DSTF
Dipartimento di Scienza
e Tecnologia del Farmaco
UNIVERSITÀ DI TORINO

UNIVERSITÀ
DEGLI STUDI
DI TORINO
ALMA UNIVERSITAS
TAURINENSIS

Bringing MMFF to the RDKit

Paolo Tosco

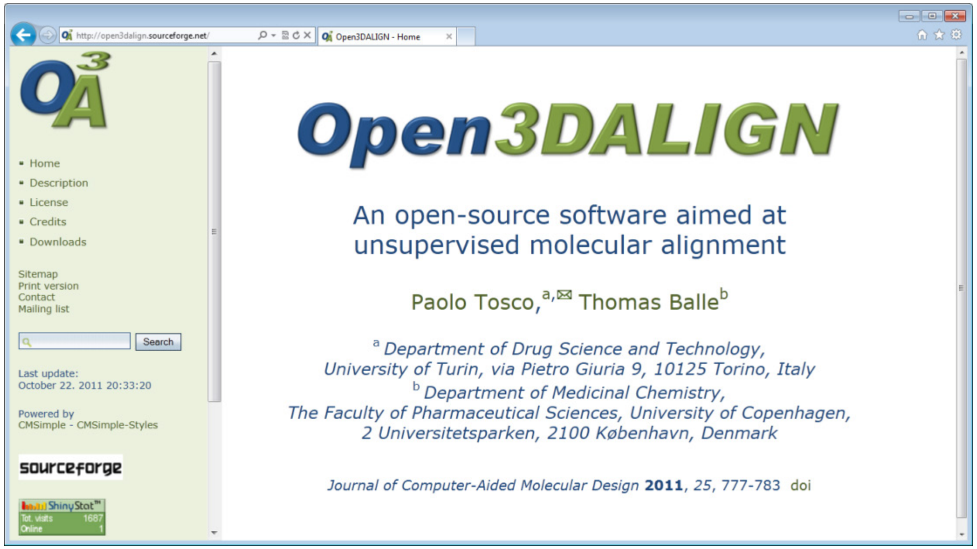
RDKit User Group Meeting 2013

October 2-4, European Bioinformatics Institute, Hinxton, UK

Topics

- The MMFF implementation
- UFF gains something, too
- MMFF applications

Open3DTOOLS: Open3DQSAR & Open3DALIGN



Open3DALIGN

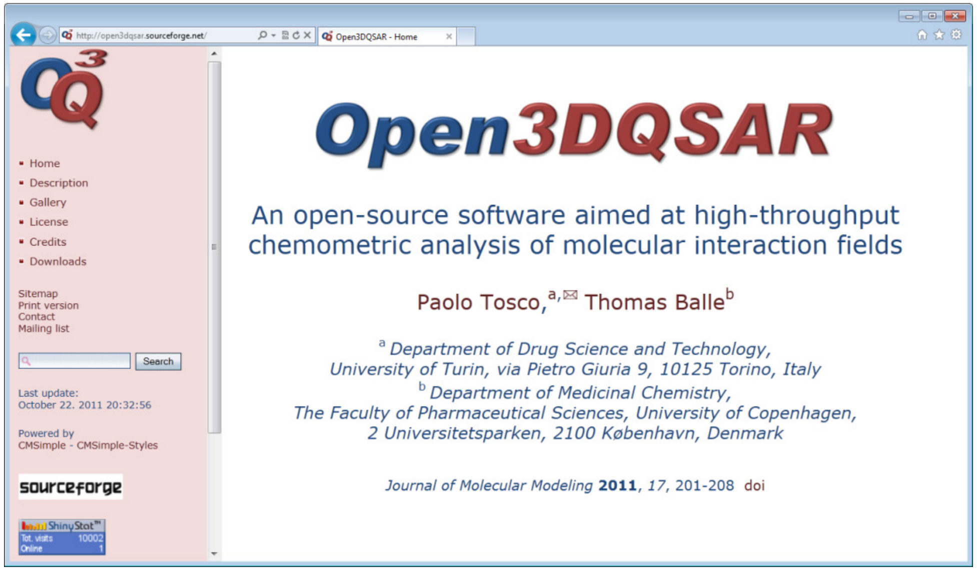
An open-source software aimed at
unsupervised molecular alignment

Paolo Tosco,[a,✉] Thomas Balle[b]

[a] *Department of Drug Science and Technology,*
University of Turin, via Pietro Giuria 9, 10125 Torino, Italy
[b] *Department of Medicinal Chemistry,*
The Faculty of Pharmaceutical Sciences, University of Copenhagen,
2 Universitetsparken, 2100 København, Denmark

Journal of Computer-Aided Molecular Design **2011**, *25*, 777-783 doi

- **Open3DALIGN**
  - conformational search
  - molecular alignment
    - single/multiple conformations
    - atom-based/pharmacophore-based/mixed alignment
    - single-run/iterative



Open3DQSAR

An open-source software aimed at high-throughput
chemometric analysis of molecular interaction fields

Paolo Tosco,[a,✉] Thomas Balle[b]

[a] *Department of Drug Science and Technology,*
University of Turin, via Pietro Giuria 9, 10125 Torino, Italy
[b] *Department of Medicinal Chemistry,*
The Faculty of Pharmaceutical Sciences, University of Copenhagen,
2 Universitetsparken, 2100 København, Denmark

Journal of Molecular Modeling **2011**, *17*, 201-208 doi

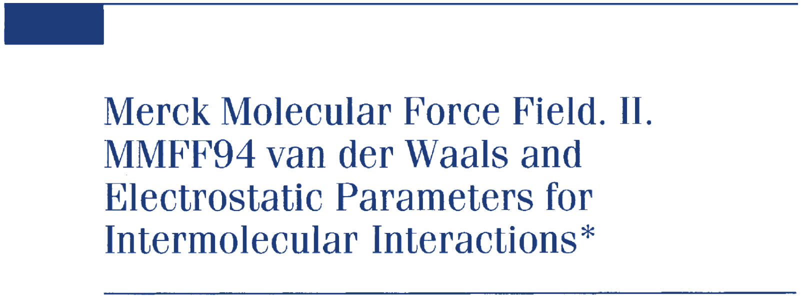
- **Open3DQSAR**
  - MIF computation
  - PLS model building and validation
  - variable selection

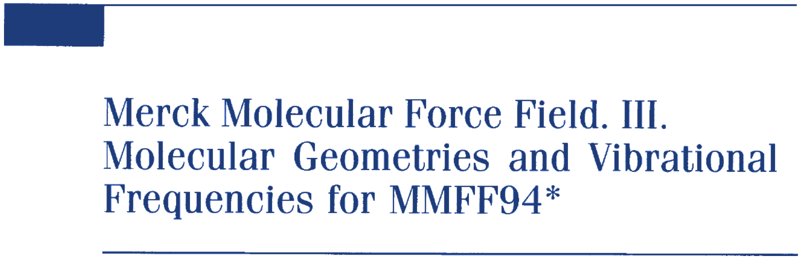
Why bringing MMFF to the RDKit?

- Preliminary step before implementing **Open3DQSAR** and **Open3DALIGN** functionality
- In principle, only MMFF atom types and charges are needed for that purpose…
- …but one shouldn't be lazy, right? ;-)

MMFF94 documentation

Merck Molecular Force Field. I. Basis, Form, Scope, Parameterization, and Performance of MMFF94*


Merck Molecular Force Field. II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions*

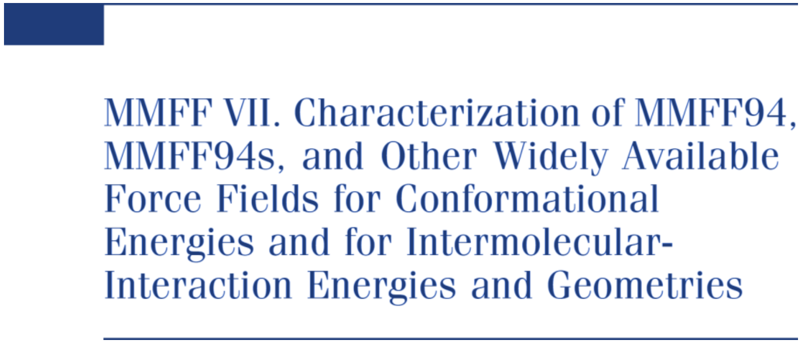

Merck Molecular Force Field. III. Molecular Geometries and Vibrational Frequencies for MMFF94*

MMFF94(s) is fully documented by a series of 7 papers published by Halgren and Nachbar (Merck) on *J. Comp. Chem.* between 1995 and 1998

Merck Molecular Force Field. IV. Conformational Energies and Geometries for MMFF94*

Merck Molecular Force Field. V. Extension of MMFF94 Using Experimental Data, Additional Computational Data, and Empirical Rules*

MMFF VI. MMFF94s Option for Energy Minimization Studies


MMFF VII. Characterization of MMFF94, MMFF94s, and Other Widely Available Force Fields for Conformational Energies and for Intermolecular-Interaction Energies and Geometries

MMFF94 energy expression: 7 terms

$E_{MMFF} = \sum EB_{ij}$ bond stretching

$$EB_{ij} = 143.9325\frac{kb_{ij}}{2}\Delta r_{ij}^2\left(1 + cs\Delta r_{ij} + \frac{7}{12}cs^2\Delta r_{ij}^2\right)$$

$+\sum EA_{ijk}$ angle bending

$$EA_{ijk} = 0.043844\frac{ka_{IJK}}{2}\Delta\vartheta_{ijk}^2\left(1 + cb\Delta\vartheta_{ijk}\right)$$

$+\sum EBA_{ijk}$ stretch-bend

$$EBA_{ijk} = 2.51210\left(kba_{IJK}\Delta r_{ij} + kba_{KJI}\Delta r_{kj}\right)\Delta\vartheta_{ijk}$$

$+\sum EOOP_{ijk;l}$ out-of-plane bending

$$EOOP_{ijk;l} = 0.043844\frac{koop_{IJK:L}}{2}\chi_{ijk;l}^2$$

$+\sum ET_{ijkl}$ torsion

$$ET_{ijkl} = 0.5\left[V_1\left(1 + \cos\phi\right) + V_2\left(1 - \cos 2\phi\right) + V_3\left(1 + \cos 3\phi\right)\right]$$

$+\sum EvdW_{ij}$ van der Waals

$$EvdW_{ij} = \varepsilon_{IJ}\left(\frac{1.07R_{IJ}^*}{R_{ij} + 0.07R_{IJ}^*}\right)^7\left(\frac{1.12R_{IJ}^{*7}}{R_{ij}^7 + 0.12R_{IJ}^{*7}} - 2\right)$$

