



**OpenEye**  
Scientific Software

**POSIT**  
*Release 1.0.1*

**OpenEye Scientific Software, Inc.**

March 13, 2012



# CONTENTS

<b>1</b>	<b>Front Matter</b>	<b>1</b>
<b>2</b>	<b>Installation and Platform Notes</b>	<b>3</b>
2.1	Licenses . . . . .	3
2.2	Installation . . . . .	3
2.3	Uninstallation . . . . .	5
<b>3</b>	<b>Introduction</b>	<b>7</b>
3.1	Using the Bound Ligand . . . . .	7
<b>4</b>	<b>Theory</b>	<b>11</b>
4.1	TanimotoCombo . . . . .	12
4.2	Optimization . . . . .	12
4.3	POSIT Cross Docking Results . . . . .	13
4.4	Predicting the Quality of the Pose . . . . .	14
4.5	A Note About Strain . . . . .	15
4.6	Additional Constraints (MCS) . . . . .	15
<b>5</b>	<b>Usage</b>	<b>17</b>
5.1	Concerning Clashing Poses . . . . .	17
<b>6</b>	<b>MAKE_POSE_RECEPTOR Usage</b>	<b>21</b>
6.1	Command Line Interface . . . . .	21
6.2	Example Commands . . . . .	23
<b>7</b>	<b>COMBINE_RECEPTORS Usage</b>	<b>25</b>
7.1	Command Line Interface . . . . .	26
7.2	Example Commands . . . . .	27
<b>8</b>	<b>POSIT Usage</b>	<b>29</b>
8.1	Command Line Interface . . . . .	30
8.2	Example Commands . . . . .	34
<b>9</b>	<b>Release Notes</b>	<b>37</b>
9.1	POSIT v1.0.1 . . . . .	37
9.2	POSIT v1.0.0 . . . . .	37
	<b>Bibliography</b>	<b>39</b>
	<b>Index</b>	<b>41</b>



# FRONT MATTER

Copyright 1997-2011 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.



# INSTALLATION AND PLATFORM NOTES

## 2.1 Licenses

To run POSIT and the associated utilities you will need to obtain a license file for POSIT from OpenEye Scientific Software ([business@eyesopen.com](mailto:business@eyesopen.com)). The license file should be in a file pointed to by the `OE_LICENSE` environment variable.

## 2.2 Installation

### 2.2.1 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory there will be a subdirectory for each application. Within that will be another subdirectory for each version of that application.
- bin** This directory contains a startup script for each application that has been installed. This script determines, at run-time, what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.
- data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.
- docs** This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

**examples** This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information, as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- `OE_ARCH` can be used to specify a colon separated list of compatible distributions for the current platform such as:

```
redhat-RHEL5-x64:redhat-RHEL4-x64
```

Specification of this environment variable overrides the auto-detection process, if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- `APPNAME_OE_ARCH` can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like `OE_ARCH` as detailed above.

Specification of this environment variable overrides the `OE_ARCH` environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the `OE_ARCH` list first and then to the auto-detection process.

Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

- |                     |  |
|---------------------|--|
| <b>-path</b>        | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present.  |
| <b>-print_arch</b>  | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| <b>-use_version</b> | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run.   |

## 2.2.2 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the `PATH` environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the `PATH` environment variable should contain: `/usr/local/openeye/bin`.

## 2.2.3 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, as well as to a Windows command shell which has

the appropriate `PATH` settings already defined to allow the user to simply type the executable name at the prompt without concern for where the executable is actually installed.

For GUI applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

## 2.2.4 Mac OS X

Mac OS X distributions are provided as a standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the `PATH` environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

## 2.3 Uninstallation

### 2.3.1 Linux/Unix

To uninstall a single distribution of a product the relevant subdirectories for that product and version simply need to be deleted from within the following directories:

- arch** In the `openeye/arch` directory is a platform specific subdirectory. Within this are directories for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled. For example, to delete or uninstall v1.0.0 of a product, delete the folder “<product\_name>/1.0.0”.
- data** In the `openeye/data` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.
- docs** In the `openeye/docs` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.
- examples** In the `openeye/examples` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

### 2.3.2 Windows

Installation of an OpenEye product on Windows causes an OpenEye group with an application specific subgroup to be added to the *Start* menu. One of the items in the application specific subgroup is a link to the uninstaller. Clicking on the uninstaller initiates a wizard which guides the user through uninstallation.

For GUI applications, uninstallation also removes the desktop link to the application as well as in the application specific subgroup of the *Start* menu.

### 2.3.3 Mac OS X

To uninstall a single distribution of a GUI application simply drag the application from /Applications/OpenEye/bin to the Trash can.

To uninstall a single distribution of a command line application you will need to delete the executable/folder from /Applications/OpenEye/arch/osx-10.6-x64/. For example, to delete or uninstall v1.0.0 of a product, delete the folder “1.0.0” located in /Applications/OpenEye/arch/osx-10.6-x64/<product\_name>.

Associated documentation, data and example files for a single distribution can be uninstalled by deleting the subdirectory/folder from within the /Applications/OpenEye/data, /Applications/OpenEye/docs and /Applications/OpenEye/examples directories.

# INTRODUCTION

During a drug discovery campaign, thousands of small molecule inhibitors are made in the course of optimizing molecular properties. For projects that have X-Ray crystallographic (XRC) coordinates, structure-based designs help guide the medicinal chemistry efforts. In many cases XRC provides a detailed picture of the binding of a small-molecule inhibitor into the binding site.

Many techniques exist for pose-prediction and are well documented [Vieth-2004]. However, very few provide a probability that the generated pose is correct, where correct is typically considered to be less than 2.0 Ångströms RMSD (root mean square distance) from experimental crystal structure. In fact, many docking scores such as Chemscore, Chemgauss3, PLP [Martinelli-2010] are not very correlated with correct ligand pose, and worse, are not transferable between systems: the best docking score in one system may not even be close to the best docking score in another.

Two definitions will be used during the remainder of this discussion:

- **Bound-Ligand** This is a known, experimentally derived bound ligand from the same protein context in which POSIT is attempting to find poses of ligands.
- **Fit-Ligand** This is the unknown ligand that is being pose-predicted.

## 3.1 Using the Bound Ligand

POSIT overcomes these issues by comparing predicted poses to observed bound ligands in related co-crystals. As the observed ligand becomes more similar to the predicted pose, both the binding mode and, indeed, the shape of the receptor pocket itself tends to become more similar. This is shown in figure: *Active Sites*.

The similarity measures being used are 2D path-based fingerprints and the 3D *TanimotoCombo* [Hawkins-2010] that compares shape and the Mills-Dean approximation of electrostatics [MillsDean-1996]. These similarity measures choose the most appropriate system to dock against and provide a prediction of the quality of the result. A full description of the Tanimoto measures are given in the *Theory* section.

The *TanimotoCombo* measure is agnostic of how the poses in question are generated; it can be used to validate and provide a pose prediction probability regardless of how the pose was generated. In practice, POSIT exploits the predictive capabilities of the *TanimotoCombo* measure by using it in an optimization function that drives a flexible fitting routine.

Essentially, during pose optimization, POSIT attempts to force a predicted pose into the binding mode of a known ligand. If the induced strain becomes too large, the optimization stops. This final pose is used to predict the overall quality of fit. In this fashion, POSIT is able to rescue 10-20% of the original rigidly overlaid poses and place them within 2.0 Ångströms RMSD of the experimental crystal structure.

Typically during docking, only the protein structure is used to model unknown structures. Given a molecule that is known to bind, POSIT searches through XRC coordinates of known ligand-protein complexes, determines the complex

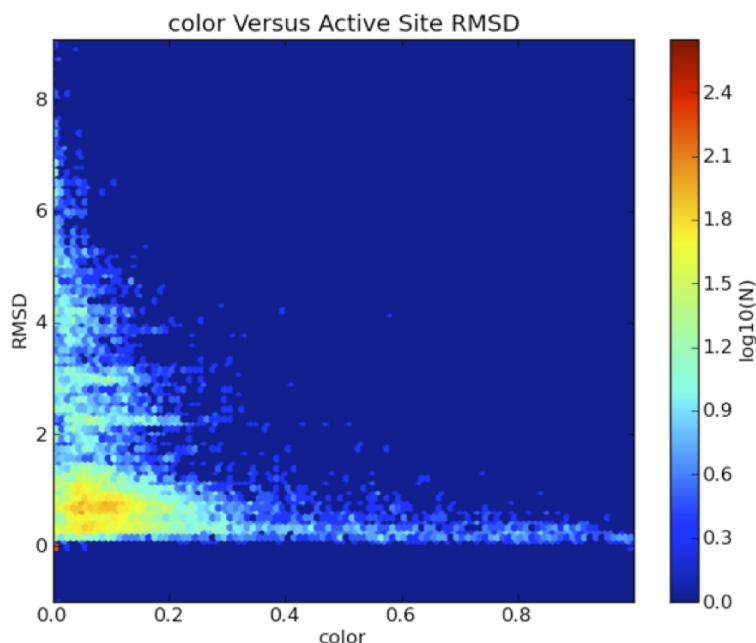


Figure 3.1: **Active Sites:** The RMSD of aligned active sites decreases as the Color Tanimoto between bound ligands decreases

best able to predict the pose of the molecule and then generates both a pose and the probability that the pose is correct, usually in well under a minute per ligand. POSIT's basic algorithm:

1. Given a set of potential complexes, POSIT chooses the appropriate complex based on the 2D or 3D similarity to the bound ligand. The best complex, in general, has the highest 2D or 3D similarity of the input molecule to the chosen complex's bound ligand.

