# Identifying Pockets in STEP models

Chaman Singh Verma, Morad Behandish, Sai Netaluri

July 31, 2017

For the last four decades‘ CAD systems have been continuously evolving. Despite intuitive interfaces for interactive modifications, most of the CAD systems do not provide high-order features classification. The objective of this work is to identify simple *Pockets* from a STEP file. (an example shown in Figure below).

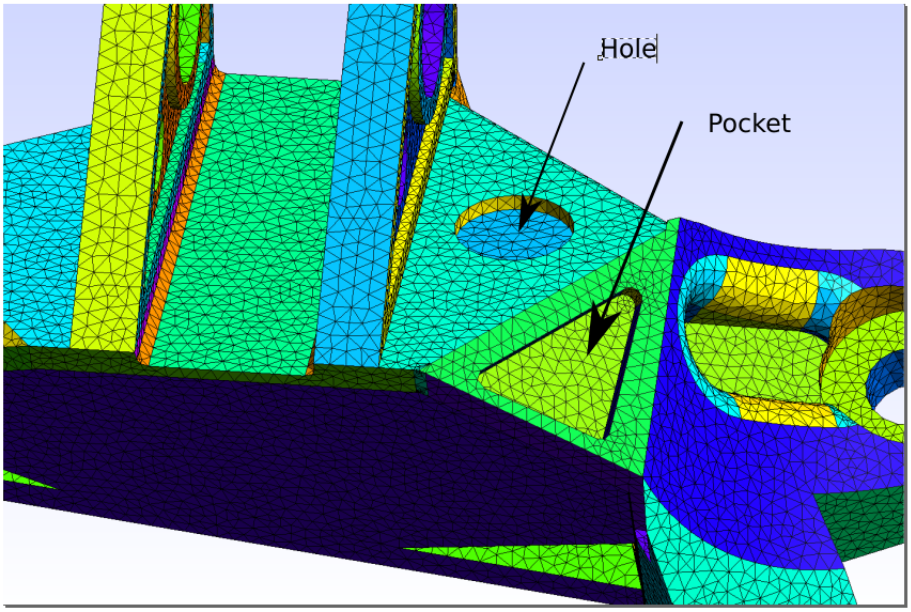


Figure 1: Pocket and hole in a CAD model

## 1 Pocket

A simple *pocket* has the following characteristics

- The base of a pocket is planar and it consists of a “‘non-circular boundary.
- The sides of a pocket are perpendicular to the base.
- The top of pocket consists of non-circul‘ar boundaries. Surfaces on the top are parallel to the base surface.
- A pocket should be clearly visible from outside i.e. no part of the pocket should be hidden by any surface above it.
- In the direction of normals, the base surface is below the top surfaces.

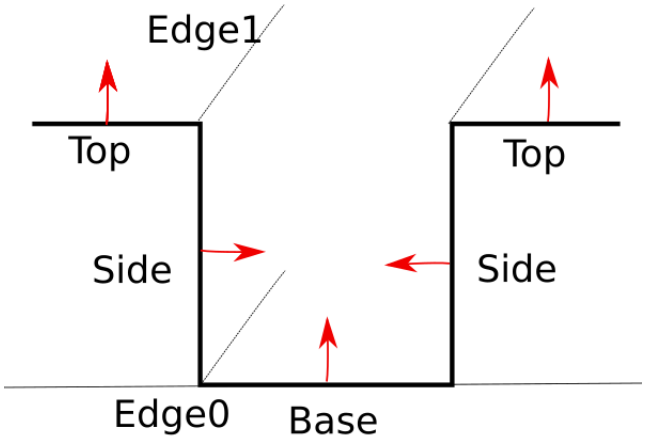


Figure 2: Defining various entities of a pocket

## 2 Identifying Pocket

We identifying pockets using *Breadth First Search (BFs)* of a topological graph of a CAD model. The topological graph of a CAD model is constructed directly from a STEP file using OPENCASCADE engine. A node in this graph represents a face and edges are constructed from adjacent nodes. All the planar surfaces are marked and they become seed nodes from where pocket searching start.