$+\sum EQ_{ij}$ electrostatic

$$EQ_{ij} = 332.0716\frac{q_i q_j}{D\left(R_{ij} + \delta\right)^n}$$

CR	1	ALKYL CARBON, SP3
C=C	2	VINYLIC CARBON, SP2
CSP2	2	GENERIC SP2 CARBON
C=O	3	GENERAL CARBONYL CARBON
C=N	3	SP2 CARBON IN C=N
CGD	3	GUANIDINE CARBON, DOUBLY BONDED TO N
C=OR	3	KETONE OR ALDEHYDE CARBONYL CARBON
C=ON	3	AMIDE CARBONYL CARBON
CONN	3	UREA CARBONYL CARBON
COO	3	CARBOXYLIC ACID OR ESTER CARBONYL CARBON
COON	3	CARBAMATE CARBONYL CARBON
COOO	3	C ARBONIC ACID OR ESTER CARBONYL CARBON
C=OS	3	THIOESTER CARBONYL CARBON, DOUBLE BONDED TO O
C=S	3	THIOESTER CARBON, DOUBLY BONDED TO S
C=SN	3	THIOAMIDE, CARBON, DOUBLY BONDED TO S
CSO2	3	CARBON IN >C=SO2
CS=O	3	CARBON IN >C=S=O (SULFINYL GROUP)
CSS	3	THIOCARBOXYLIC ACID OR ESTER CARBONYL CARBON
C=P	3	CARBON DOUBLE BONDED TO PHOSPHOROUS
CSP	4	ACETYLENIC CARBON
=C=	4	ALLENIC CARBON
HC	5	H ATTACHED TO C
HSI	5	H ATTACHED TO SI
OR	6	ALCOHOL OR ETHER OXYGEN
OC=O	6	ESTER OR CARBOXYLIC ACID -O-
OC=C	6	ENOLIC OR PHENOLIC OXYGEN
OC=N	6	DIVALENT OXYGEN
OC=S	6	THIOESTER OR THIOACID -O-
ONO2	6	DIVALENT NITRATE "ETHER" OXYGEN
ON=O	6	DIVALENT NITRITE "ETHER" OXYGEN
OSO3	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OSO2	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OSO	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OS=O	6	DIVALENT OXYGEN ATTACHED TO SULFOXIDE SULFUR
-OS	6	GENERAL DIVALENT OX ATTACHED TO S
OPO3	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
OPO2	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
OPO	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
-OP	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
-O-	6	GENERAL DIVALENT O
O=C	7	GENERAL C=O
O=CN	7	CARBONYL OXYGEN, AMIDES
O=CR	7	CARBONYL OXYGEN, ALDEHYDES AND KETONES
O=CO	7	CARBONYL OXYGEN, CARBOXYLIC ACIDS AND ESTERS
O=N	7	NITROSO OXYGEN
O=S	7	O=S IN SULFOXIDES
O=S=	7	O=S ON SULFUR DOUBLY BONDED TO, E.G., CARBON
NR	8	NITROGEN IN ALIPHATIC AMINES
N=C	9	NITROGEN IN IMINES
N=N	9	NITROGEN IN AZO COMPOUNDS
NC=O	10	NITROGEN IN AMIDES
NC=S	10	NITROGEN IN N-C=S, THIOAMIDE
NN=C	10	NITROGEN IN N-N=C
NN=N	10	NITROGEN IN N-N=N
F	11	FLUORINE
CL	12	CHLORINE
BR	13	BROMINE
I	14	IODINE
S	15	SULFUR IN THIOETHERS AND MERCAPTANS
S=C	16	TERMINAL SULFUR DOUBLY BONDED TO CARBON
S=O	17	SULFUR IN SULFOXIDES
>S=N	17	SULFUR, TRICOORD, DOUBLY BONDED TO N
SO2	18	SULFUR IN SULFONES
SO2N	18	SULFUR IN SULFONAMIDES
SO3	18	SULFONATE SULFUR
SO4	18	SULFATE SULFUR
=SO2	18	SULFONE SULPHER DOUBLY BONDED TO CARBON
SNO	18	SULFUR IN NITROGEN ANALOG OF A SULFONE
SI	19	SILICON
CR4R	20	CARBON IN 4-MEMBERED RINGS
HOR	21	HYDROGEN IN ALCOHOLS
HO	21	GENERAL H ON OXYGEN
HOM	21	HYDROGEN IN HYDROXIDE ANION
CR3R	22	CARBON IN A 3-MEMBERED RING
HNR	23	H-N(SP3)
H3N	23	H-N(SP3), AMMONIA
HPYL	23	H-N IN PYRROLE
HNOX	23	H-N IN IN A N-OXIDE
HNM	23	H ON DICOORD, NEGATIVELY CHARGED NITROGEN
HN	23	GENERAL H ON NITROGEN
HOCO	24	H-O IN CARBOXYLIC ACIDS
HOP	24	HYDROGEN ON OXYGEN ATTACHED TO PHOSPHOROUS
PO4	25	PHOSPHOROUS IN PHOSPHATES AND PHOSPHODIESTERS
PO3	25	TETRACOORDINATE P WITH THREE ATTACHED OXYGENS
PO2	25	TETRACOORDINATE P WITH TWO ATTACHED OXYGENS
PO	25	TETRACOORDINATE P WITH ONE ATTACHED OXYGEN
PTET	25	GENERAL TETRACOORDINATE PHOSPHORUS
P	26	TRICOORDINATE P, AS IN PHOSPHINES
HN=N	27	AZO HYDROGEN
HN=C	27	IMINE HYDROGEN
HNCO	28	AMIDE HYDROGEN
HNCS	28	THIOAMIDE HYDROGEN
HNCC	28	H-N IN ENAMINES
HNCN	28	H-N IN H-N-C=N
HNNC	28	H-N IN H-N-N=C
HNNN	28	H-N IN H-N-N=N
HNSO	28	H-N IN SULFONAMIDE
HNPO	28	H-N IN PHOSPHONAMIDE
HNC%	28	HYDROGEN ON N ATTACHED TO TRIPLY BONDED CARBON
HSP2	28	GENERAL H ON SP2 NITROGEN
HOCC	29	H-O IN ENOLS AND PHENOLS
HOCN	29	H-O IN HO-C=N
CE4R	30	OLEFINIC CARBON IN 4-MEMBERED RINGS
HOH	31	HYDROGEN IN H2O
O2CM	32	OXYGEN IN CARBOXYLATE ANION
OXN	32	N-OXIDE OXYGEN
O2N	32	NITRO OXYGEN
O2NO	32	NITRO-GROUP OXYGEN IN NITRATE
O3N	32	NITRATE ANION OXYGEN
O-S	32	SINGLE TERMINAL OXYGEN ON TETRACOORD SULFUR
O2S	32	TERMINAL O-S IN SULFONES AND SULFONAMIDES
O3S	32	TERMINAL O IN SULFONATES
O4S	32	TERMINAL O IN SO4(-3)
OSMS	32	TERM O IN THIOSULFINATE ANION - FORMAL CHARGE=-0.5
OP	32	TERMINAL O IN PHOSPHOXIDES
O2P	32	TERMINAL O IN PHOSPHINATES
O3P	32	TERMINAL OXYGEN IN PHOSPHONATES
O4P	32	TERMINAL OXYGEN IN PHOSPHATES AND PHOSPHODIESTERS
O4CL	32	OXYGEN IN CLO4(-) ANION - FORMAL CHARGE=-0.25
HOS	33	H ON OXYGEN ATTACHED TO SULFUR
NR+	34	QUATERNARY NITROGEN, SP3, POSITIVELY CHARGED
OM	35	ALKOXIDE OXYGEN, NEGATIVELY CHARGED
OM2	35	OXIDE OXYGEN ON SP2 CARBON, NEGATIVELY CHARGED
HNR+	36	H ON QUATERNARY NITROGEN
HIM+	36	H ON IMIDAZOLIUM-TYPE NITROGEN
HPD+	36	H ON PROTONATED PYRIDINE NITROGEN
HNN+	36	H ON AMIDINIUM-TYPE NITROGEN
HNC+	36	H ON PROTONATED IMINE NITROGEN
HGD+	36	H ON GUANIDINIUM-TYPE NITROGEN
HN5+	36	H ON N5+, N5A+ OR N5B+
CB	37	CARBON AS IN BENZENE, PYRROLE
NPYD	38	NITROGEN, AS IN PYRIDINE
NPYL	39	NITROGEN, AS IN PYRROLE
NC=C	40	NITROGEN ON N-C=C
NC=N	40	NITROGEN IN N-C=N
NC=P	40	NITROGEN IN N-C=P
NC%C	40	NITROGEN ATTACHED TO C-C TRIPLE BOND
CO2M	41	CARBOXYLATE ANION CARBON
CS2M	41	CARBON IN THIOCARBOXYLATE ANION
NSP	42	NITROGEN, TRIPLE BONDED
NSO2	43	NITROGEN IN SULFONAMIDES
NSO3	43	NITROGEN IN SULFONAMIDES, THREE O'S ON S
NPO2	43	NITROGEN IN PHOSPHONAMIDES
NPO3	43	NITROGEN IN PHOSPHONAMIDES, THREE O'S ON P
NC%N	43	NITROGEN ATTACHED TO CYANO GROUP
STHI	44	SULFUR AS IN THIOPHENE
NO2	45	NITRO GROUP NITROGEN
NO3	45	NITRATE GROUP NITROGEN
N=O	46	NITROSO NITROGEN
NAZT	47	TERMINAL NITROGEN IN AZIDO OR DIAZO GROUP
NSO	48	DIVALENT NITROGEN REPLACING MONOVALENT O IN SO2 GROUP
O+	49	POSITIVELY CHARGED OXONIUM (TRICOORDINATE) OXYGEN
HO+	50	HYDROGEN ON O+ OXYGEN
O=+	51	POSITIVELY CHARGED OXENIUM (DICOORDINATE) OXYGEN
HO=+	52	HYDROGEN ON OXENIUM OXYGEN
=N=	53	NITROGEN IN C=N=N OR -N=N=N
N+=C	54	POSITIVELY CHARGED IMINIUM NITROGEN
N+=N	54	POSITIVELY CHARGED NITROGEN DOUBLE-BONDED TO N
NCN+	55	N IN +N=C-N RESONANCE STRUCTURES - FORMAL CHARGE=1/2
NGD+	56	GUANIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1/3
CGD+	57	GUANIDINIUM CARBON
CNN+	57	C IN +N=C-N RESONANCE STRUCTURES
NPD+	58	PYRIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1
OFUR	59	AROMATIC OXYGEN AS IN FURAN
C%	60	ISONITRILE CARBON
NR%	61	ISONITRILE NITROGEN [FC = 0] OR DIAZO NITROGEN [FC=1]
NM	62	DEPROTONATED SULFONAMIDE N-; FORMAL CHARGE=-1
C5A	63	ALPHA CARBON IN 5-MEMBERED HETEROAROMATIC RING
C5B	64	BETA CARBON IN 5-MEMBERED HETEROAROMATIC RING
N5A	65	ALPHA AROM HETEROCYCLIC 5-RING NITROGEN
N5B	66	BETA AROM HETEROCYCLIC 5-RING NITROGEN
N2OX	67	SP2-HYDRIDIZED N-OXIDE NITROGEN
N3OX	68	SP3-HYDRIDIZED N-OXIDE NITROGEN
NPOX	69	PYRIDINE N-OXIDE NITROGEN
OH2	70	OXYGEN ON WATER
HS	71	H ATTACHED TO DIVALENT, DICOORDINATE S
HS=N	71	H ATTACHED TO TETRAVALENT, TRICOODR S DBL BONDED TO N
HP	71	H ATTACHED TO TRI- OR TETRACOORDINATE PHOSPHORUS
S-P	72	TERMINAL SULFUR BONDED TO PHOSPHORUS
S2CM	72	TERMINAL SULFUR IN THIOCARBOXYLATE ANION
SM	72	TERMINAL SULFUR - FORMAL CHARGE=-1
SSMO	72	TERMINAL SULFUR IN THIOSULFINATE GROUP
SO2M	73	SULFUR IN NEGATIVELY CHARGED SULFINATE GROUP
SSOM	73	TRICOORD SULFUR IN THIOSULFINATE GROUP
=S=O	74	SULFINYL SULFUR, EG. IN C=S=O
-P=C	75	PHOSPHOROUS DOUBLY BONDED TO CARBON
N5M	76	NEGATIVELY CHARGED N IN, E.G, TRI- OR TETRAZOLE ANION
CLO4	77	CHLORINE IN PERCHLORATE ANION, CLO4(-)
C5	78	GENERAL CARBON IN 5-MEMBERED HETEROAROMATIC RING
N5	79	GENERAL NITROGEN IN 5-MEMBERED HETEROCYCLIC RING
CIM+	80	C IN N-C-N IN IMIDAZOLIUM ION
NIM+	81	IMIDAZOLIUM-TYPE NITROGEN - FORMAL CHARGE=1/2
N5A+	81	POSITIVE N5A NITROGEN - FORMAL CHARGE=1
N5B+	81	POSITIVE N5B NITROGEN - FORMAL CHARGE=1
N5+	81	POSITIVE N5 NITROGEN - FORMAL CHARGE=1
N5AX	82	N-OXIDE NITROGEN IN 5-RING ALPHA POSITION
N5BX	82	N-OXIDE NITROGEN IN 5-RING BETA POSITION
N5OX	82	N-OXIDE NITROGEN IN GENERAL 5-RING POSITION
FE+2	87	IRON +2 CATION
FE+3	88	IROM +3 CATION
F-	89	FLUORIDE ANION
CL-	90	CHLORIDE ANION
BR-	91	BROMIDE ANION
LI+	92	LITHIUM CATION
NA+	93	SODIUM CATION
K+	94	POTASSIUM CATION
ZINC	95	DIPOSITIVE ZINC
ZN+2	95	DIPOSITIVE ZINC
CA+2	96	DIPOSITIVE CALCIUM
CU+1	97	MONOPOSITIVE COPPER
CU+2	98	DIPOSITIVE COPPER
MG+2	99	DIPOSITIVE MAGNESIUM CATION