POSIT can employ multiple methods to generate the initial overlays including an *TanimotoCombo* and *MCS* (Maximum Common Substructure) overlays. Both are done by default.

2. After the complex is chosen, a flexible fit is performed that attempts to match the binding mode of the bound ligand using an adiabatic optimization method [Wlodek-2006]. This optimization method is known as the POSIT potential.

The term adiabatic comes from the Greek "impassable", and in this case POSIT sets up a chemical strain boundary that the optimization cannot broach.

POSIT seeds the flexible fit by expanding the poses generated by the original 3D similarity as described in (1) and then applying the shape constraint of the bound ligand.

As shown in figure *POSIT Optimization*, POSIT works by first using the known bound ligand to position the input molecule and follows up by using the bound ligand as a shape constraint during MMFF optimization [Halgren-I-1996] [Halgren-II-1996] [Halgren-III-1996] [Halgren-IV-1996] [Halgren-V-1996] [Halgren-VI-1999] [Halgren-VII-1999]. While the input molecule is being forced into the shape constraint, MMFF strain is monitored to form the adiabatic boundary. When the strain becomes too large, the optimization is reversed or stops altogether.

3. The interactions from the bound ligand are then used as a further constraint during ligand-protein optimization. This helps to remove clashes with the protein and provide better interactions between un-constrained ligand atoms.

4. Finally, POSIT supplies a robust probability that the given pose is reasonable. It is generally recognized that docking and scoring methods have inaccuracies and do not provide a measure that can be compared between different complexes. For example, a docking score from one complex cannot be directly compared to a docking score from another.

POSIT probabilities were generated using a large testset containing over 25,000 pose predictions and verified through a smaller number (around 100) of predictions that were then validated with X-Ray crystallography. It is important to note that POSIT does not give a probability of binding, rather it gives a probability that **if the ligand does actually bind**, what is the likelihood of the POSIT pose being the actual pose.

This is a long winded way of saying that POSIT's optimization attempts to force the molecule into the known binding mode without creating undue strain on the molecule being placed into the protein. For more detail, please see the *Theory* section.

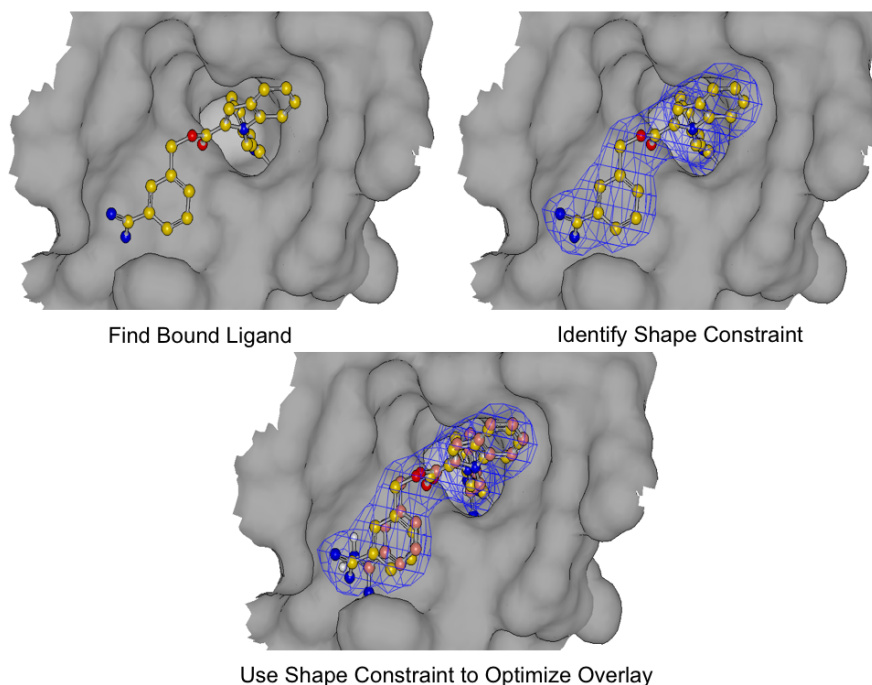


Figure 3.2: **POSIT Optimization:** Starting from an initial alignment, use the shape constraint of the bound-ligand to drive a flexible fit while simultaneously limiting strain



# THEORY

POSIT is primarily based on the assumption that similar ligands bind similarly. Unlike most docking protocols, POSIT **requires** the presence of a known bound ligand. The bound ligand is used to impart docking constraints when placing and optimizing the geometry of the molecule being docked.

The basic POSIT workflow is shown in Figure *POSIT Application Workflow*.

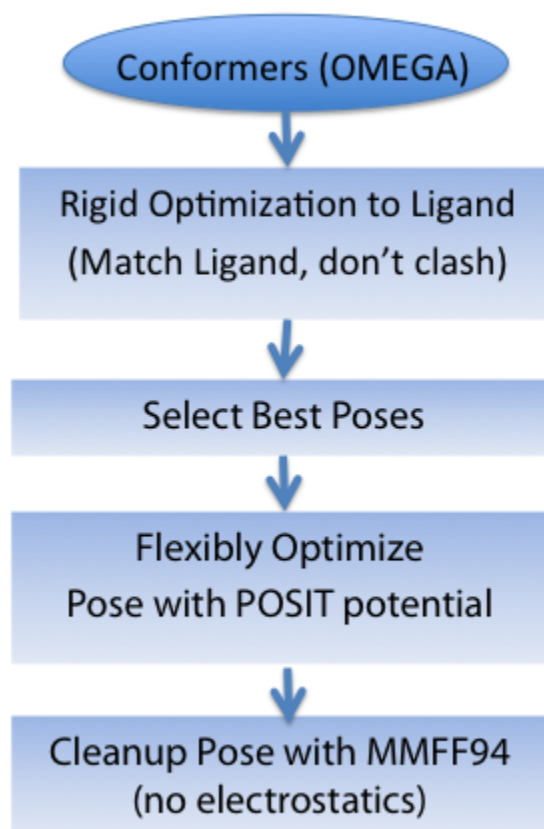


Figure 4.1: **POSIT Optimization Workflow:** The process POSIT uses to predict a pose is as follows. First, high-quality bio-active conformations are generated. These are then rigidly compared to the known bound ligands and the most appropriate receptor is chosen. Then, a full coordinate optimization is used to see if the current pose can fit the known interactions without undue strain. Finally, hydrogens and portions of the molecule that do not fit the known data are allowed to cleanup and are optimized with the protein.

POSIT's first pass is to identify a target receptor that has the most similar bound ligand. Multiple receptors are not required, but increase the odds of POSIT finding a suitable receptor. Given an input molecule, POSIT identifies the most similar bound ligand using a combination of graph similarity to the bound ligand (*MACCS 166* set or OpenEye's *PATH fingerprints*) and 3D similarity, *TanimotoCombo*, to the bound ligand.

These similarities have been calibrated to produce a probability of predicting a pose within 2.0 Ångströms RMSD. For example, consider the following POSIT log file output when fitting against a collection of vascular endothelial growth factor receptors:

```
Ligand 7 (3CJF)
Ligand 7 (3CJF): Receptor: 1Y6A_AAZA201A.oeb.gz (...) Probability: 0.36 TanimotoCombo: 0.95
Ligand 7 (3CJF): Receptor: 1Y6B_AAX201A.oeb.gz (...) Probability: 0.36 TanimotoCombo: 0.93
Ligand 7 (3CJF): Receptor: 1YWN_LIF301A.oeb.gz (...) Probability: 0.53 TanimotoCombo: 0.77
Ligand 7 (3CJF): Receptor: 2OH4_GIG303A.oeb.gz (...) Probability: 0.53 TanimotoCombo: 0.76
Ligand 7 (3CJF): Receptor: 2P2I_608501A.oeb.gz (...) Probability: 0.53 TanimotoCombo: 0.90
Ligand 7 (3CJF): Receptor: 3C7Q_XIN1172A.oeb.gz (...) Probability: 0.20 TanimotoCombo: 0.70
Ligand 7 (3CJF): Receptor: 3CJF_SAV1167A.oeb.gz (...) Probability: 0.98 TanimotoCombo: 1.56
Ligand 7 (3CJF): Receptor: 3CJF_SAV1167A.oeb.gz (...) Probability: 0.98 TanimotoCombo: 1.56
Ligand 7 (3CJF): Receptor: 3CJF_SAV1167A.oeb.gz (...)
    Automatically Chosen (probability > 0.900000)
```

In this kinase example the best match to Ligand 7 (taken from 3CJF) is, unsurprisingly, the 3CJF receptor which yields a probability of 98% percent of producing a predicted pose within 2 Ångströms RMSD to the actual bound ligand. (*The receptors' names have been clipped to ... for space issues*)

As described below, these probabilities were computed using an analysis of deposited structures in the RCSB protein data bank [PDB].