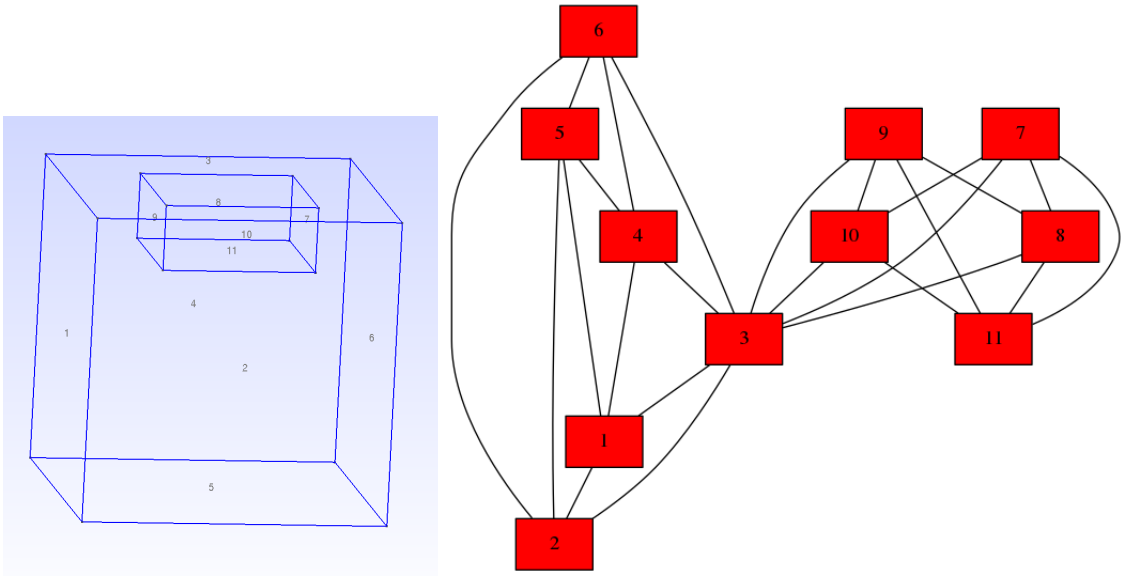


Figure 3: Defining various entities of a pocket

### 2.1 Pseudo code

The basic steps in the proposed algorithms are:

- 1: Construct topological graph of the STEP model and identify all the planar surfaces.
- 2. For each planar surface:

  - 3. Identify the boundary edges.
  - 4. If the boundaries make a circular shape then goto step 2.
  - 5. Transform the model so that that planar surface at y = 0
  - 6. Using the planar surface as a seed, continue searching through the graph if the angle between a surface and the base surface is 90 degree.

- Collect all the side surfaces and the base surface.

# 3 Walk through an example

We will walk through a simple example as shown in Figure [**?**] to explain how the code works.

We first construct a topological graph from the given step file. Since this model has 11 faces, therefore, the graph has 11 nodes. An edge in this graph is created when two faces are adjacent to each other. For example, the face 5, is adjacent to face 1, 2,4,and 6.

Since the face 5 is planar and its boundaries make a non-circular profile. It is valid seed to start. After transforming the model such that the outer normal pointing towards the +y axis, the entire object will lie in -y axis. Therefore 5 can be discarded for further calculations. Similar conditions will happen for face 1,2,3,4,6.

Next let us check for face 7. There will be missing face in the +y direction( because of empty space on the top), therefore it can be discared. Similar conditions will happen for face 9,8, and 10.

Next let us start face 11. Its boundary in non-circular and each boundary edge (i.e. faces 7,8,9,10) has a face in the +y direction. These faces make 90 angle with the face 8. Face of these faces are attached to the face 3 which is parallel to the face 3. Therefore, face 3 along with faces 7,8,9,10 make a pocket.

# 4 Result

The first implementation of the above algorithm is not robust but it can identify simple pockets correctly. It correctly discards pockets with circular profiles. However, it also identifies features which are not considered as pocket as shown in Figure below.

# 5 Limitations

The proposed algorithm identifies a pocket using local analysis. However, for machining purpose, some pockets may not be reachable by standard tools, therefore, they may not be listed as pocket. Such an example is shown below.