212 (!) different atom types, grouped into 95 categories

CR	1	ALKYL CARBON, SP3
C=C	2	VINYLIC CARBON, SP2
CSP2	2	GENERIC SP2 CARBON
C=O	3	GENERAL CARBONYL CARBON
C=N	3	SP2 CARBON IN C=N
CGD	3	GUANIDINE CARBON, DOUBLY BONDED TO N
C=OR	3	KETONE OR ALDEHYDE CARBONYL CARBON
C=ON	3	AMIDE CARBONYL CARBON
CONN	3	UREA CARBONYL CARBON
COO	3	CARBOXYLIC ACID OR ESTER CARBONYL CARBON
COON	3	CARBAMATE CARBONYL CARBON
COOO	3	C ARBONIC ACID OR ESTER CARBONYL CARBON
C=OS	3	THIOESTER CARBONYL CARBON, DOUBLE BONDED TO O
C=S	3	THIOESTER CARBON, DOUBLY BONDED TO S
C=SN	3	THIOAMIDE, CARBON, DOUBLY BONDED TO S
CSO2	3	CARBON IN >C=SO2
CS=O	3	CARBON IN >C=S=O (SULFINYL GROUP)
CSS	3	THIOCARBOXYLIC ACID OR ESTER CARBONYL CARBON
C=P	3	CARBON DOUBLE BONDED TO PHOSPHOROUS
CSP	4	ACETYLENIC CARBON
=C=	4	ALLENIC CARBON
HC	5	H ATTACHED TO C
HSI	5	H ATTACHED TO SI
OR	6	ALCOHOL OR ETHER OXYGEN
OC=O	6	ESTER OR CARBOXYLIC ACID -O-
OC=C	6	ENOLIC OR PHENOLIC OXYGEN
OC=N	6	DIVALENT OXYGEN
OC=S	6	THIOESTER OR THIOACID -O-
ONO2	6	DIVALENT NITRATE "ETHER" OXYGEN
ON=O	6	DIVALENT NITRITE "ETHER" OXYGEN
OSO3	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OSO2	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OSO	6	DIVALENT OXYGEN ATTACHED TO SULFUR
OS=O	6	DIVALENT OXYGEN ATTACHED TO SULFOXIDE SULFUR
-OS	6	GENERAL DIVALENT OX ATTACHED TO S
OPO3	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
OPO2	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
OPO	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
-OP	6	DIVALENT OXYGEN ATTACHED TO PHOSPHOROUS
-O-	6	GENERAL DIVALENT O
O=C	7	GENERAL C=O
O=CN	7	CARBONYL OXYGEN, AMIDES
O=CR	7	CARBONYL OXYGEN, ALDEHYDES AND KETONES
O=CO	7	CARBONYL OXYGEN, CARBOXYLIC ACIDS AND ESTERS
O=N	7	NITROSO OXYGEN
O=S	7	O=S IN SULFOXIDES
O=S=	7	O=S ON SULFUR DOUBLY BONDED TO, E.G., CARBON
NR	8	NITROGEN IN ALIPHATIC AMINES
N=C	9	NITROGEN IN IMINES
N=N	9	NITROGEN IN AZO COMPOUNDS
NC=O	10	NITROGEN IN AMIDES
NC=S	10	NITROGEN IN N-C=S, THIOAMIDE
NN=C	10	NITROGEN IN N-N=C
NN=N	10	NITROGEN IN N-N=N
F	11	FLUORINE
CL	12	CHLORINE
BR	13	BROMINE
I	14	IODINE
S	15	SULFUR IN THIOETHERS AND MERCAPTANS
S=C	16	TERMINAL SULFUR DOUBLY BONDED TO CARBON
S=O	17	SULFUR IN SULFOXIDES
>S=N	17	SULFUR, TRICOORD, DOUBLY BONDED TO N
SO2	18	SULFUR IN SULFONES
SO2N	18	SULFUR IN SULFONAMIDES
SO3	18	SULFONATE SULFUR
SO4	18	SULFATE SULFUR
=SO2	18	SULFONE SULPHER DOUBLY BONDED TO CARBON
SNO	18	SULFUR IN NITROGEN ANALOG OF A SULFONE
SI	19	SILICON
CR4R	20	CARBON IN 4-MEMBERED RINGS
HOR	21	HYDROGEN IN ALCOHOLS
HO	21	GENERAL H ON OXYGEN
HOM	21	HYDROGEN IN HYDROXIDE ANION
CR3R	22	CARBON IN A 3-MEMBERED RING
HNR	23	H-N(SP3)
H3N	23	H-N(SP3), AMMONIA
HPYL	23	H-N IN PYRROLE
HNOX	23	H-N IN IN A N-OXIDE
HNM	23	H ON DICOORD, NEGATIVELY CHARGED NITROGEN
HN	23	GENERAL H ON NITROGEN
HOCO	24	H-O IN CARBOXYLIC ACIDS
HOP	24	HYDROGEN ON OXYGEN ATTACHED TO PHOSPHOROUS
PO4	25	PHOSPHOROUS IN PHOSPHATES AND PHOSPHODIESTERS
PO3	25	TETRACOORDINATE P WITH THREE ATTACHED OXYGENS
PO2	25	TETRACOORDINATE P WITH TWO ATTACHED OXYGENS
PO	25	TETRACOORDINATE P WITH ONE ATTACHED OXYGEN
PTET	25	GENERAL TETRACOORDINATE PHOSPHORUS
P	26	TRICOORDINATE P, AS IN PHOSPHINES
HN=N	27	AZO HYDROGEN
HN=C	27	IMINE HYDROGEN
HNCO	28	AMIDE HYDROGEN
HNCS	28	THIOAMIDE HYDROGEN
HNCC	28	H-N IN ENAMINES
HNCN	28	H-N IN H-N-C=N
HNNC	28	H-N IN H-N-N=C
HNNN	28	H-N IN H-N-N=N
HNSO	28	H-N IN SULFONAMIDE
HNPO	28	H-N IN PHOSPHONAMIDE
HNC%	28	HYDROGEN ON N ATTACHED TO TRIPLY BONDED CARBON
HSP2	28	GENERAL H ON SP2 NITROGEN
HOCC	29	H-O IN ENOLS AND PHENOLS
HOCN	29	H-O IN HO-C=N
CE4R	30	OLEFINIC CARBON IN 4-MEMBERED RINGS
HOH	31	HYDROGEN IN H2O
O2CM	32	OXYGEN IN CARBOXYLATE ANION
OXN	32	N-OXIDE OXYGEN
O2N	32	NITRO OXYGEN
O2NO	32	NITRO-GROUP OXYGEN IN NITRATE
O3N	32	NITRATE ANION OXYGEN
O-S	32	SINGLE TERMINAL OXYGEN ON TETRACOORD SULFUR
O2S	32	TERMINAL O-S IN SULFONES AND SULFONAMIDES
O3S	32	TERMINAL O IN SULFONATES
O4S	32	TERMINAL O IN SO4(-3)
OSMS	32	TERM O IN THIOSULFINATE ANION - FORMAL CHARGE=-0.5
OP	32	TERMINAL O IN PHOSPHOXIDES
O2P	32	TERMINAL O IN PHOSPHINATES
O3P	32	TERMINAL OXYGEN IN PHOSPHONATES
O4P	32	TERMINAL OXYGEN IN PHOSPHATES AND PHOSPHODIESTERS
O4CL	32	OXYGEN IN CLO4(-) ANION - FORMAL CHARGE=-0.25
HOS	33	H ON OXYGEN ATTACHED TO SULFUR
NR+	34	QUATERNARY NITROGEN, SP3, POSITIVELY CHARGED
OM	35	ALKOXIDE OXYGEN, NEGATIVELY CHARGED
OM2	35	OXIDE OXYGEN ON SP2 CARBON, NEGATIVELY CHARGED
HNR+	36	H ON QUATERNARY NITROGEN
HIM+	36	H ON IMIDAZOLIUM-TYPE NITROGEN
HPD+	36	H ON PROTONATED PYRIDINE NITROGEN
HNN+	36	H ON AMIDINIUM-TYPE NITROGEN
HNC+	36	H ON PROTONATED IMINE NITROGEN
HGD+	36	H ON GUANIDINIUM-TYPE NITROGEN
HN5+	36	H ON N5+, N5A+ OR N5B+
CB	37	CARBON AS IN BENZENE, PYRROLE
NPYD	38	NITROGEN, AS IN PYRIDINE
NPYL	39	NITROGEN, AS IN PYRROLE
NC=C	40	NITROGEN ON N-C=C
NC=N	40	NITROGEN IN N-C=N
NC=P	40	NITROGEN IN N-C=P
NC%C	40	NITROGEN ATTACHED TO C-C TRIPLE BOND
CO2M	41	CARBOXYLATE ANION CARBON
CS2M	41	CARBON IN THIOCARBOXYLATE ANION
NSP	42	NITROGEN, TRIPLE BONDED
NSO2	43	NITROGEN IN SULFONAMIDES
NSO3	43	NITROGEN IN SULFONAMIDES, THREE O'S ON S
NPO2	43	NITROGEN IN PHOSPHONAMIDES
NPO3	43	NITROGEN IN PHOSPHONAMIDES, THREE O'S ON P
NC%N	43	NITROGEN ATTACHED TO CYANO GROUP
STHI	44	SULFUR AS IN THIOPHENE
NO2	45	NITRO GROUP NITROGEN
NO3	45	NITRATE GROUP NITROGEN
N=O	46	NITROSO NITROGEN
NAZT	47	TERMINAL NITROGEN IN AZIDO OR DIAZO GROUP
NSO	48	DIVALENT NITROGEN REPLACING MONOVALENT O IN SO2 GROUP
O+	49	POSITIVELY CHARGED OXONIUM (TRICOORDINATE) OXYGEN
HO+	50	HYDROGEN ON O+ OXYGEN
O=+	51	POSITIVELY CHARGED OXENIUM (DICOORDINATE) OXYGEN
HO=+	52	HYDROGEN ON OXENIUM OXYGEN
=N=	53	NITROGEN IN C=N=N OR -N=N=N
N+=C	54	POSITIVELY CHARGED IMINIUM NITROGEN
N+=N	54	POSITIVELY CHARGED NITROGEN DOUBLE-BONDED TO N
NCN+	55	N IN +N=C-N RESONANCE STRUCTURES - FORMAL CHARGE=1/2
NGD+	56	GUANIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1/3
CGD+	57	GUANIDINIUM CARBON
CNN+	57	C IN +N=C-N RESONANCE STRUCTURES
NPD+	58	PYRIDINIUM-TYPE NITROGEN - FORMAL CHARGE=1
OFUR	59	AROMATIC OXYGEN AS IN FURAN
C%	60	ISONITRILE CARBON
NR%	61	ISONITRILE NITROGEN [FC = 0] OR DIAZO NITROGEN [FC=1]
NM	62	DEPROTONATED SULFONAMIDE N-; FORMAL CHARGE=-1
C5A	63	ALPHA CARBON IN 5-MEMBERED HETEROAROMATIC RING
C5B	64	BETA CARBON IN 5-MEMBERED HETEROAROMATIC RING
N5A	65	ALPHA AROM HETEROCYCLIC 5-RING NITROGEN
N5B	66	BETA AROM HETEROCYCLIC 5-RING NITROGEN
N2OX	67	SP2-HYDRIDIZED N-OXIDE NITROGEN
N3OX	68	SP3-HYDRIDIZED N-OXIDE NITROGEN
NPOX	69	PYRIDINE N-OXIDE NITROGEN
OH2	70	OXYGEN ON WATER
HS	71	H ATTACHED TO DIVALENT, DICOORDINATE S
HS=N	71	H ATTACHED TO TETRAVALENT, TRICOODR S DBL BONDED TO N
HP	71	H ATTACHED TO TRI- OR TETRACOORDINATE PHOSPHORUS
S-P	72	TERMINAL SULFUR BONDED TO PHOSPHORUS
S2CM	72	TERMINAL SULFUR IN THIOCARBOXYLATE ANION
SM	72	TERMINAL SULFUR - FORMAL CHARGE=-1
SSMO	72	TERMINAL SULFUR IN THIOSULFINATE GROUP
SO2M	73	SULFUR IN NEGATIVELY CHARGED SULFINATE GROUP
SSOM	73	TRICOORD SULFUR IN THIOSULFINATE GROUP
=S=O	74	SULFINYL SULFUR, EG. IN C=S=O
-P=C	75	PHOSPHOROUS DOUBLY BONDED TO CARBON
N5M	76	NEGATIVELY CHARGED N IN, E.G, TRI- OR TETRAZOLE ANION
CLO4	77	CHLORINE IN PERCHLORATE ANION, CLO4(-)
C5	78	GENERAL CARBON IN 5-MEMBERED HETEROAROMATIC RING
N5	79	GENERAL NITROGEN IN 5-MEMBERED HETEROCYCLIC RING
CIM+	80	C IN N-C-N IN IMIDAZOLIUM ION
NIM+	81	IMIDAZOLIUM-TYPE NITROGEN - FORMAL CHARGE=1/2
N5A+	81	POSITIVE N5A NITROGEN - FORMAL CHARGE=1
N5B+	81	POSITIVE N5B NITROGEN - FORMAL CHARGE=1
N5+	81	POSITIVE N5 NITROGEN - FORMAL CHARGE=1
N5AX	82	N-OXIDE NITROGEN IN 5-RING ALPHA POSITION
N5BX	82	N-OXIDE NITROGEN IN 5-RING BETA POSITION
N5OX	82	N-OXIDE NITROGEN IN GENERAL 5-RING POSITION
FE+2	87	IRON +2 CATION
FE+3	88	IROM +3 CATION
F-	89	FLUORIDE ANION
CL-	90	CHLORIDE ANION
BR-	91	BROMIDE ANION
LI+	92	LITHIUM CATION
NA+	93	SODIUM CATION
K+	94	POTASSIUM CATION
ZINC	95	DIPOSITIVE ZINC
ZN+2	95	DIPOSITIVE ZINC
CA+2	96	DIPOSITIVE CALCIUM
CU+1	97	MONOPOSITIVE COPPER
CU+2	98	DIPOSITIVE COPPER
MG+2	99	DIPOSITIVE MAGNESIUM CATION