## 4.1 TanimotoCombo

POSIT uses the *TanimotoCombo* measure to compare (and optimize) predicted and bound ligands. The *TanimotoCombo* measure is simply two separate Tanimoto measures added together. While most uses of Tanimoto have been to compare fingerprints together, there is a direct relation between the 1D fingerprint bit vector and 3D space:

The basic equation for a field Tanimoto between two fields A and B is:

$$Tanimoto_{A,B} = \frac{\int A(\vec{r}) * B(\vec{r})}{\int A(\vec{r}) * A(\vec{r}) + \int B(\vec{r}) * B(\vec{r}) - \int A(\vec{r}) * B(\vec{r})}$$

In the case of POSIT, the field in question can be thought of as field of voxel space. For *Tanimoto<sub>shape</sub>*, where A and B are now molecules: if two objects fill the same voxels, then the Tanimoto value is 1.0. If two objects don't overlap by half, the Tanimoto value is 0.5 and so on. (The term voxel is used for purposes edification, in actuality the volumes estimated using a fast approximate method)

The field can also contain colored representations of chemistry. For example, if two voxels are colored as hydrogen bond donors and overlap, the *Tanimoto<sub>color</sub>* increases.

Hence, *TanimotoCombo* is:

$$TanimotoCombo = Tanimoto_{shape} + Tanimoto_{color}$$

*TanimotoCombo* values range from 0 (no overlap) to 2.0 (full shape overlap and full color or chemistry overlap).

## 4.2 Optimization

POSIT attempts to increase the *TanimotoCombo* overlap of the predicted pose to the known pose. Given a collection of potential rigid overlays, POSIT adapts the initial conformation to the computed electron density of a known pose using

Overlap(q,t) = Intersection of Volumes q and t

$$T = \frac{\text{Overlap}(q,t)}{\text{Overlap}(q,q) + \text{Overlap}(t,t) - \text{Overlap}(q,t)}$$

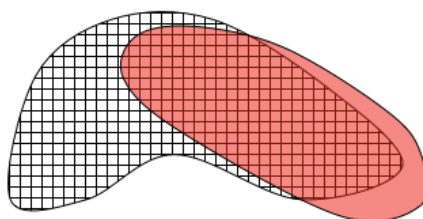


Figure 4.2: **Voxel Representation of Shape:** Similar to fingerprint bits in 1D, voxels can be used to represent 3D space and compared with the Tanimoto measure. Note that  $\text{Overlap}(q,q)$  is essentially the same thing as the volume of  $q$

a modern force-field, MMFF94 [Halgren-I-1996] [Halgren-II-1996] [Halgren-III-1996] [Halgren-IV-1996] [Halgren-V-1996] [Halgren-VI-1999] [Halgren-VII-1999].

The potential function being used to adapt the ligand is:

$$V = V_{ff} + \lambda V_{shape}$$

where  $V_{ff}$  represents the internal energy of the ligand and  $V_{shape}$  is the overlap between the ligand and the electron density.  $\lambda$  is a mixing parameter that represents the degree to which the shape of the density dominates the combined potential during the current optimization step [Wlodek-2006]. Most of the time, as the mixing parameter  $\lambda$  increases, the molecular strain also increases. This strain forms the adiabatic, or impassable, boundary that limits the conformations to reasonable geometries.

Essentially, the strain placed on the ligand is bounded while the ligand is optimized in to the shape constrain. This produces high-quality alignments while maintaining low-strain ligand conformations.

### 4.3 POSIT Cross Docking Results

As described in the description of the base algorithm given in section *POSIT base algorithm*, POSIT performs a fast initial overlay and then a flexible optimized overlay. Interestingly, if POSIT can optimize the structure closer to the bound ligand, the probability of the correct pose can actually increase.

As shown in figure *POSIT Cross Docking Results*, analyzing the Kinase data set used in Tuccinardi et al, [Martinelli-2010] pose-prediction using POSIT is seen to perform remarkably better for similar ligands than standard docking techniques:

While POSIT is not a good technique for determining the pose between known bound-ligands and fit ligands with low similarity, as the similarity increases, the probability of determining the correct pose increases rapidly. The similarity where this crosses over is remarkably small, only around 0.9 *TanimotoCombo*. This is most likely because, as shown in figure *Active Site RMSD Versus Color Tanimoto*, as the similarity increases, the active site similarity also increases

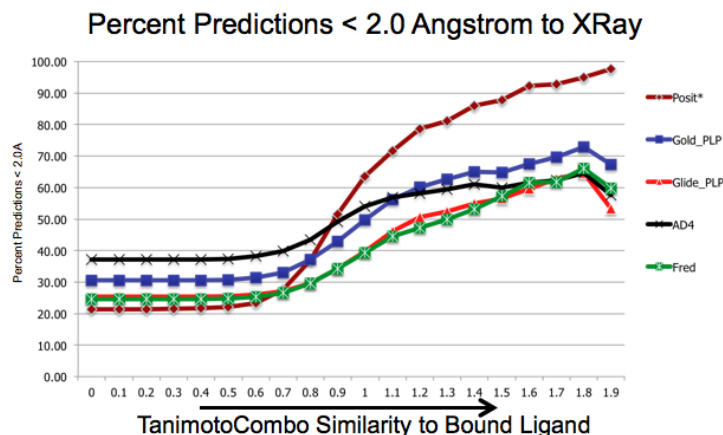


Figure 4.3: **POSIT Cross Docking Results:** Probability of finding a good pose based on bound-ligand fit-ligand *TanimotoCombo* similarity. Standard docking results are essentially the same and follow the same trajectories flattening out as they hit their limit of accuracy. While POSIT performs worse at low similarities it continually increases as similarity increases.

## 4.4 Predicting the Quality of the Pose

POSIT uses a robust measure of shape and chemical similarity to provide a probability of the generating the correct pose. Again, the definition used for *correct* is a pose less than 2.0 Ångströms to the experimental crystal structure.

Using a combination of public data and proprietary data obtained experimentally from collaborators, basic descriptors of 2D and 3D similarity between the ligand being fit and known bound ligands were analyzed to provide a basis for predicting the likelihood of obtaining a docked pose within 2.0 Ångströms when using a known target receptor as the docking target.

Figure *POSIT Probability MAP* shows how the beliefs given by the 2D and 3D measures are combined into a probability of having a good pose. Remember that this probability has been generated from ligands that actually bind, hence, it is not a probability of binding.

This result is different from the result shown in [Martinelli-2010], where they reported that having a high *TanimotoCombo* to the known bound ligand did not dramatically increase the quality of the resulting pose (even for FRED). The reason is subtle: Martinelli *et al* were computing the highest *TanimotoCombo* that the two molecules could obtain, while POSIT computes the actual, docked, in-place *TanimotoCombo* of the fitted pose. That is, if the docking algorithm produces an alignment of fit molecule to known bound ligand that overlaps with a given *TanimotoCombo*, then one can look up the probability in section *POSIT base algorithm*. In point of fact, POSIT is specifically designed so that the docked pose obtains the highest *TanimotoCombo* score possible while simultaneously minimizing induced strain and maintaining interactions with the protein.

Using the data shown in *POSIT Probability MAP*, for each pose, POSIT reports a simple score for the quality of the pose. The probability of a good pose is binned into the following results:

Result	Meaning
GREAT	The computed pose is likely (75%-100%) probability) the crystallographic pose.
GOOD	The computed pose may be (50%-75%) probability) the crystallographic pose.
POOR	Take with a huge grain of salt (<50% probability)

By default, POSIT only outputs poses that are GOOD or better or that have a probability greater than or equal to 50% of being correct.

Poses that clash with the reference protein are not output. Both of these properties, probability and clash distance, can be tailored to individual preferences.

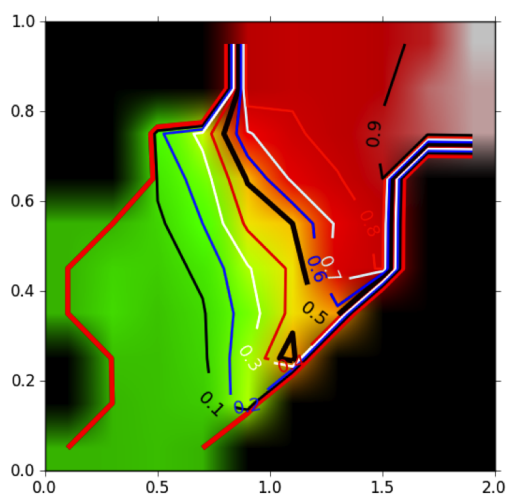


Figure 4.4: **POSIT Probability Map:** Given a 2D similarity (in this case the MACCS 166 descriptor set) and a 3D similarity (*TanimotoCombo*) POSIT computes a probability of finding the correct pose based on an analysis of historical and experimental data.

If multiple reference receptors are input, if a receptor complex is found with a probability greater than 90% of finding a good pose, that receptor is automatically chosen for POSIT, otherwise the best matching receptor is used.

## 4.5 A Note About Strain

As POSIT attempts to flexibly fit the pose into the known bound-ligand, it allows a measure of strain to be induced. By default, POSIT limits the strain to 10 kcal. This number was chosen because of an independent analysis of ligands in the pdb [Perola-2004]. This strain is called *Local Strain* and is an estimate of the induced energy difference between the minimum and the posed molecule. Even if a molecule fits well or has a high probability, it is worth considering the *Local Strain* when analyzing the molecule.

## 4.6 Additional Constraints (MCS)

Walter's *et al* noted that a large portion of ligands bound to the same protein kinase share a large maximum common substructure(MCS). This was the basis for their **CORES** algorithm [Bemis-2004]. POSIT can optionally identify matching regions and use them as additional constraints during optimization.

