# *DEEP*

## **D**ual-space **E**xpansion for **E**stimating **P**enetration **D**epth between convex polytopes

http://gamma.cs.unc.edu/DEEP

Version 1.0

Application Manual

**Young J. Kim**

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3179
mailto:youngkim@cs.unc.edu
http://www.cs.unc.edu/~youngkim

## *INTRODUCTION*

Penetration depth (PD) is defined as the minimum translational distance to make the interiors of two inter-penetrating objects disjoint. DEEP is an incremental penetration depth algorithm to estimate the PD between *convex* polytopes. It is designed and implemented on top of SWIFT++ system, which you can obtain freely at http://gamma.cs.unc.edu/SWIFT++/. Hence, DEEP's API is essentially the same as that of SWIFT++. This manual will cover DEEP's basic functionality and its usage. For more technical information about DEEP, we refer the readers to the related publications available at http://gamma.cs.unc.edu/DEEP.

## *LICENSE AGREEMENT*

and J. O'Rourke. DEEP is written in Visual C++ and all the source code is included in the distribution. The LICENSE file also covers redistribution and code modifications.

## *BUILD*

### 1. FILES

First of all, you will need to download SWIFT++ 1.1 library and other necessary library files that SWIFT++ requires as specified in its manual. Once you unzip DEEP.zip, you will find the following files in each directory:

./include
SWIFT.h
SWIFT_common.h
SWIFT_convconv.h
SWIFT_front.h
SWIFT_hash.h
SWIFT_linalg.h
SWIFT_mesh.h
SWIFT_pqueue.h

./src
convconv.cpp
hash.cpp
mesh.cpp
pair.cpp
scene.cpp

./example
pdtest.cpp
sphere100.tri
sphere100_xz.tri
DEEP.doc
DEEP.pdf

You need to overwrite the header files and source files in SWIFT++ directory appropriately with the above files. Test sample files are also provided to give an example of the usage of DEEP library.

### 2. COMPILATION

The compilation process of DEEP is the same as that of SWIFT++. However, DEEP is written under Visual Studio environment, thus you will need to compile it under Visual C++. You need to turn on the following compilation switches to use DEEP in SWIFT++:

**SWIFT_PD_EST**
**SWIFT_FRONT_TRACKING**

## *API*

Basically DEEP uses the same API in SWIFT++, and its semantics is also the same. Therefore, you are strongly encouraged to get familiarized with SWIFT++ and its API's before using DEEP.

When the user does not need PD computation, he/she can use all the functions that SWIFT++ provide. However, when it comes to computing PD using DEEP, the user will use the following two API's most of time: *Add_General_Object* and *Query_Contact_Determiniation*.

1. *Add_General_Object* has the same usage as SWIFT++, except that you must use only **convex** objects to be used in DEEP (note: You can also use *Add_Convex_Object* too).

2. *Query_Contact_Determination* is the essence of DEEP library. In fact, *Query_Contact_Determination* is the only query API that is supported in DEEP. In other words, the other query methods explained in SWIFT++ manual Sec 6.5 have no meaning as far as the PD computation is concerned. In DEEP, *Query_Contact_Determination* has the following meaning:

   - **early_exit**: needs to be turned off.
   - **tolerance**: same as SWIFT++.
   - **nearest_pts**: meaningless in DEEP.
   - **num_pairs, oids, num_contacts**: same as SWIFT++.
   - **distances**: PD value.
   - **normals**: same as SWIFT++. The normals are normal vectors associated with feature_ids explained next. In DEEP, you can think of this vector as penetration direction for each convex polytope.
   - **feature_types, feature_ids**: types and ids of features that realize the returned PD. More specifically, there exist a supporting plane on each convex polytope that is normal to the features (feature_types and feature_ids). For more information, refer to the DEEP paper.

## *INPUT*

You can use any convex object that is supported by SWIFT++ library in TRI, POLY or DCP format. However, for non-convex objects, either DEEP will crash or it will report meaningless results.

4/15/2002 Young J. Kim