MMFF atom type and charge assignment (C++)

```
MMFFMolProperties mmffMolProperties(ROMol &mol, std::string mmffVariant = "MMFF94",
  boost::uint8_t verbosity = MMFF_VERBOSITY_NONE, std::ostream &oStream = std::cout);
```

```
// g++ -Wall -Wno-unused-function -o assignAtomTypes assignAtomTypes.cpp \
//   -I$RDBASE/Code -L$RDBASE/lib -lFileParsers -lRDGeneral -lForceFieldHelpers

#include <GraphMol/RDKitBase.h>
#include <GraphMol/SmilesParse/SmilesParse.h>
#include <GraphMol/ForceFieldHelpers/MMFF/AtomTyper.h>

using namespace RDKit;

int main()
{
  // assign and print MMFF atom types & charges

  ROMol *mol = SmilesToMol("CC(=O)C");
  MMFF::MMFFMolProperties mp(*mol);

  for (unsigned int i = 0; i < mol->getNumAtoms(); ++i) {
    std::cout << i + 1 << "\t" <<
      (unsigned int)mp.getMMFFAtomType(i) << "\t" <<
      mp.getMMFFFormalCharge(i) << "\t" <<
      mp.getMMFFPartialCharge(i) << std::endl;
  }

  return 0;
}
```

MMFF atom type and charge assignment (Python)

```
pyMMFFMolProperties = MMFFGetMoleculeProperties(mol, mmffVariant = "MMFF94", mmffVerbosity = 0)
```

```
# assign and print MMFF atom types

from rdkit import Chem
from rdkit.Chem import ChemicalForceFields

mol = Chem.MolFromSmiles("CC(=O)C")
mp = ChemicalForceFields.MMFFGetMoleculeProperties(mol)
for i in range(0, mol.GetNumAtoms()):
  print i + 1, '\t', mp.GetMMFFAtomType(i), \
  '\t', float(mp.GetMMFFFormalCharge(i)), \
  '\t', float(mp.GetMMFFPartialCharge(i))
```

MMFFMolProperties class: methods (C++)

Get MMFF atom types/charges

```
const boost::uint8_t mmffMolProperties.getMMFFAtomType(const unsigned int idx)
const double mmffMolProperties.getMMFFFormalCharge(const unsigned int idx)
const double mmffMolProperties.getMMFFPartialCharge(const unsigned int idx)
```

Include/exclude energy terms

```
void mmffMolProperties.setMMFFBondTerm(const bool state)
void mmffMolProperties.setMMFFAngleTerm(const bool state)
void mmffMolProperties.setMMFFStretchBendTerm(const bool state)
void mmffMolProperties.setMMFFOopTerm(const bool state)
void mmffMolProperties.setMMFFTorsionTerm(const bool state)
void mmffMolProperties.setMMFFVdWTerm(const bool state)
void mmffMolProperties.setMMFFEleTerm(const bool state)
```

Set MMFF94(s) variant, dielectric constant/model, verbosity

```
void mmffMolProperties.setMMFFVariant(const std::string mmffVariant)
void mmffMolProperties.setMMFFDielectricConstant(const double dielConst)
void mmffMolProperties.setMMFFDielectricModel(boost::uint8_t dielModel)
void mmffMolProperties.setMMFFVerbosity(boost::uint8_t verbosity)
void mmffMolProperties.setMMFFOStream(std::ostream *oStream)
```

MMFFMolProperties class: methods (Python)

Get MMFF atom types/charges

```
atomType = pyMMFFMolProperties.GetMMFFAtomType(idx)
formalCharge = pyMMFFMolProperties.GetMMFFFormalCharge(idx)
partialCharge = pyMMFFMolProperties.GetMMFFPartialCharge(idx)
```

Include/exclude energy terms

```
pyMMFFMolProperties.SetMMFFBondTerm(state)
pyMMFFMolProperties.SetMMFFAngleTerm(state)
pyMMFFMolProperties.SetMMFFStretchBendTerm(state)
pyMMFFMolProperties.SetMMFFOopTerm(state)
pyMMFFMolProperties.SetMMFFTorsionTerm(state)
pyMMFFMolProperties.SetMMFFVdWTerm(state)
pyMMFFMolProperties.SetMMFFEleTerm(state)
```

Set MMFF94(s) variant, dielectric constant/model, verbosity

```
pyMMFFMolProperties.SetMMFFVariant(mmffVariant)
pyMMFFMolProperties.SetMMFFDielectricConstant(dielConst)
pyMMFFMolProperties.SetMMFFDielectricModel(dielModel)
pyMMFFMolProperties.SetMMFFVerbosity(verbosity)
```