If the *-mcs* flag is supplied, POSIT searches for an MCS match to the bound ligand. The matching portion is used as the shape constraint, the rest is optimized against the protein.



# USAGE

Along with FRED, POSIT is part of the **OEDocking** suite of docking tools. Similar to FRED, protein-ligand complexes are stored in receptor format for easy analysis.

Receptors are simply proteins with extra information attached such as the bound ligand and the location and shape potential of the active site. Unlike FRED, POSIT only operates on receptors with bound ligands. Receptors files and the extended information may be viewed in VIDA.

The POSIT suite of tools includes:

- **make\_pose\_receptors** - tool to generate receptors with bound ligands
- **combine\_receptors** - tool to automatically combine receptors with bound ligands that cover different portions of the active site
- **posit** - pose prediction using known bound ligands (at minimum, requires a receptor and a molecule to pose predict).

The typical workflow is to first use **make\_pose\_receptors** to create receptor targets for all the proteins in a project and then use **posit** to select the best receptor for a given pose prediction and generate a predicted pose. This workflow is shown in Figure *Posit Application Suite Workflow*.

The usage of each tool is described in the following chapters.

## 5.1 Concerning Clashing Poses

The definition of clashes is somewhat problematic for purposes of pose prediction. In general, serious clashes where interpenetration with the protein should be avoided at all costs. However, when docking into a rigid protein that does not have the appropriate conformation, rigid docking ignores that fact that the active sight may adopt a conformation suitable to the posed ligand.

POSIT deals with clashes by allowing the user to specify three allowable clash levels with cutoffs taken by analyzing various ligands in deposited in the protein data bank (*RSCB*).

Allowed Clash	Description
noclashes	No clashes are allowed. Actually there is a little wiggle room here less than 0.2 Ångström penetration is not considered a clash.
mild-clashes	Mild clashes are allowed ( $\geq 0.2$ Ångstroml $< 0.65$ Ångström interpenetration)
allclashes	All clashes are allowed.

If a pose clashes, it is not thrown away, it is written to the specified fail file. Fail files hold ligands with decent probability when compared to the bound ligand, but have unallowable clashes with the protein.

# POSIT Workflow

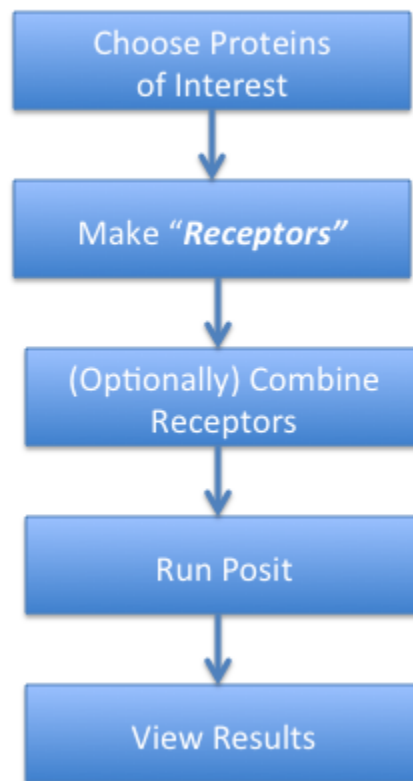


Figure 5.1: **POSIT Application Suite Workflow:**

Clashing can also affect the known bound ligand. When making a receptor for POSIT, if a bound ligand clashes with the protein beyond the allowable clash level, a warning will be given and the receptor will not be generated. In this case, the appropriate command line switch will be given in order to generate the receptor. It is advisable to visually inspect clashing complexes (and electron density if available) to decide whether the clash is acceptable or not.

Note that if a receptor is made with the *-allclashes* option, POSIT should also be run with the same option or the *-clash* file should be specified on the command line.

See below for more details.



---

# MAKE\_POSE\_RECEPTOR USAGE

---

POSIT provides a simple program for automatically generating receptors from a protein and a ligand. It can generate receptors by providing the ligand and protein pre-split or by providing the full complex and specifying the residues of the bound ligand.

All receptors created by **make\_pose\_receptor** are accepted by any of the **OEDocking** suite of docking tools.

If a crystallographic ligand has significant clashes with the protein a warning will also be output with the recommended setting when running POSIT. Serious clashes will not be output unless the user specifies the *-allowedClashes* option.

When using the *-auto* flag, If the complex has multiple bound ligands, multiple receptors will be generated for each bound ligand (even if they are copies of the same ligand).

## 6.1 Command Line Interface

A description of the command line interface can be obtained by executing **make\_pose\_receptor** with the *--help* option.

```
prompt> make_pose_receptor --help
```

will generate the following output:

```
Help functions:
  make_pose_receptor --help simple      : Get a list of simple parameters
  make_pose_receptor --help all        : Get a complete list of parameters
  make_pose_receptor --help <parameter> : Get detailed help on a parameter
  make_pose_receptor --help html       : Create an html help file for this program
```

### 6.1.1 Required Parameters

#### **-protein**

The protein to use as the basis for the receptor.

#### **-out**

The output receptor file. Note that all receptors must be stored in OEB formats so any extension given will be forced to be OEB.

Note: If the **-auto** flag is used, each output receptors will be named based on the auto-detected residue names. See *-auto* below for details.

#### **-ligand**

Either **-ligand**, **-residues** or **-auto** is required.

If **-ligand** is given, a receptor will be generated with the supplied ligand. As a sanity check, if the ligand overlaps the protein, the receptor will fail with a warning and list the clashing residues.

**-residues**

Either **-ligand** or **-residues** are required to create a complex. If neither are supplied, then **make\_pose\_receptor** will attempt to find a bound ligand and print out the **-residues** string for the suggested ligand.

If **-residues** is supplied, the matching residues will be used as the ligand portion of the molecule. Residues are supplied as comma separated list of ligand residues.

The easiest way to find what to enter for the **-residues** flag is to simply run **make\_pose\_receptor** with only a **-prot** flag.

**-auto**

Automatically extract receptors using all non-covalent ligands.

```
prompt> make_pose_receptors -prot 1Y6A.pdb.gz
Found ligands:
  1. CCS(=O)(=O)c1ccc(c(c1)Nc2ncc(o2)c3cccc(c3)c4cccn4)OC
To extract ligand, add command line switch:
  -residues AAZA201A or -residues 1
```

Note that, as a convenience, you can simply select the ligand by number as well.

Residues may be comma separated, for example AAZA201A,AAZA201B to indicate that the ligand is actually two residues.

[default = false]

## 6.1.2 Optional Parameters

**-allowedClashes**

Clashes allowed between the ligand and protein of a receptor. There are three levels:

Allowed Clash	Description
noclashes	No clashes are allowed. Actually there is a little wiggle room here less than 0.2 Ångström penetration is not considered a clash.
mildclashes	Mild clashes are allowed ( $\geq 0.2$ Ångstroml $< 0.65$ Ångström interpenetration)
allclashes	All clashes are allowed.

The clash ranges have been tuned to account for average coordinate error in a sampling of PDB files, they are intended to be used as guidelines and may not be indicative of some clash states.

[default = mildclashes]

By default, receptors are not output if serious clashes exist between the specified ligand and protein. By specifying **-allowedClashes allclashes** all clashing receptors will be output.

[default = mildclashes]

**-param** <filename>

Defines the control parameter file. This file can contain a collection of parameters which can be used instead of writing each parameter to the command-line. In addition, the parameter file written by any POSIT run (see **-prefix** below), can be used with the **-param** flag in subsequent POSIT executions. Any command given explicitly on the command line will supersede any command found in a file specified with the **-param** parameter.

**-prefix**

Controls the name of the default param file.

[default = make\_pose\_receptor]

## 6.2 Example Commands

The example commands in this section can be run with files found in the `example/posit/1.0.1` directory under the top-level installation directory.

To use these examples, enter the renin directory

```
prompt> make_pose_receptor -prot renin/2IKO.pdb.gz
```

You will see the following output (Banner omitted)

```
Found ligands:
 1. CCc1c(c(nc(n1)N)N)c2ccc(cc2)NCc3cc(cc(c3)F)F
    To extract ligand, add command line switch:
    -residues 7IG601A or -residues 1
```

What `make_receptor` is indicating is its best guess at which portion of the molecule is the ligand. Simply copy this command line switch to your command:

```
prompt> make_pose_receptor -prot 2IKO.pdb.gz -out 2IKO.rec.oeb.gz -residues 7IG601A
```

or

```
prompt> make_pose_receptor -prot 2IKO.pdb.gz -out 2IKO.rec.oeb.gz -residues 1
```

And the indicated residues will be used to separate the ligand from the protein.

When the following command line is executed, notice that the ligand is detected as clashing with the protein:

```
prompt> make_pose_receptor -prot %(DATA)s/renin/2IKU.pdb.gz -out receptors/2IKU_a.rec.oeb.gz \
  -residues LIY336A -allowedClashes noclashes
```

```
Protein (RENINLIY336A): Writing to receptor receptors/2IKU_a.rec.oeb.gz
  Adding LIY336B CCc1c(c(nc(n1)N)N)c2ccc3c(c2)N(C(CC3)c4ccccc4)CCCOC as an extra molecule
Warning: Receptor ligand LIY336A clashes with the proteins
  To force output please use command line switch -allowedClashes mildclashes
  Note that POSIT should most likely be run with the same switch.
Warning: No receptors written.
```

`MAKE_POSE_RECEPTOR` gives the appropriate command line in order to create the receptor, again it is advisable to visually inspect the complex and the original electron density if available.

