimum spanning tree iteratively is that only a Family tree that satisfies the root and all relationships is considered, and the approach is robust against edge relationships that are reversed.