How do I use MMFF? Well, almost like UFF

Load dataset from SDF;
minimize;
compute MMFF energy
(C++)

```cpp
// g++ -Wall -Wno-unused-function -o computeEnergy computeEnergy.cpp \
//  -I$RDBASE/Code -L$RDBASE/lib -lFileParsers -lRDGeneral -lForceFieldHelpers -lForceField

#include <GraphMol/RDKitBase.h>
#include <GraphMol/FileParsers/MolSupplier.h>
#include <GraphMol/FileParsers/MolWriters.h>
#include <GraphMol/ForceFieldHelpers/MMFF/AtomTyper.h>
#include <GraphMol/ForceFieldHelpers/MMFF/Builder.h>
#include <ForceField/ForceField.h>

using namespace RDKit;

int main() {
  std::string sdf = "ref_e2.sdf";
  SDMolSupplier supplier(sdf, true, false);
  SDWriter::SDWriter sdfWrite (sdf.substr(0, sdf.size() - 4) + "_MMFF_min.sdf");
  ROMol *mol;
  std::cout << "MMFF ENERGY\tBEFORE MIN\tAFTER MIN\n";
  for (unsigned int i = 0; i < supplier.length(); ++i) {
    mol = supplier[i];
    std::string molName = "";
    if (mol->hasProp("_Name")) {
      mol->getProp("_Name", molName);
    }
    MMFF::MMFFMolProperties mp(*mol);
    ForceFields::ForceField *field = MMFF::constructForceField(*mol, &mp);
    field->initialize();
    double e1 = field->calcEnergy();
    field->minimize();
    double e2 = field->calcEnergy();
    std::cout << molName << std::setprecision(3) << std::fixed << "\t\t" << e1 << "\t\t" << e2 << "\n";
    sdfWrite.write(*mol);
    delete mol;
  }
  sdfWrite.close();
  return 0;
}
```

How do I use MMFF? Well, almost like UFF

```
from rdkit import Chem
from rdkit.Chem import AllChem

sdf = 'ref_e2.sdf'
supplier = Chem.SDMolSupplier(sdf, True, False)
sdfWrite = Chem.SDWriter(sdf[:-4] + '_MMFF_min.sdf')

print 'MMFF ENERGY\tBEFORE MIN\tAFTER MIN'
for i in range(0, len(supplier)):
  mol = supplier[i]
  if (mol.HasProp('_Name')):
    molName = mol.GetProp('_Name')
  mp = AllChem.MMFFGetMoleculeProperties(mol)
  field = AllChem.MMFFGetMoleculeForceField(mol, mp)
  e1 = field.CalcEnergy()
  field.Minimize()
  e2 = field.CalcEnergy()
  print '{0:}\t\t{1:.3f}\t\t{2:.3f}'.format(molName, e1, e2)
  sdfWrite.write(mol)
sdfWrite.close()
```

Load dataset from SDF;
minimize;
compute MMFF energy
(Python)

Load dataset from SMILES; generate 3D; MMFF-minimize (C++)

```
// g++ -Wall -Wno-unused-function -o gen3D gen3D.cpp -I$RDBASE/Code -L$RDBASE/lib \
//  -lFileParsers -lRDGeneral -lForceFieldHelpers -lForceField -lDistGeomHelpers

#include <GraphMol/RDKitBase.h>
#include <GraphMol/FileParsers/MolSupplier.h>
#include <GraphMol/FileParsers/MolWriters.h>
#include <GraphMol/ForceFieldHelpers/MMFF/AtomTyper.h>
#include <GraphMol/ForceFieldHelpers/MMFF/Builder.h>
#include <GraphMol/MolOps.h>
#include <GraphMol/DistGeomHelpers/Embedder.h>
#include <ForceField/ForceField.h>

using namespace RDKit;

int main() {
  std::string smi = "ref_e2.smi";
  SmilesMolSupplier supplier(smi);
  SDWriter::SDWriter sdfWrite (smi.substr(0, smi.size() - 4) + "_MMFF_gen3D.sdf");
  ROMol *mol, *molH;
  std::cout << "NAME\t\tMMFF ENERGY\n";
  for (unsigned int i = 0; i < supplier.length(); ++i) {
    mol = supplier[i];
    std::string molName = "";
    if (mol->hasProp("_Name")) {
      mol->getProp("_Name", molName);
    }
    molH = MolOps::addHs(*mol);
    DGeomHelpers::EmbedMolecule(*molH);
    MMFF::MMFFMolProperties mp(*molH);
    ForceFields::ForceField *field = MMFF::constructForceField(*molH, &mp);
    field->initialize();
    field->minimize();
    double e = field->calcEnergy();
    std::cout << molName << std::setprecision(3) << std::fixed << "\t\t" << e << "\n";
    sdfWrite.write(*molH);
    delete mol;
    delete molH;
  }
  sdfWrite.close();
  return 0;
}
```

How do I use MMFF? Well, almost like UFF

```python
from rdkit import Chem
from rdkit.Chem import AllChem

smi = 'ref_e2.smi'
supplier = Chem.SmilesMolSupplier(smi)
sdfWrite = Chem.SDWriter(smi[:-4] + '_MMFF_gen3D.sdf')

print 'NAME\t\tMMFF ENERGY'
for i in range(0, len(supplier)):
  mol = supplier[i]
  if (mol.HasProp('_Name')):
    molName = mol.GetProp('_Name')
  molH = AllChem.AddHs(mol)
  AllChem.EmbedMolecule(molH)
  mp = AllChem.MMFFGetMoleculeProperties(molH)
  field = AllChem.MMFFGetMoleculeForceField(molH, mp)
  field.Minimize()
  e = field.CalcEnergy()
  print '{0:}\t\t{1:.3f}'.format(molName, e)
  sdfWrite.write(molH)
sdfWrite.close()
```

Load dataset from SMILES;
generate 3D;
MMFF-minimize
(Python)

Validation of the RDKit MMFF implementation

The RDKit implementation was validated against the official MMFF94 (761 molecules) and MMFF94s (265 molecules) validation suites available in the CCL data archives (both dative and hypervalent notations):

`http://ccl.net/chemistry/resources/data/index.shtml`

- All atoms are assigned correct MMFF types and charges
- All force-field terms are assigned correct force constants
- All MMFF energies are correctly computed
- Gradients were also validated against the TINKER MMFF implementation

- Even better, atom type, charge and force constants are correctly assigned also starting from SMILES representations of the validation suite molecules

- MMFF validation is included as part of the RDKit test suite (`Code/ForceField/MMFF/testMMFFForceField.cpp`)

Force-field wish list (open to suggestions)

- GBSA implicit solvation model

During energy minimization, support for:

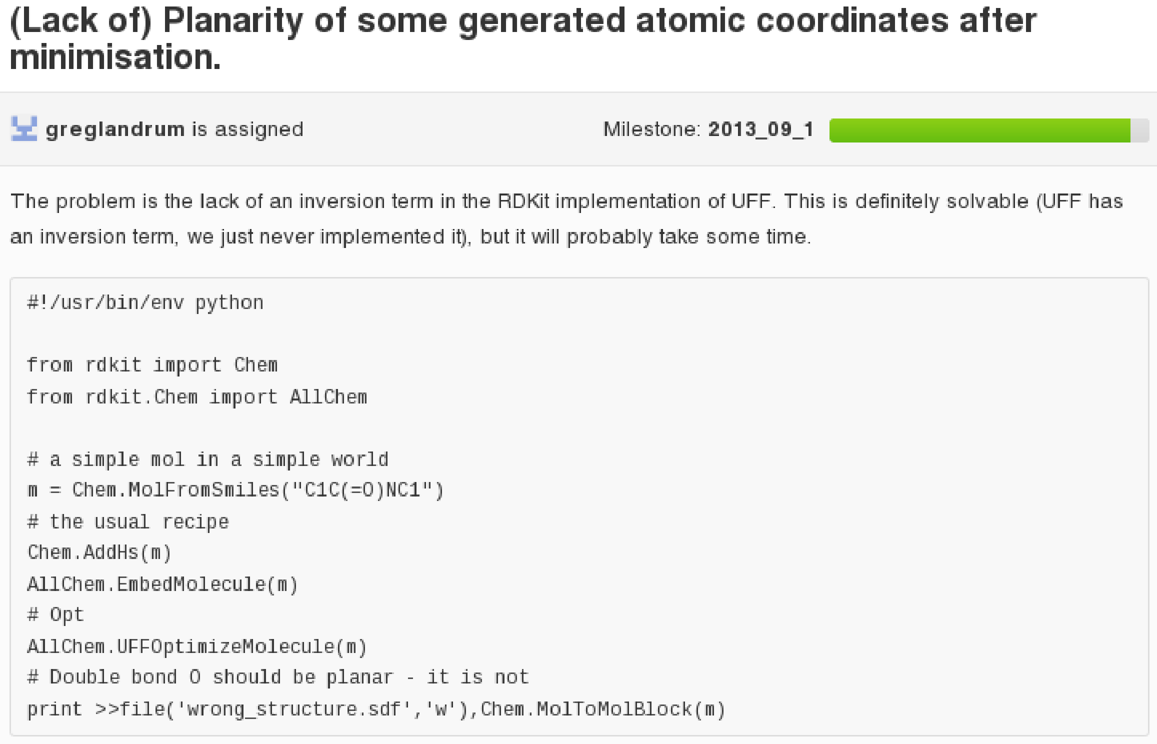
- Fixed atoms
- Harmonic constraints on selected atoms/groups
- Internal coordinate constraints
  (in addition to distances, also angles and torsions)

Topics

- The MMFF implementation
- UFF gains something, too
- MMFF applications

UFF issue 62

The RDKit implementation of UFF had an issue connected with sp^2 atoms lacking planarity in 3- and 4-membered rings



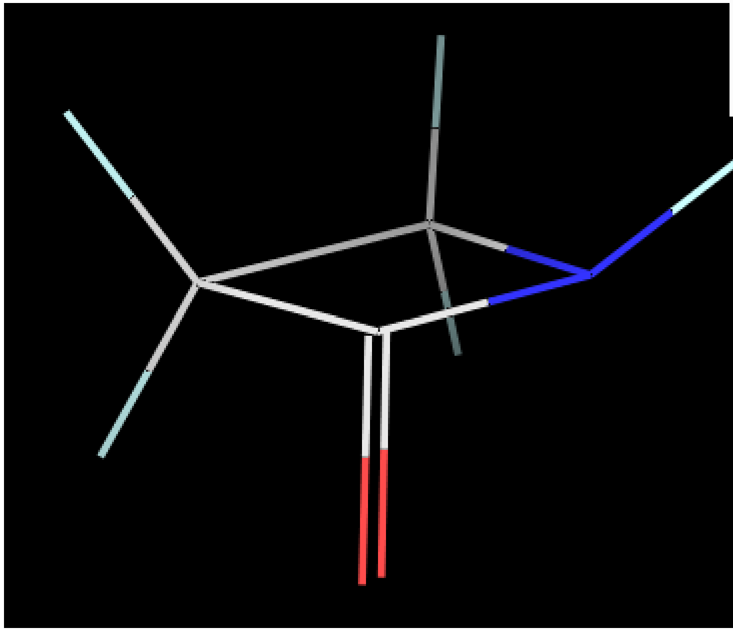
(Lack of) Planarity of some generated atomic coordinates after minimisation.