In this case the following command will generate the desired receptor (note that `mildclashes` is the default allowable clash level).

```
prompt> make_pose_receptor -prot %(DATA)s/renin/2IKU.pdb.gz -out receptors/2IKU_a.rec.oeb.gz \
  -residues LIY336A-allowedClashes mildclashes
```



## COMBINE\_RECEPTORS USAGE

In many cases ligands from different receptors may be combined to yield more productive posing targets. For example, if, when the proteins are aligned, two ligands overlap but occupy different spaces in the binding pocket, `combine_receptors` will make a single receptor using both ligands that may be a better target for predicting some ligands.

When given multiple receptors, `combine_receptors` will find the best matched pair for each input receptor. The best matched pair is the combination of two ligands that overlap but have a large difference in shape.

For each set of combined ligands, two receptors are output, one for each input protein. This also helps when analyzing clashes.

Output names are automatically generated so that if two receptors are merged, say `A_receptor.oeb.gz` and `B_receptor.oeb.gz`, two outputs are formed:

- `A_receptor_merged_B_receptor.oeb.gz`
- `B_receptor_merged_A_receptor.oeb.gz`

For example, consider the merged receptors shown in figures *Merged receptors* and *Matched to merged receptor*.

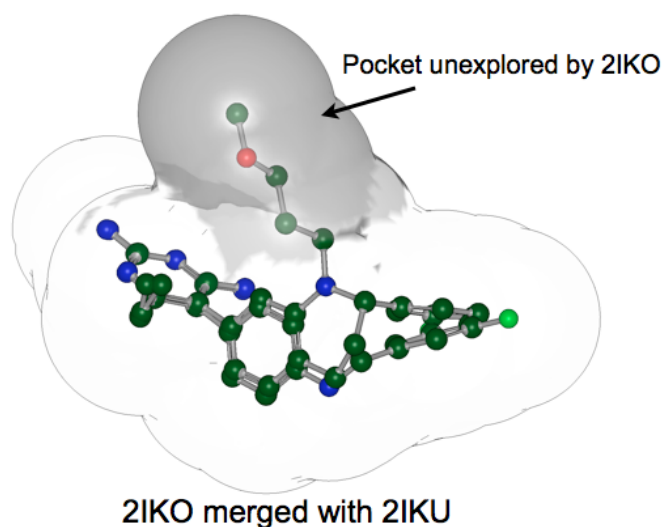


Figure 7.1: **Merged Receptors:** Merging the receptors 2IKO and 2IKU capture more potential interaction constraints than either does alone

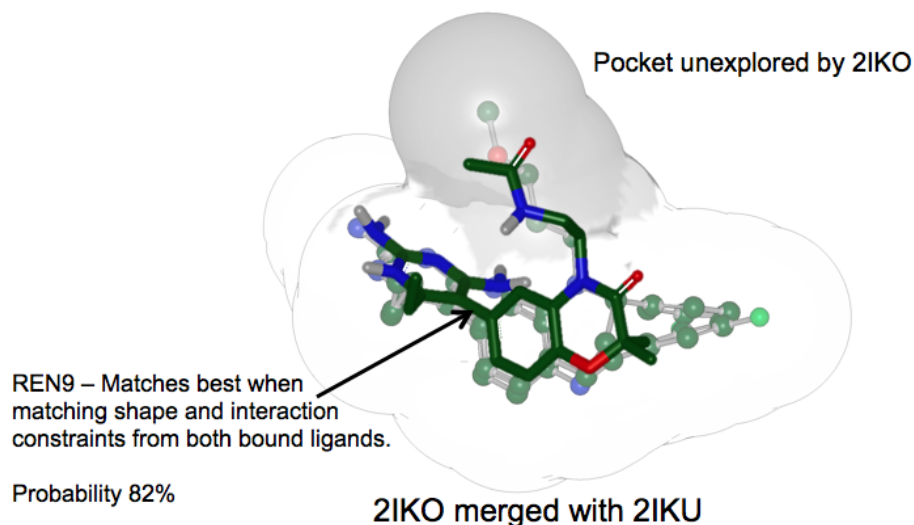


Figure 7.2: **Matched to Merged Receptors:** REN9 is correctly predicted by the merged receptor

If a combined ligand has significant clashes with the protein a warning will also be output with the recommended setting when running POSIT. Serious clashes will not be output unless the user specifies the *-allowedClashes* option.

## 7.1 Command Line Interface

A description of the command line interface can be obtained by executing COMBINE\_RECEPTORS with the *--help* option.

```
prompt> combine_receptors --help
```

will generate the following output (omitting the OpenEye banner for space)

```
Help functions:
  combine_receptors --help simple      : Get a list of simple parameters
  combine_receptors --help all        : Get a complete list of parameters
  combine_receptors --help <parameter> : Get detailed help on a parameter
  combine_receptors --help html       : Create an html help file for this program
```

### 7.1.1 Required Parameters

**-receptors** <filenames>

List of receptors to combine. If a file is not a receptor, **combine\_receptors** will abort.

### 7.1.2 Optional Parameters

**-allowedClashes**

Clashes allowed between the ligand and protein of a receptor. There are three levels:

Allowed Clash	Description
noclashes	No clashes are allowed. Actually there is a little wiggle room here less than 0.2 Ångström penetration is not considered a clash.
mildclashes	Mild clashes are allowed ( $\geq 0.2$ Ångström $< 0.65$ Ångström interpenetration)
allclashes	All clashes are allowed.

The clash ranges have been tuned to account for average coordinate error in a sampling of PDB files, they are intended to be used as guidelines and may not be indicative of some clash states.

[default = mildclashes]

By default, receptors are not output if serious clashes exist between the specified ligand and protein. By specifying **-allowedClashes allclashes** all clashing receptors will be output.

[default = mildclashes]

**-param** <filename>

Defines the control parameter file. This file can contain a collection of parameters which can be used instead of writing each parameter to the command-line. In addition, the parameter file written by any POSIT run (see *-prefix* below), can be used with the *-param* flag in subsequent MAKE\_POSE\_RECEPTOR executions. Any command given explicitly on the command line will supersede any command found in a file specified with the *-param* parameter.

**-prefix**

Controls the name of the default param file.

[default = combine\_receptors]

**-outputdir**

Write the merged receptors to the directory specified (the directory must already exist).

**-prealigned**

Indicate that your receptors are pre-aligned and do not change their alignment on pain of death.

[default = false]

**-verbose**

Show that matching status of every receptor pair (whether it is merged or not)

[default = false]

## 7.2 Example Commands

To run combine\_receptors, simply provide a list of known receptors:

First, make some receptors:

```
prompt> mkdir receptors
prompt> make_pose_receptor -prot 2IL2.pdb.gz -out receptors/2IL2_b.rec.oeb.gz -residues CIT501A
prompt> make_pose_receptor -prot 2IL2.pdb.gz -out receptors/2IL2_a.rec.oeb.gz -residues LIX402B
prompt> make_pose_receptor -prot 2IL2.pdb.gz -out receptors/2IL2_c.rec.oeb.gz -residues LIX401A
prompt> make_pose_receptor -prot 2IKO.pdb.gz -out receptors/2IKO.rec.oeb.gz -residues 7IG601A
prompt> make_pose_receptor -prot 2IKU.pdb.gz -out receptors/2IKU_a.rec.oeb.gz -residues LIY336A
prompt> make_pose_receptor -prot 2IKU.pdb.gz -out receptors/2IKU_b.rec.oeb.gz -residues LIY336B

prompt> combine_receptors -receptors receptors/*
```

**Note:** On WINDOWS systems, you may have to expand out the wildcard:

```
prompt> combine_receptors -receptors receptors/2IL2_b.rec.oeb.gz receptors/2IL2_a.rec.oeb.gz \
receptors/2IL2_c.rec.oeb.gz receptors/2IKO.rec.oeb.gz receptors/2IKU_a.rec.oeb.gz \
receptors/2IKU_b.rec.oeb.gz
```

In this case, there were only two good merges which results in four output files, one for each reference frame. For example if protein A can be combined with protein B, then `A_merge_B` and `B_merge_A` will be written. :

```
Wrote combined receptor:merged/2IKO.rec_7IG601A._merge_2IKU_a.rec_LIY336A.oeb.gz ShapeTanimoto:0.6..
Wrote combined receptor:merged/2IKU_a.rec_LIY336A._merge_2IKO.rec_7IG601A.oeb.gz
Wrote combined receptor:merged/2IKU_a.rec_LIY336A._merge_2IKO.rec_7IG601A.oeb.gz ShapeTanimoto:0.6..
Wrote combined receptor:merged/2IKO.rec_7IG601A._merge_2IKU_a.rec_LIY336A.oeb.gz
```

It is often useful to place the merged receptors into a different directory, to do this, simply use the `-outputdir` option.

```
prompt> combine_receptors -receptors receptors/* -outputdir merged
```

---

**Note:** Again, on some WINDOWS systems, you may have to expand out the wildcard (this will be omitted for all future examples):

---

```
prompt> combine_receptors -receptors receptors/2IL2_b.rec.oeb.gz receptors/2IL2_a.rec.oeb.gz \
receptors/2IL2_c.rec.oeb.gz receptors/2IKO.rec.oeb.gz receptors/2IKU_a.rec.oeb.gz \
receptors/2IKU_b.rec.oeb.gz -outputdir merged
```