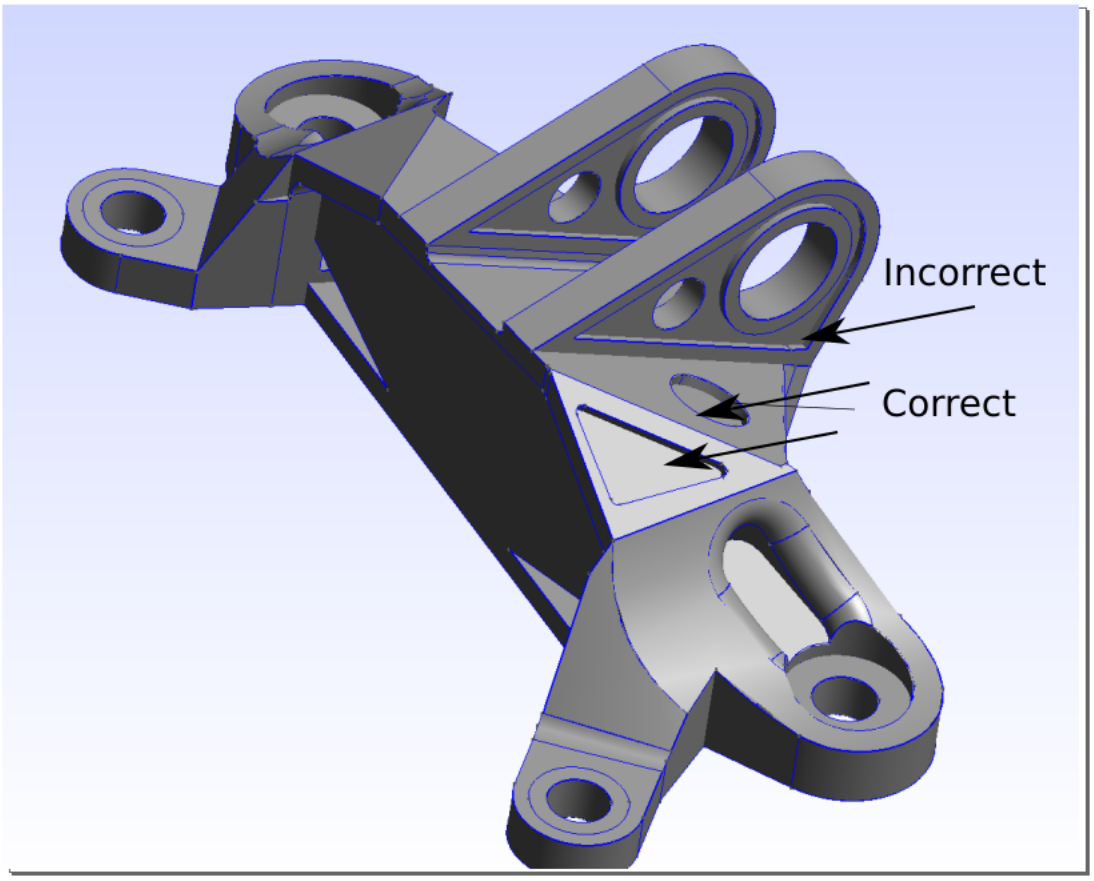


Figure 4: Defining various entities of a pocket

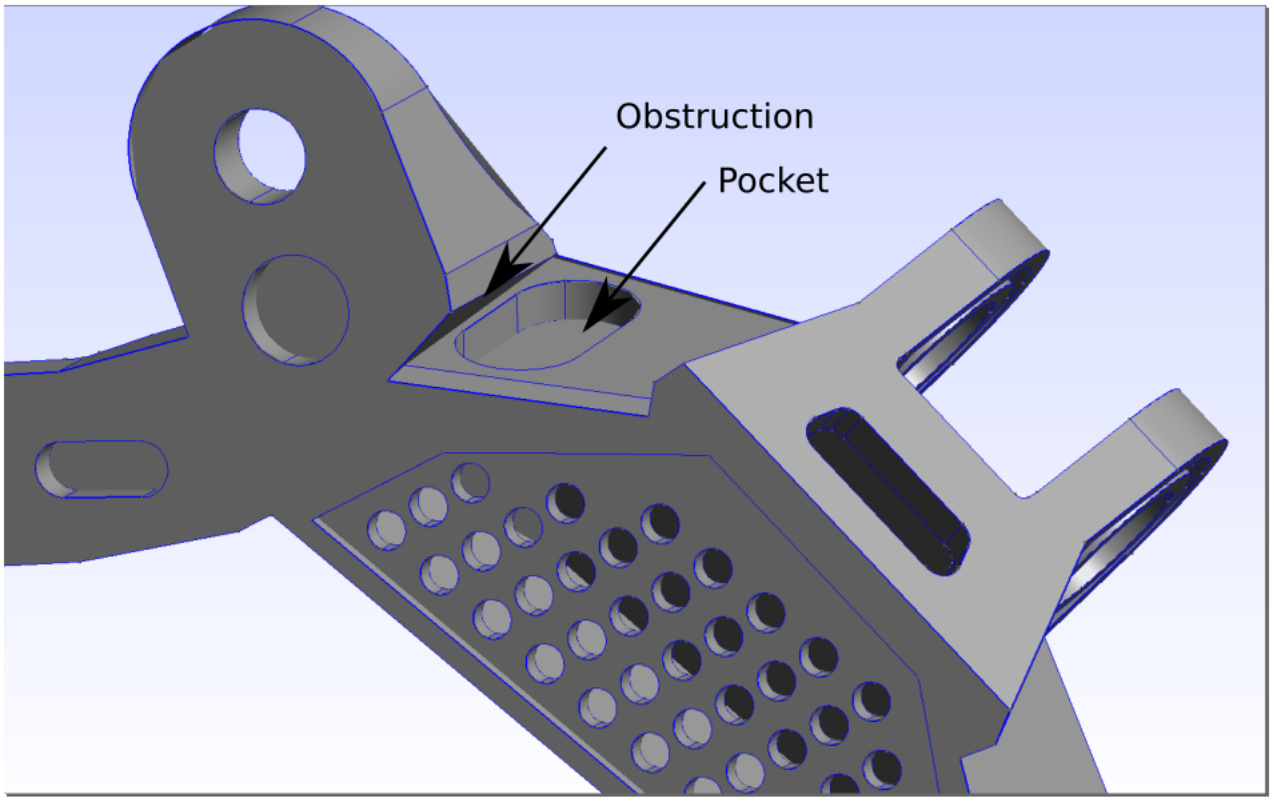


Figure 5: Defining various entities of a pocket