greglandrum is assigned

Milestone: **2013_09_1**

The problem is the lack of an inversion term in the RDKit implementation of UFF. This is definitely solvable (UFF has an inversion term, we just never implemented it), but it will probably take some time.

```
#!/usr/bin/env python

from rdkit import Chem
from rdkit.Chem import AllChem

# a simple mol in a simple world
m = Chem.MolFromSmiles("C1C(=O)NC1")
# the usual recipe
Chem.AddHs(m)
AllChem.EmbedMolecule(m)
# Opt
AllChem.UFFOptimizeMolecule(m)
# Double bond O should be planar - it is not
print >>file('wrong_structure.sdf','w'),Chem.MolToMolBlock(m)
```

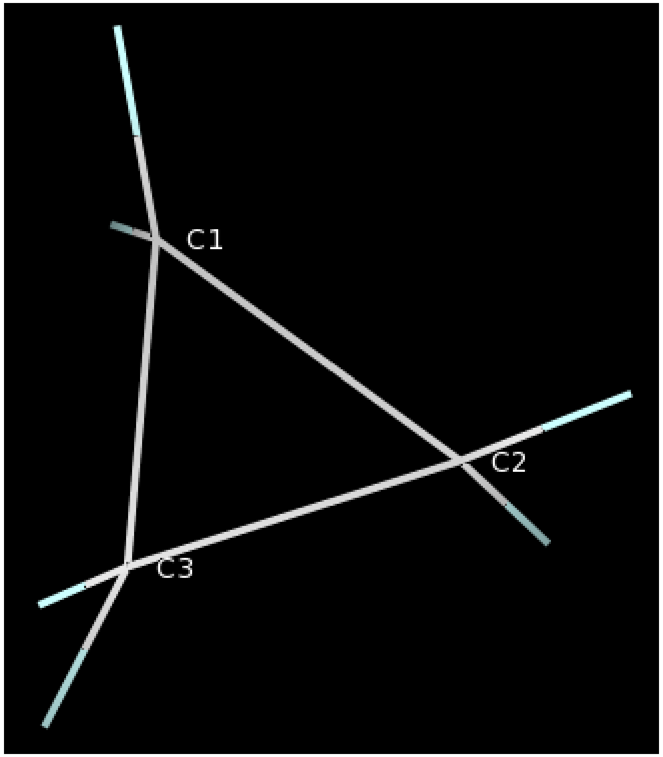


- The issue is arguably connected with the lack of the inversion term
- Back-porting the OOP term from MMFF to UFF should most likely fix things

Angle enumeration

```
PyMOL>get_angle id 1, id 2, id 3
 cmd.get_angle: 56.674 degrees.
PyMOL>get_angle id 1, id 3, id 2
 cmd.get_angle: 66.654 degrees.
PyMOL>get_angle id 2, id 1, id 3
 cmd.get_angle: 56.672 degrees.
```

```
PyMOL>get_distance id 1, id 2
 cmd.get_distance: 1.610 Angstroms.
PyMOL>get_distance id 1, id 3
 cmd.get_distance: 1.465 Angstroms.
PyMOL>get_distance id 2, id 3
 cmd.get_distance: 1.465 Angstroms.
```

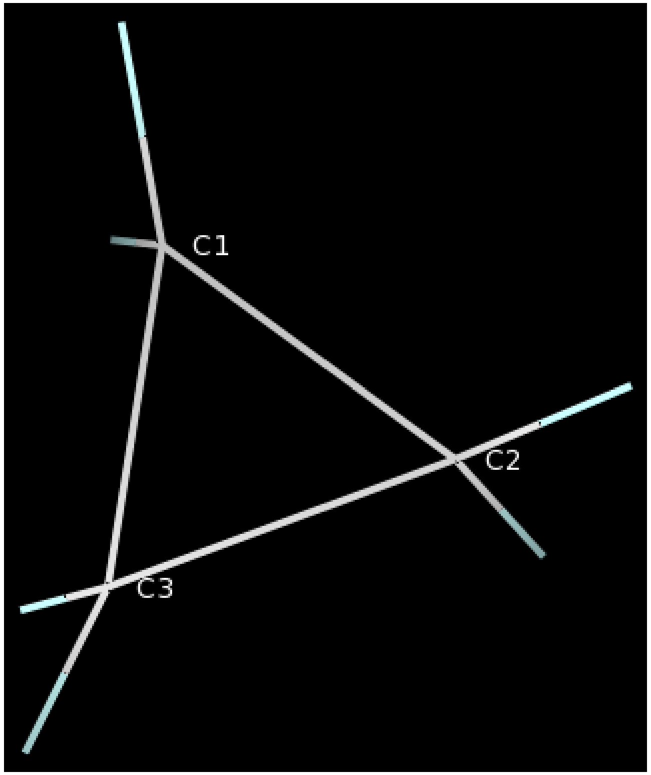


- I noticed that also geometries of small rings containing only sp^3 atoms were somehow ill
- Angle enumeration in UFF was implemented using the same `neighborMatrix` used for non-bonded interactions
- However, since atoms C1, C3 are connected both directly and through atom C2 and the matrix has only one node per atom pair, some angle terms may be missing

Angle enumeration

```
PyMOL>get_angle id 1, id 2, id 3
 cmd.get_angle: 60.003 degrees.
PyMOL>get_angle id 1, id 3, id 2
 cmd.get_angle: 59.998 degrees.
PyMOL>get_angle id 2, id 1, id 3
 cmd.get_angle: 59.999 degrees.
```

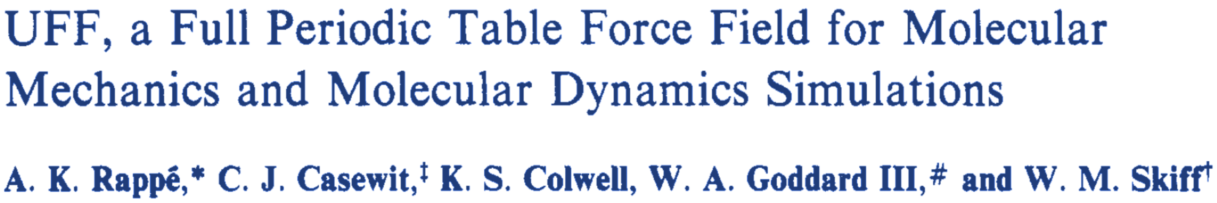
```
PyMOL>get_distance id 1, id 2
 cmd.get_distance: 1.517 Angstroms.
PyMOL>get_distance id 1, id 3
 cmd.get_distance: 1.517 Angstroms.
PyMOL>get_distance id 2, id 3
 cmd.get_distance: 1.517 Angstroms.
```



- Both in MMFF and UFF a graph-based method is now used for angle enumeration, while `neighborMatrix` is used only for non-bonded interactions
- This also allows reducing the size of `neighborMatrix` nodes from a 32-bit `int` to 2 bits, resulting in a 256-fold lower RAM usage
- All torsions are now included also in 3-membered rings
- Angles and distances now look OK

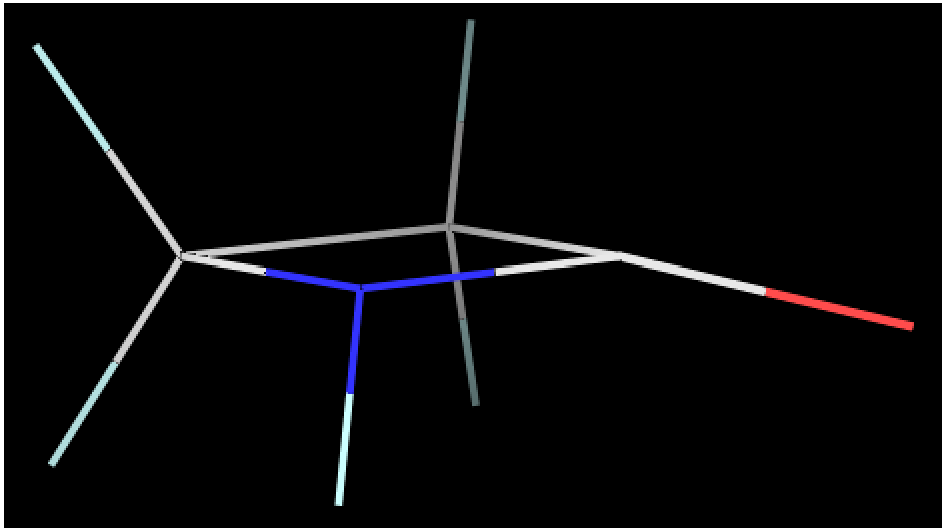
Implementation of inversions in UFF

- Then, I implemented inversions according to the original paper by Rappé et al.:



UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations

A. K. Rappé,[*] C. J. Casewit,[‡] K. S. Colwell, W. A. Goddard III,[#] and W. M. Skiff[†]



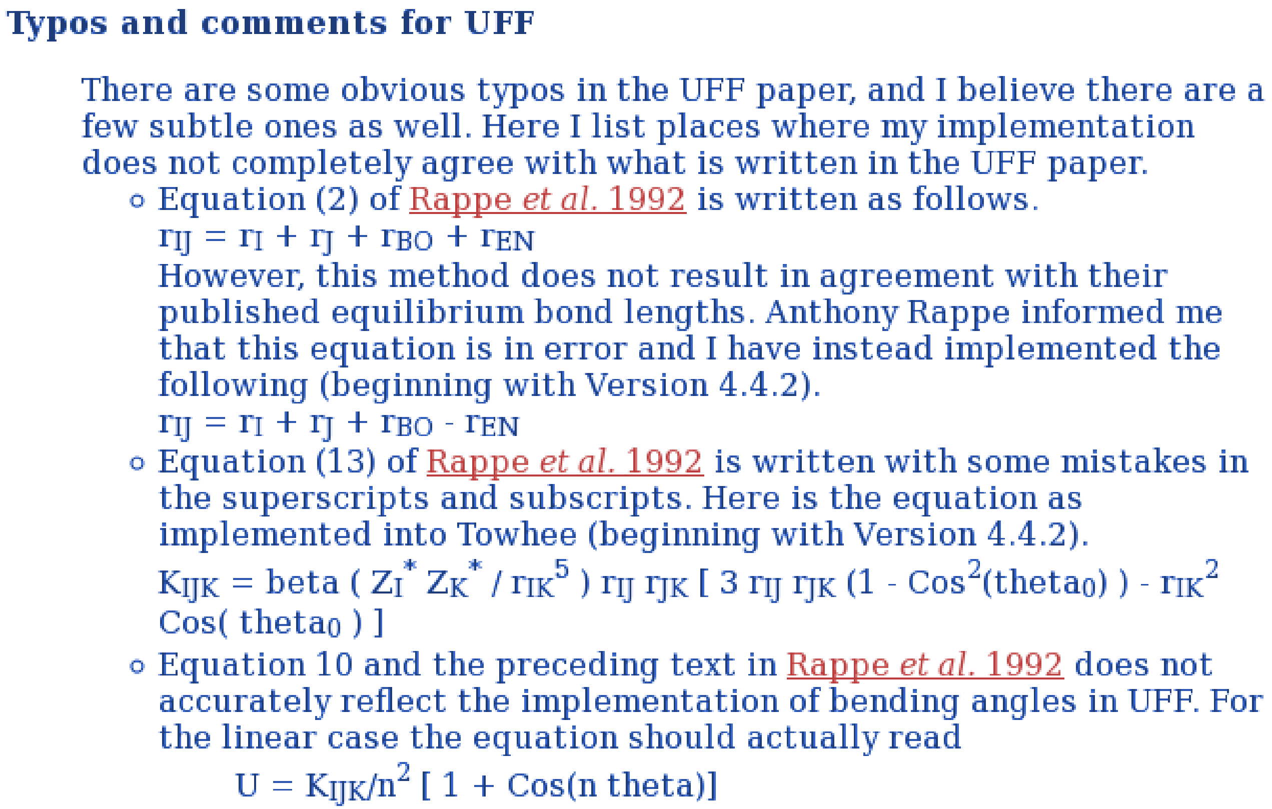
- After making sure that the inversion code was working properly, I ran the script which triggered issue 62:

```
# a simple mol in a simple world
m = Chem.MolFromSmiles("C1C(=O)NC1")
m2 = Chem.AddHs(m)
AllChem.EmbedMolecule(m2)
pyFF = AllChem.UFFGetMoleculeForceField(m2)
pyFF.Minimize()
print >>file('uff_beta-lactam.sdf','w'), \
  Chem.MolToMolBlock(m2)
```

The geometry improved slightly,
but was still far from being satisfactory! Very disappointing.