**COMBINE\_RECEPTORS** deals with clashes identically to **MAKE\_POSE\_RECEPTOR**. If a merged ligand has an unallowable clash with a protein, it will not result in a receptor. An example output is as follows:

```
Warning: Combined ligand: 2IKU_LIY336A.oeb.gz clashes with protein in 2IKO_7IG601A.oeb.gz
To force output please use command line switch -allowedClashes mildclashes
Note that POSIT should most likely be run with the same switch.
Warning: Combined ligand: 2IKO_7IG601A.oeb.gz clashes with protein in 2IKU_LIY336A.oeb.gz
To force output please add the command line switch -allowedClashes mildclashes
it is recommended that POSIT be run with the same switch.
Warning: Combined ligand: 2IKO_7IG601A.oeb.gz clashes with protein in 2IKU_LIY336A.oeb.gz
To force output please use command line switch -allowedClashes mildclashes
Note that POSIT should most likely be run with the same switch.
Warning: Combined ligand: 2IKU_LIY336A.oeb.gz clashes with protein in 2IKO_7IG601A.oeb.gz
To force output please add the command line switch -allowedClashes mildclashes
it is recommended that POSIT be run with the same switch.
Warning: All potential mergings clashed with the merged protein.
```

# POSIT USAGE

POSIT works best when fitting to a collection of co-crystal ligands, however it can be used for a single ligand targets. Each pose that POSIT generates is analyzed with a simple heuristic and marked with a probability (seen below).

The structures output by POSIT are annotated with information about the best-fit receptor and with various metrics describing the details of the pose. These details are stored in SDDATA and can be viewed with most molecular structure viewers. The details are:

Name	Description
Receptor Name	the name of the receptor (taken from the original protein)
Receptor File	the filename of the original receptor
Result	GREAT, GOOD or POOR detailing the quality of the result
Probability	estimated probability of being within 2 A  RMSD of the real structure.
MACCS 166	the MACCS 166 similarity to the receptor ligand (used to compute Probability)
TanimotoCombo	the <i>TanimotoCombo</i> overlap with the receptor ligand
Local strain	the strain difference induced by the flexible optimization.
Forcefield	the forcefield used during the optimization MMFF94, MMFF Planar Constraint (see <i>-forcePlanarAromatic</i> below)
Alternate Pose	the alternate pose number if applicable
CLASH	Clashing distance in Ångströms

These values are also written to the report file in a tab separated format, see *-prefix* for details.

The probability that the computed pose is a correct pose is generate as described in *Predicting the Quality of the Pose*.

The values in the Result field are as follows:

Result	Meaning
GREAT	Computed pose is likely (75%-100%) probability) to be within 2.0 Å of experimentally-derived pose.
GOOD	Computed pose may be (50%-75%) probability) to be within 2.0 Å of experimentally-derived pose.
POOR	Take with a huge grain of salt (<50% probability)

By default, **Poor** poses are not written out, they are rejected as unsuitable. The entry in the log file looks as follows:

```
Ligand 1 (1ajx): Rejected for Receptor: receptors/recluwj.oeb.gz
  final tanimoto combo (0.854158) too low, should be at least 1.2000
  Use -minTanimotoCombo command line to decrease threshold
```

These annotations are computed using the final *TanimotoCombo* of the testset predicted-pose versus the co-crystal ligands.

## 8.1 Command Line Interface

A description of the command line interface can be obtained by executing POSIT with the `--help` option.

```
prompt> posit --help
```

will generate the following output:

```
Help functions:
  posit --help simple      : Get a list of simple parameters
  posit --help all         : Get a complete list of parameters
  posit --help <parameter> : Get detailed help on a parameter
  posit --help html        : Create an html help file for this program
```

### 8.1.1 Required Parameters

**-in** <filename>

Input molecules to pose-predict. If 3D molecules are input, only perceived stereochemistry is retained, all conformations are ignored and regenerated.

---

**Note:** POSIT uses an internal conformation sampling to generate

---

high-quality conformations. However, input ring structures (chair/boat) may not be reproduced as POSIT may choose a lower energy ring template during the sampling process.

Supported input file formats are:

File type	Extension
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
Smiles	.smi .smi.gz .ism.gz

**-receptors** <filenames>

List of receptors used to predict the pose of the input molecules. Receptors must include a bound ligand.

Multiple receptors can be input:

```
prompt> posit -receptors rec*.oeb
```

Will use all files that start with rec and end with ".oeb".

### 8.1.2 POSIT Output

POSIT can output results in three different styles:

- **-out** - Ligand only output in one file
- **-outcomplexes** - Interleaved complex output in one file
- **-outreceptors** - one file for each receptor matched containing all ligands that were fit to the receptor

Note: By default, not all poses may be written to the output, to see where ligands were placed, consult the log file for more details. To control this behavior use the **-outputall** flag.

- **-outputall** outputs all output to the output files regardless of whether they clash or fit poorly.

**-out** <filename>

The output file for the resulting poses. The best outputs for POSIT are .sdf and .oeb.gz since the poses are annotated with the best fit receptor and various optimization results such as the strain added to the rigid conformer.

Ligands are written to the output file based on their input order.

**-outcomplexes** <filename>

POSIT will output protein complexes to the specified filename. This is the largest output format that POSIT generates as each fit molecule will be output with the corresponding protein complex. POSIT interleaves the molecules as:

- 1.protein + bound ligand followed by fit poses for first ligand
- 2.protein + bound ligand followed by fit poses for second ligand

The filename will be use to generate a prefix to the output files. The extension must be a protein compatible format (.pdb, .oeb, .mol2)

**-outreceptors** <filename>

POSIT will write out the results with the best matching receptor. This allows for facile comparison with the bound ligand and protein complex when loaded in VIDA.

The matched receptor's filename will be used to generate a prefix to the output files.

The extension must be a protein compatible format (.pdb, .oeb, .mol2)

For example:

```
prompt> posit -outreceptors pcp_complex.oeb.gz -receptors rec1fpu.oeb reclogl.oeb
```

will generate the following output files:

- pcp\_complex\_rec1fpu.oeb.gz
- pcp\_complex\_reclogl.oeb.gz

Each file will start with the receptor as the first molecule and then proceed with the sorted list of best matching poses.

**-outputall**

Outputs all posed ligands to the file specified by *-out*, *-outreceptors* or *-outcomplexes* whether they clash or fit poorly. This is a shorthand for:

```
-minInitialTanimotoCombo 0.0 -minTanimotoCombo 0.0 -minProbability 0.0 \
  -allowedClashes allclashes
```

Using *-outputall* will override any non default settings.

**-clash** <filename>

Occasionally POSIT rejects ligands to due uncorrectable clashes. If desired, these clashing ligands will be written to this file using the same specification as the *-out* command option.

**-clashcomplexes** <filename>

Occasionally POSIT rejects ligands to due uncorrectable clashes. If desired, these clashing ligands will be written to this file using the same specification as the *-outcomplexes* command option.

**-clashreceptors** <filename>

Occasionally POSIT rejects ligands to due uncorrectable clashes. If desired, these clashing ligands will be written to this file using the same specification as the *-outreceptors* command option.

### 8.1.3 Optional Parameters

**-alternatePoses**

Generate alternate poses, the number of which depends on the internal algorithm. Duplicate conformers will be pruned, see -alternatePosesRMSD

[default = true]

**-alternatePoseRMSD**

When pruning alternate poses, reject any poses lower than this RMSD in Ångströms to any accepted pose.

[default = 0.5]

**-expandMissingStereo**

If the ligands to pose predict have non-assigned stereo centers, expand all stereo centers ( unless -ignoreNitrogenStereo is being used )

**-ignoreNitrogenStereo**

When examining the ligands to pose predict for stereo, ignore any missing nitrogen chirality. (This is normally caused by time averaging of crystal structures).

When expanding stereo, nitrogen stereo centers will not be assigned.

---

**Note:** POSIT may complain about stereo centers being changed by the optimization, this is more likely when ignoring nitrogen stereo centers since the optimization may decide a different stereo configuration is optimal.

---

**-allowedClashes**

Clashes allowed at the end of optimization. There are three levels:

Allowed Clash	Description
noclashes	No clashes are allowed. Actually there is a little wiggle room here less than 0.2 Ångström penetration is not considered a clash.
mildclashes	Mild clashes are allowed ( $\geq 0.2$ Ångström $< 0.65$ Ångström interpenetration)
allclashes	All clashes are allowed.

The clash ranges have been tuned to account for average coordinate error in a sampling of PDB files, they are intended to be used as guidelines and may not be indicative of some clash states.

**n.b. if any of the receptors themselves have clashing ligands, the -allowedClashes a warning will be generated but POSIT will automatically be set to the receptor ligands clash setting.**

All poses that are accepted by the probability calculation yet clash with the protein are written to the fail file (if specified).

[default = mildclashes]

**-forcePlanarAromatic**

Add a planar potential to aromatic rings, this attempts to keep aromatic systems planar. This mostly affects molecules that require high strain to match the bound ligand.

[default = true]

**-mcs**

Add MCS (Maximum Common Substructure) overlays to the final structures. The MCS overlay must have a minimum overlap compared to the bound ligand. If an MCS can be found, the base coordinates of the bound ligand are applied prior to optimization in an effort to retain all of the matching information.

[default = true]

**-minProbability**

Reject receptors that have lower than this probability of finding a correct pose under 2.0 Ångström after optimization.

[default = 0.333333]

**-minInitialProbability**

Reject receptors that have lower than this probability of finding a correct pose under 2.0 Ångström before optimization.

[default = 0.05]

**-minInitialTanimotoCombo**

Reject receptors that have lower than this initial rigid TanimotoCombo before optimization.

[default = 0.8]

**-minTanimotoCombo**

Set the minimum *TanimotoCombo* for outputting poses. This is currently set to 1.2 which has been empirically determined to generate good poses on average.

[default = 1.0]

**-optUseProtein**

Perform a final optimization with the protein but preserving the interactions associated with the atoms involved in the TanimotoCombo score. This allows the “spinach” and hydrogens an attempt to unclash with the protein if necessary.

[default = true]

**-param <filename>**

Defines the control parameter file. This file can contain a collection of parameters which can be used instead of writing each parameter to the command-line. In addition, the parameter file written by any POSIT run (see *-prefix* below), can be used with the *-param* flag in subsequent POSIT executions. Any command given explicitly on the command line will supersede any command found in a file specified with the *-param* parameter.

**-prefix**

The prefix controls the name of the report and log files.

The report file is written as a tab separated file, with the ligand number and name as well as the SDData fields previously specified.

[default = posit]

**-probabilityStop**

If probability cutoff is set and selecting by **finalTanimotoCombo**, then stop when the probability of a correct pose is greater than the one specified.

For example:

```
prompt> posit -selectReceptorBy finalTanimotoCombo -probabilityStop 0.7
```

Will stop when a probability of 0.7 or higher is found of finding a correct pose < 2.0 Ångströms.

When not selecting by a probability measure (**probability**, **finalTanimotoCombo**) **-probabilityStop** has no affect.

[default = 0.9]

**-scatter**

Instead of using the molecule’s shape (as defined by ShapeTanimoto) for optimizing the fit ligand, use the scattering function of the bound ligand (using the bound ligand’s bfactors and occupancies if applicable)

[default = false]

**-selectReceptorBy**

POSIT provides several methods of initial receptor selection.

Receptor Search Method	Description
probability	A single receptor is chosen with the highest probability of finding the correct pose. The probability is determined by a combination of 2D and initial 3D similarities.
finalTanimotoCombo	Fit to all receptors and take the TanimotoCombo of the final optimized pose. Use this for an exhaustive search.
pathsim	Fit to the receptor that has the highest path fingerprint similarity
maccessim	Fit to the receptor that has the highest MACCS166 fingerprint similarity
initialTanimotoCombo	Fit to the receptor that has the highest initial TanimotoCombo (3D) similarity

[default = probability]

**-strain**

The maximum amount of strain (in kcal) to accept when fitting to the pose-prediction target.

[default = 10.0]

**-verbose**

Give verbose output to console instead of simple progress.

[default = false]

## 8.2 Example Commands

The example commands in this section can be run with files found in the `example/posit/1.0.1` directory under the top-level installation directory.

POSIT always requires a bound-ligand in a receptor to fit against and an output file. The ligand (or ligands) to fit is specified with the `-in` option and the structures to fit are specified with the `-receptors`. Multiple receptors can be input at a time, in fact this is the preferred mode of running POSIT.

### 8.2.1 Basic Usage

The basic form of running `posit` is to fit against multiple receptors. The best fit receptor will be chosen for the final pose. Assuming that you have made receptors and combined them as shown above:

```
prompt> posit -receptors renin/receptors/*.oeb.gz renin/merged/*.oeb.gz -lig renin/all.smi \
-out results.sdf
```

This will output a lot of information, a portion of which is as follows:

```
Writing log file to: posit.log
Writing report file to: posit.rpt
Read receptor: RENIN from: renin/receptors/2IKO.rec.oeb.gz
Read receptor: RENIN from: renin/receptors/2IKU_a.rec.oeb.gz
Read receptor: RENIN from: renin/receptors/2IKU_b.rec.oeb.gz
Read receptor: RENIN from: renin/receptors/2IL2_a.rec.oeb.gz
Read receptor: RENIN from: renin/receptors/2IL2_b.rec.oeb.gz
Read receptor: RENIN from: renin/receptors/2IL2_c.rec.oeb.gz
Read receptor: RENIN from: renin/merged/2IKO.rec._merge_2IKU_a.rec.oeb.gz
Read receptor: RENIN from: renin/merged/2IKU_a.rec._merge_2IKO.rec.oeb.gz
```

All results are written to the specified output file. To write out all matching complexes, use the **-outcomplexes** or **-outreceptors** flags.

Remember that:

- **-outcomplexes** - writes a complex out for each matched receptor
- **-outreceptors** - writes a separate file for each matched receptor with all matching ligands. This file is the most compact.

The remaining portion of the output shows the specific information in choosing the appropriate receptor and showing the final optimization with respect to the protein. Note that different matching techniques may show different information.

The first ligand is analyzed as follows and results in a pose and a suggested alternate conformation:

```
Ligand 1 (ren1): Posed to Receptor: renin/receptors/2IKO.rec.oeb.gz (RENIN)
    Final Probability: 0.99 TanimotoCombo: 1.82
```

The next two are interesting in that they show common stereochemistry issues that are caused by time-averaged structures, fortunately, POSIT also shows how to correct these issues:

```
Warning: Ligand: 2 only has nitrogen centered chirality,
    (use -ignoreNitrogenStereo option to ignore nitrogen chirality.)
Warning: Ligand: 2 is missing stereo, rejecting (use -expandMissingStereo flag to correct.)
Ligand 2 (ren2) Rejected (missing stereo)
Warning: Ligand: 3 is missing stereo, rejecting (use -expandMissingStereo flag to correct.)
Ligand 3 (ren3) Rejected (missing stereo)
```

Any predicted ligand that has a good probability but clashes with the protein will either be ignored or, if one of the **-clash** outputs is specified on the command line, it will be output to the specified file.

To write the output to stdout, only use the file format extension in the **-out** parameter.

```
prompt> posit -receptors renin/receptors/*.oeb.gz renin/merged/*.oeb.gz -lig renin/all.smi \
    -out .sdf
```

---

**Note:** Only **-out** can be written to the output stream.

---

## 8.2.2 Ignoring Nitrogen Stereo

```
prompt> posit -receptors renin/receptors/*.oeb.gz renin/merged/*.oeb.gz -lig all.smi \
    -out results.sdf -ignoreNitrogenStereo
```

Now, Ligand 2 from the above example, no longer cares about stereo. However, you will error messages like the following:

```
Warning: Input stereochemistry different from output after fitting
in: CCc1c(c(nc([nH+]1)N)N)c2ccc3c(c2)N(CCC3)CCOC
out: CCc1c(c(nc([nH+]1)N)N)c2ccc3c(c2)[N@](CCC3)CCOC
Warning: Input stereochemistry different from output after fitting
in: CCc1c(c(nc([nH+]1)N)N)c2ccc3c(c2)N(CCC3)CCOC
out: CCc1c(c(nc([nH+]1)N)N)c2ccc3c(c2)[N@@](CCC3)CCOC
```

These simply serve as a warning that stereochemistry was added or modified during the fitting/optimization process.

## 8.2.3 Advanced Usage

Adjusting the POSIT algorithm requires an understanding of which parameters are used at a given step in the process. Figure *Posit Parameters* shows a basic overview of how the parameters control the algorithm.

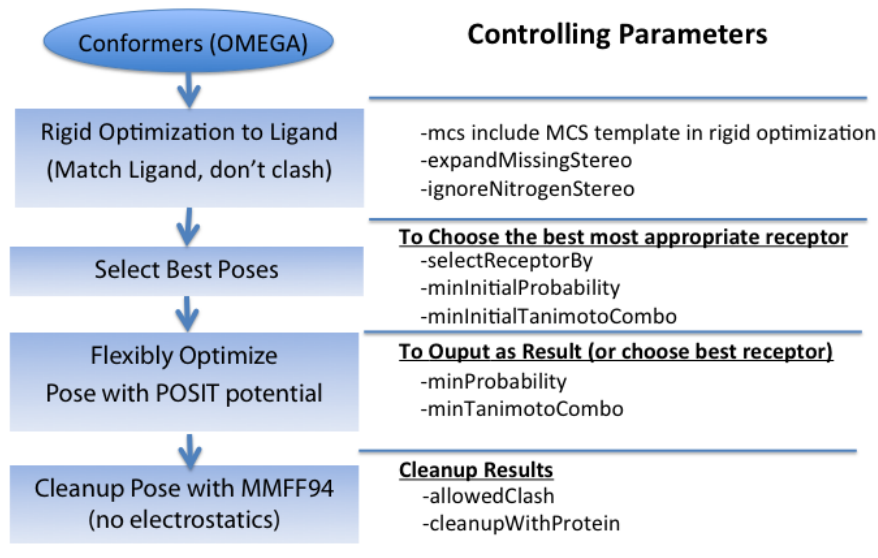


Figure 8.1: **POSIT Parameters** a basic workflow of when the command line parameters are used.

To run POSIT on absolutely everything and not worry about clashes or probabilities:

```
prompt> posit -receptors renin/receptors/*.oeb.gz -lig renin/ren1.smi \
-out results.oeb -minInitialTanimotoCombo 0.0 -minTanimotoCombo 0.0 -minProbability 0.0 \
-allowedClashes allclashes \
-selectBy finalTanimotoCombo
```

This runs posit on every receptor, does not prune overlays based on clashing proteins and picks the best fit receptor based only on the final *TanimotoCombo* overlap with the bound ligand.