What is going wrong?

- Investigations on the details of the UFF implementation revealed minor discrepancies with respect to Rappé's paper, which anyway had no impact on the geometries of small
rings containing sp^2 atoms



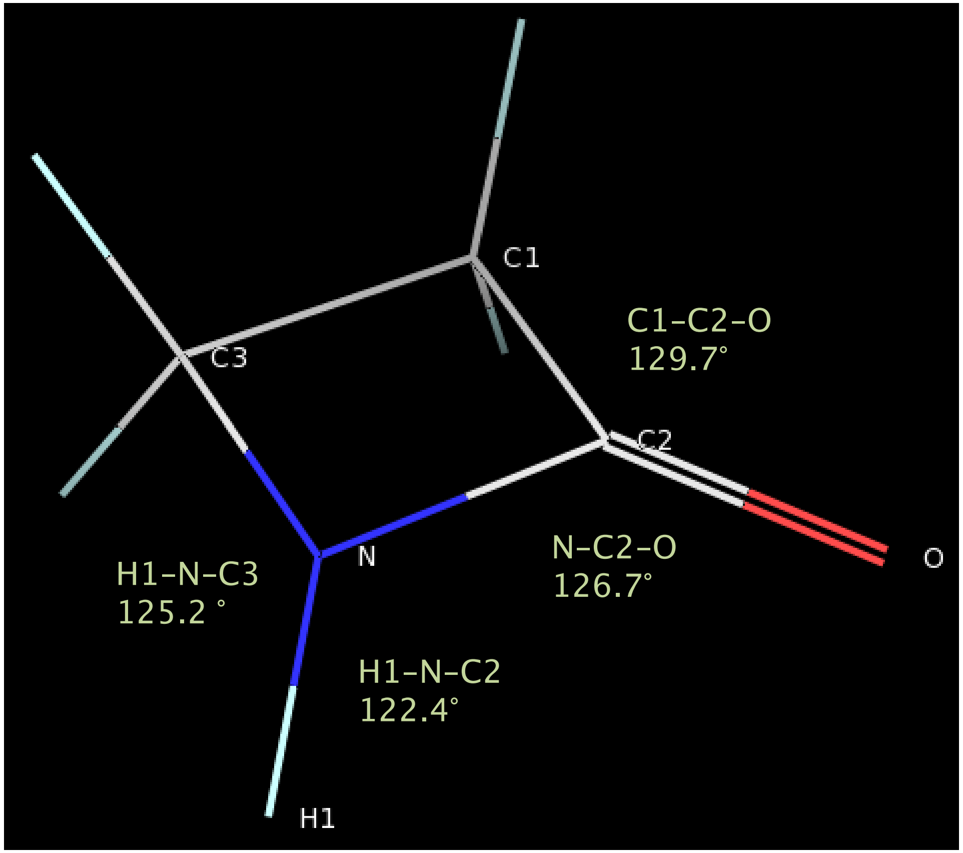
Typos and comments for UFF

There are some obvious typos in the UFF paper, and I believe there are a few subtle ones as well. Here I list places where my implementation does not completely agree with what is written in the UFF paper.

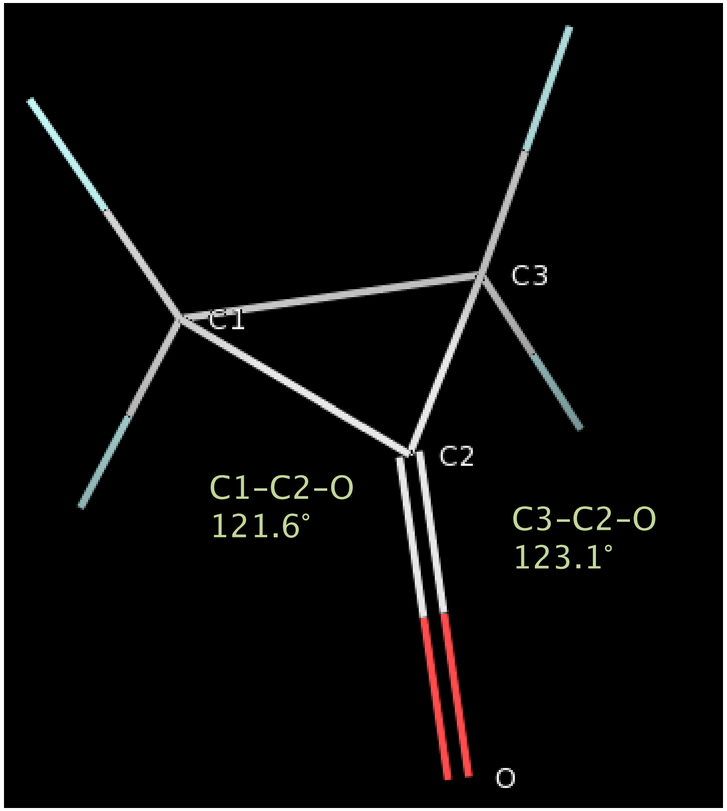
- Equation (2) of Rappe *et al.* 1992 is written as follows.
  $r_{IJ} = r_I + r_J + r_{BO} + r_{EN}$
  However, this method does not result in agreement with their published equilibrium bond lengths. Anthony Rappe informed me that this equation is in error and I have instead implemented the following (beginning with Version 4.4.2).
  $r_{IJ} = r_I + r_J + r_{BO} - r_{EN}$
- Equation (13) of Rappe *et al.* 1992 is written with some mistakes in the superscripts and subscripts. Here is the equation as implemented into Towhee (beginning with Version 4.4.2).
  $K_{IJK} = \text{beta} (Z_I^* Z_K^* / r_{IK}^5) r_{IJ} r_{JK} [3 r_{IJ} r_{JK} (1 - \text{Cos}^2(\text{theta}_0)) - r_{IK}^2 \text{Cos}(\text{theta}_0)]$
- Equation 10 and the preceding text in Rappe *et al.* 1992 does not accurately reflect the implementation of bending angles in UFF. For the linear case the equation should actually read
  $U = K_{IJK}/n^2 [1 + \text{Cos}(n \text{ theta})]$

- During my investigations I also ran across these suggestions by Marcus Martin, the author of Towhee, who found some typos/mistakes in the original paper. However, none of these fixes solved issue 62

Competition between angle and OOP bend terms

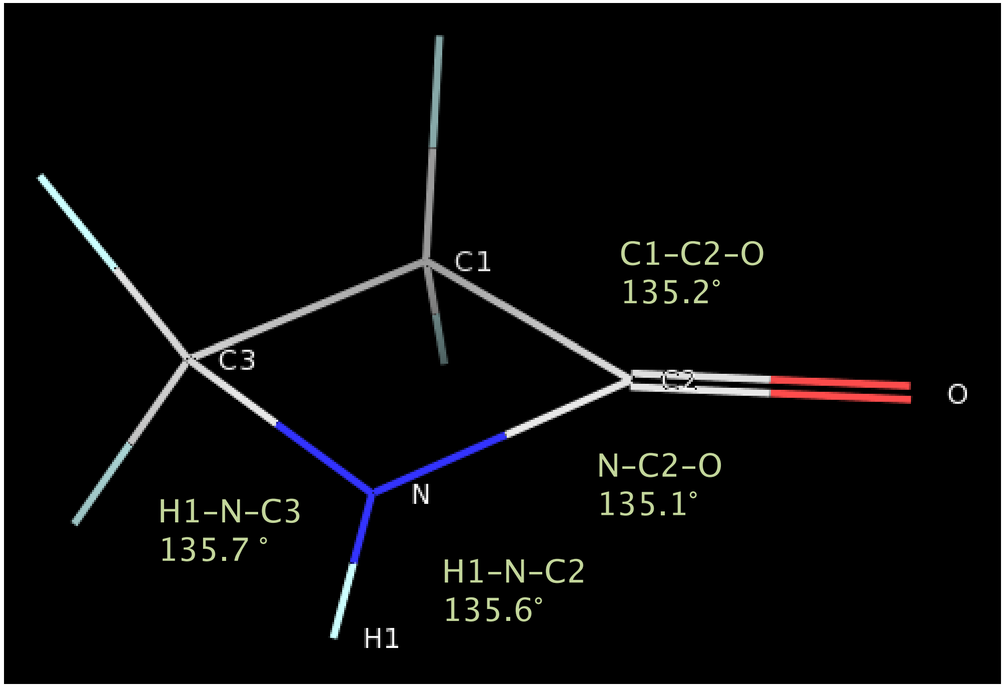


- Measuring the angles involving two edges of the small ring and an exocyclic atom shows that such angles are close to 120°, which is the equilibrium angle for sp^2 atoms