# RELEASE NOTES

## 9.1 POSIT v1.0.1

*February 2012*

### 9.1.1 Major Bug Fixes

- In some rare cases the `-mcs` flag could cause a portion of the posed ligand to be fixed in space while the rest was optimized correctly, this causes very bad geometry molecules that can score very well. POSIT has been fixed to optimize the whole molecule in such cases.
- Drastically reduced the memory used when matching to multiple receptors. It is recommended to use 64 bit systems, but many more receptors can be analyzed on all systems.

### 9.1.2 Minor Bug Fixes

- The version numbers for `make_pose_receptor` and `combine_receptors` were being incorrectly output in the Open-Eye Banner, this has been updated to reflect the correct POSIT version.

### 9.1.3 Minor Features

- `-outputall` flag has been added to send all computed output to the file specified by `-out`, `-outcomplexes` or `-outreceptors`. It is a shortcut for:

```
-minInitialTanimotoCombo 0.0 -minTanimotoCombo 0.0 -minProbability 0.0 \  
-allowedClashes allclashes
```

- if any of the input receptors has a clashing ligand, a warning will be output and POSIT will be set to accept clashes of the same severity as the receptor ligand.
- **POSIT can now stream output specified by the `-out` switch to stdout using the** command line switch `-out .sdf` or `-out .oeb`

## 9.2 POSIT v1.0.0

*November 2011*

First release of POSIT.

Traditional structure based pose-prediction has not been very accurate in reproducing crystallographic poses. This can be rectified by using all of the information present in a crystal structure - both ligand and protein structure. POSIT is a flexible docking technique that uses both the known protein and ligand structure to predict poses. Furthermore, using this information generates a probability that the predicted pose is indeed correct. This has far reaching implications for real world lead optimization scenarios including measuring confidence but also the ability to select the existing crystal structure that best predicts the docked pose for a given molecule.

### **9.2.1 Features**

- Probability based pose-prediction
- Performs better than structure-based pose-prediction for poses similar to known bound ligands.
- All-atom optimization of poses based on known bound-ligands and ligand-protein interactions.
- Fast detection of best known receptor (complex) to determine pose.

# BIBLIOGRAPHY

- [Bemis-2004] B.J. Hare, W.P. Walters, P.R. Caron and G.W. Bemis, **CORES: An Automated Method for Generating Three-Dimensional Models of Protein/Ligand Complexes**, *Journal of Medicinal Chemistry*, Vol. 47 (19), pp. 4731-4740, **2004**
- [Halgren-I-1996] T.A. Halgren, **Merck Molecular Force Field: I. Basis, Form, Scope, Parameterization and Performance of MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 490-519, **1996**
- [Halgren-II-1996] T.A. Halgren, **Merck Molecular Force Field: II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 520-552, **1996**
- [Halgren-III-1996] T.A. Halgren, **Merck Molecular Force Field: III. Molecular Geometries and Vibrational Frequencies**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 553-586, **1996**
- [Halgren-IV-1996] T.A. Halgren and R.B. Nachbar, **Merck Molecular Force Field: IV. Conformational Energies and Geometries for MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 687-615, **1996**
- [Halgren-V-1996] T.A. Halgren, **Merck Molecular Force Field: V. Extension of MMFF94 using Experimental Data, Additional Computational Data and Empirical Rules**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 616-641, **1996**
- [Halgren-VI-1999] T.A. Halgren, **MMFF VI. MMFF94s Option for Energy Minimization Studies**, *Journal of Computational Chemistry*, Vol. 20, pp. 720-729, **1999**
- [Halgren-VII-1999] T.A. Halgren, **MMFF VII. Characterization of MMFF94, MMFF94s and Other Widely Available Force Fields for Conformational Energies and for Intermolecular Interaction Energies and Geometries**, *Journal of Computational Chemistry*, Vol. 20, pp. 730-748, **1999**
- [Hawkins-2010] P.C.D. Hawkins, A.G. Skillman, G.L. Warren, B.A. Ellingson and M.T. Stahl, **Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database**, *Journal of Chemical Information and Modeling*, Vol. 50 (4), pp. 572-584, **2010**
- [Martinelli-2010] T. Tuccinardi, A. Giordano and A. Martinelli, **Protein Kinases: Docking and Homology Modelling Reliability** *Journal of Chemical Information and Modeling*, Vol. 50 (8), pp. 1432-1441, **2010**
- [MillsDean-1996] J.E.J. Mills and P.M. Dean, **Three-Dimensional Hydrogen-Bond Geometry and Probability Information from a Crystal Survey**, *Journal of Computer-Aided Molecular Design*, Vol. 10, pp. 607-622, **1996**.
- [PDB] *Protein Databank*, (online: <http://www.rcsb.org/>)
- [Perola-2004] E. Perola and P.S. Charifson, **Conformational analysis of drug-like molecules bound to proteins: an extensive study of ligand reorganization upon binding**, *J. Med. Chem.*, Vol. 47, pp. 2499-2510, **2004**

- [Vieth-2004] J.A. Erickson, M. Jalaie, D.H. Robertson, R.A. Lewis and M. Vieth **Lessons in Molecular Recognition: The Effects of Ligand and Protein Flexibility on Molecular Docking Accuracy**. *Journal of Medicinal Chemistry*, Vol. 47 (1), pp. 45-55, **2004**
- [Wlodek-2006] S. Wlodek, A.G. Skillman and A. Nicholls, **Automated Ligand Placement and Refinement with a Combined Force Field and Shape Potential**, *Acta Crystallographica Section D*, Vol. 62, pp. 741-749, **2006**

# INDEX

## Symbols

- allowedClashes
  - combine\_receptors command line option, 26
  - make\_pose\_receptor command line option, 22
  - posit command line option, 32
- alternatePoseRMSD
  - posit command line option, 32
- alternatePoses
  - posit command line option, 32
- auto
  - make\_pose\_receptor command line option, 22
- clash <filename>
  - posit command line option, 31
- clashcomplexes <filename>
  - posit command line option, 31
- clashreceptors <filename>
  - posit command line option, 31
- expandMissingStereo
  - posit command line option, 32
- forcePlanarAromatic
  - posit command line option, 32
- ignoreNitrogenStereo
  - posit command line option, 32
- in <filename>
  - posit command line option, 30
- ligand
  - make\_pose\_receptor command line option, 21
- mcs
  - posit command line option, 32
- minInitialProbability
  - posit command line option, 33
- minInitialTanimotoCombo
  - posit command line option, 33
- minProbability
  - posit command line option, 32
- minTanimotoCombo
  - posit command line option, 33
- optUseProtein
  - posit command line option, 33
- out
  - make\_pose\_receptor command line option, 21
- out <filename>
  - posit command line option, 31
- outcomplexes <filename>
  - posit command line option, 31
- outputall
  - posit command line option, 31
- outputdir
  - combine\_receptors command line option, 27
- outreceptors <filename>
  - posit command line option, 31
- param <filename>
  - combine\_receptors command line option, 27
  - make\_pose\_receptor command line option, 22
  - posit command line option, 33
- prealigned
  - combine\_receptors command line option, 27
- prefix
  - combine\_receptors command line option, 27
  - make\_pose\_receptor command line option, 22
  - posit command line option, 33
- probabilityStop
  - posit command line option, 33
- protein
  - make\_pose\_receptor command line option, 21
- receptors <filenames>
  - combine\_receptors command line option, 26
  - posit command line option, 30
- residues
  - make\_pose\_receptor command line option, 22
- scatter
  - posit command line option, 33
- selectReceptorBy
  - posit command line option, 34
- strain
  - posit command line option, 34
- verbose
  - combine\_receptors command line option, 27
  - posit command line option, 34

## A

APPNAME\_OE\_ARCH, 4

## C

combine\_receptors command line option

- allowedClashes, 26
- outputdir, 27
- param <filename>, 27
- prealigned, 27
- prefix, 27
- receptors <filenames>, 26
- verbose, 27

- outreceptors <filename>, 31
- param <filename>, 33
- prefix, 33
- probabilityStop, 33
- receptors <filenames>, 30
- scatter, 33
- selectReceptorBy, 34
- strain, 34
- verbose, 34

## E

environment variable

- APPNAME\_OE\_ARCH, 4
- OE\_ARCH, 4
- OE\_LICENSE, 3
- PATH, 4, 5

## M

make\_pose\_receptor command line option

- allowedClashes, 22
- auto, 22
- ligand, 21
- out, 21
- param <filename>, 22
- prefix, 22
- protein, 21
- residues, 22

## O

- OE\_ARCH, 4
- OE\_LICENSE, 3

## P

PATH, 4, 5

posit command line option

- allowedClashes, 32
- alternatePoseRMSD, 32
- alternatePoses, 32
- clash <filename>, 31
- clashcomplexes <filename>, 31
- clashreceptors <filename>, 31
- expandMissingStereo, 32
- forcePlanarAromatic, 32
- ignoreNitrogenStereo, 32
- in <filename>, 30
- mcs, 32
- minInitialProbability, 33
- minInitialTanimotoCombo, 33
- minProbability, 32
- minTanimotoCombo, 33
- optUseProtein, 33
- out <filename>, 31
- outcomplexes <filename>, 31
- outputall, 31