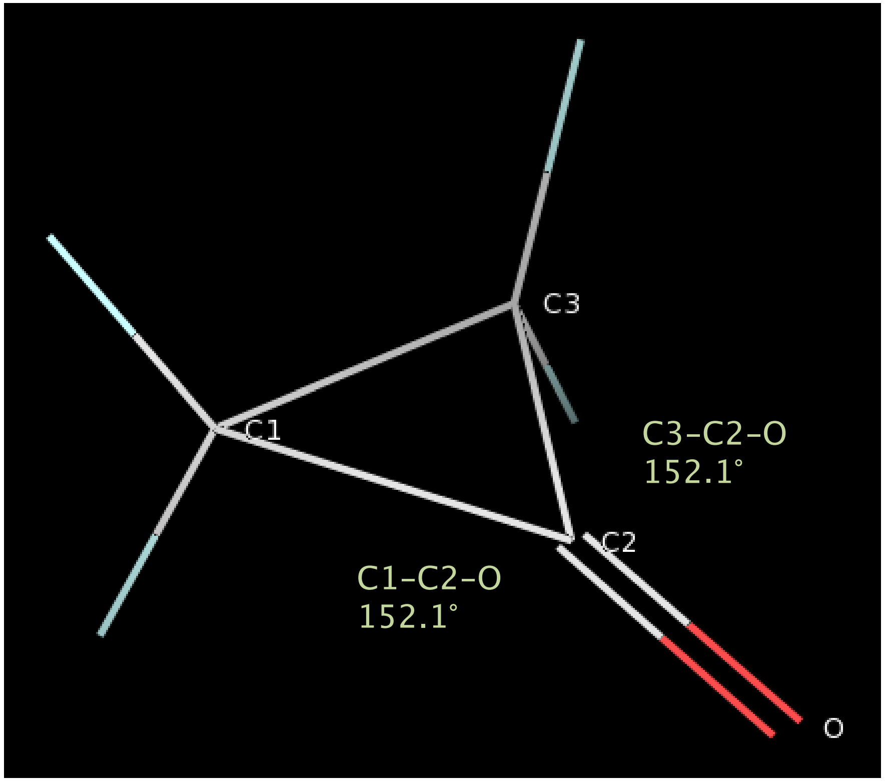


- There seems to be a competition between angle and OOP bend terms, the first pulling the exocyclic atom out-of-plane, the second trying to keep it coplanar with the ring

Hacking UFF equilibrium angles for small rings

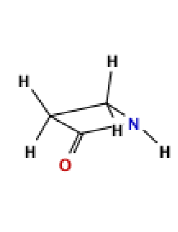
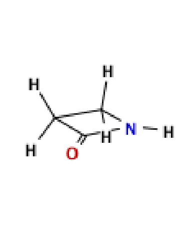
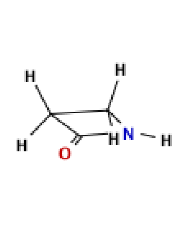
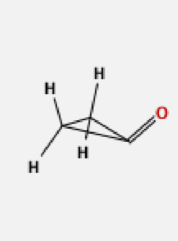
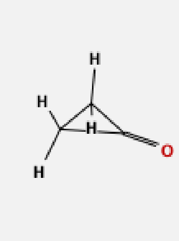
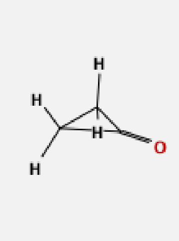
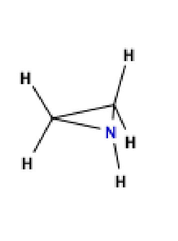
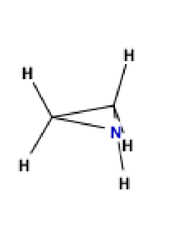
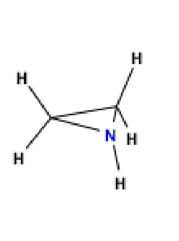
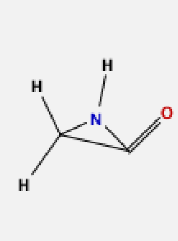
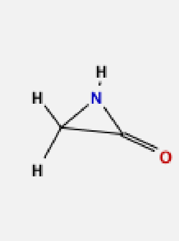
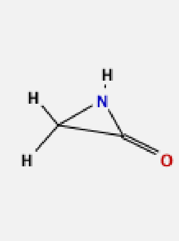
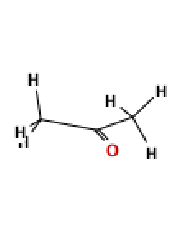
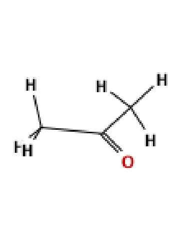
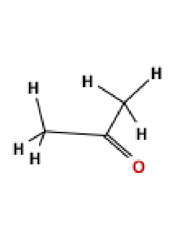
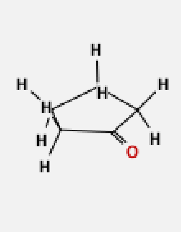
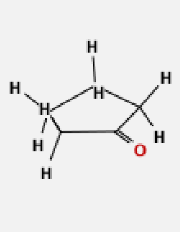
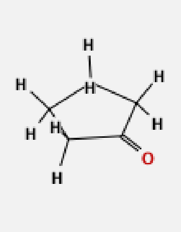
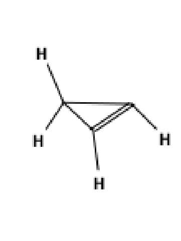
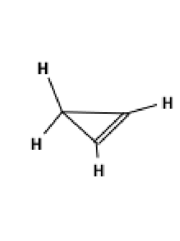
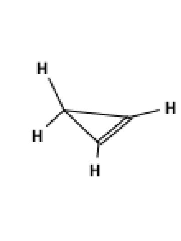
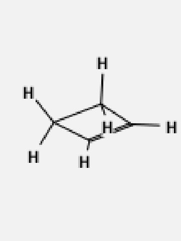
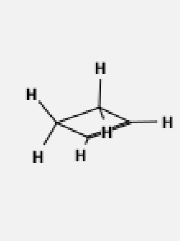
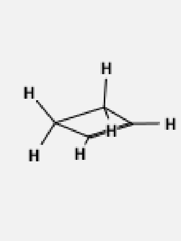
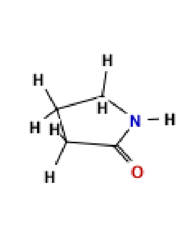
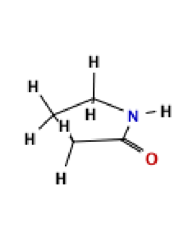
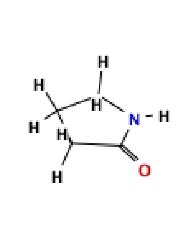
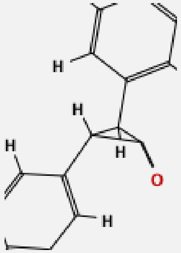
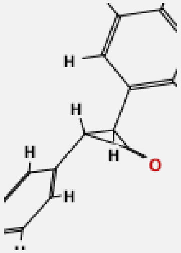
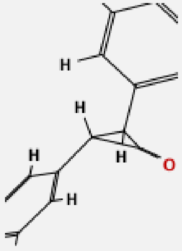
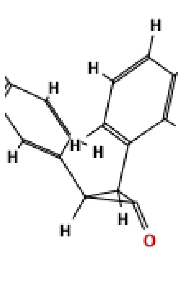
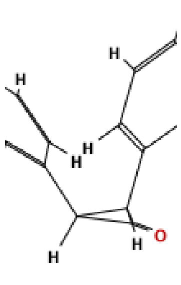
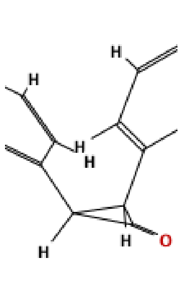
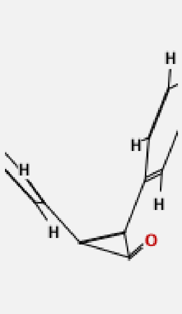
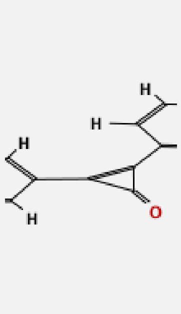
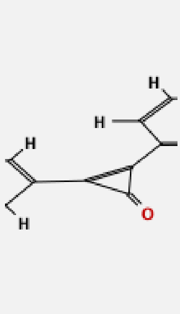
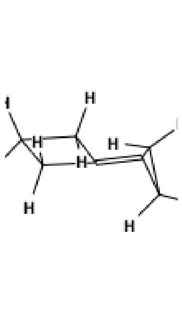
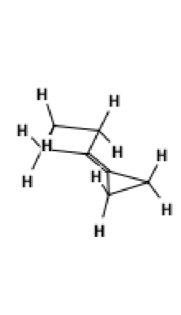
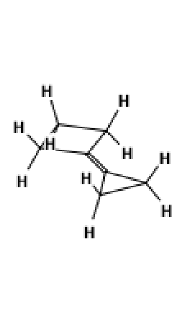
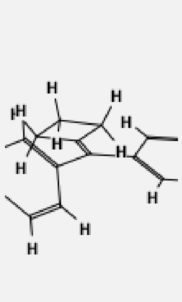
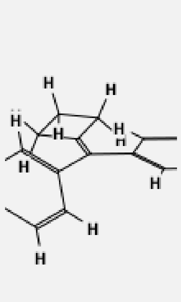
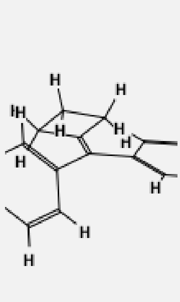


- I set equilibrium values for exocyclic angles involving sp^2 atoms to 135° and 150°, and endocyclic angles to 90° and 60° for 4- and 3-membered rings, respectively



- Now exocyclic sp^2 atoms are coplanar with the ring, as they should be
- Please note that this hack impacts only on sp^2 atoms in 3- and 4-membered rings

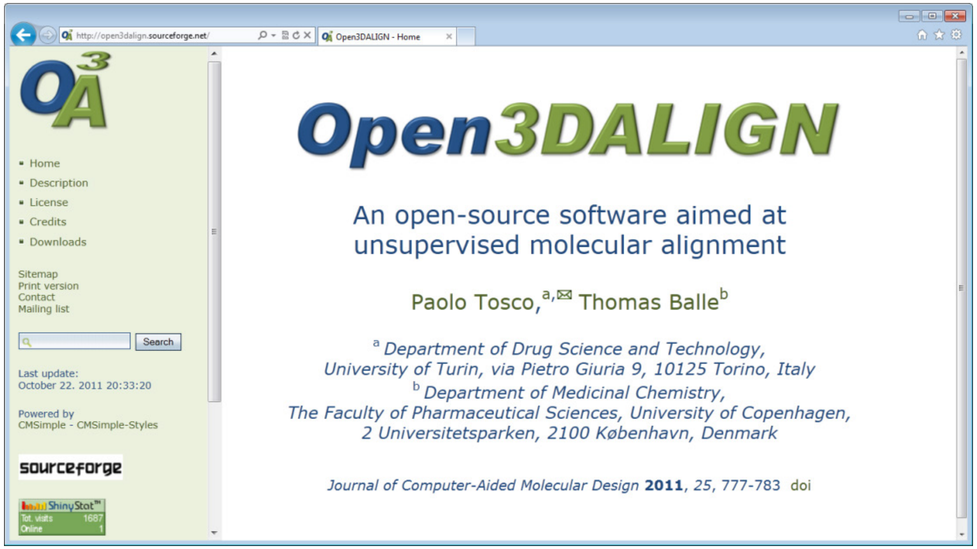
Small ring geometries obtained with original UFF, hacked UFF, AM1

	original UFF	hacked UFF	AM1
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

Topics

- The MMFF implementation
- UFF gains something, too
- MMFF applications

Implementing Open3DALIGN in the RDKit



Open3DALIGN

An open-source software aimed at unsupervised molecular alignment

Paolo Tosco,[a,✉] Thomas Balle[b]

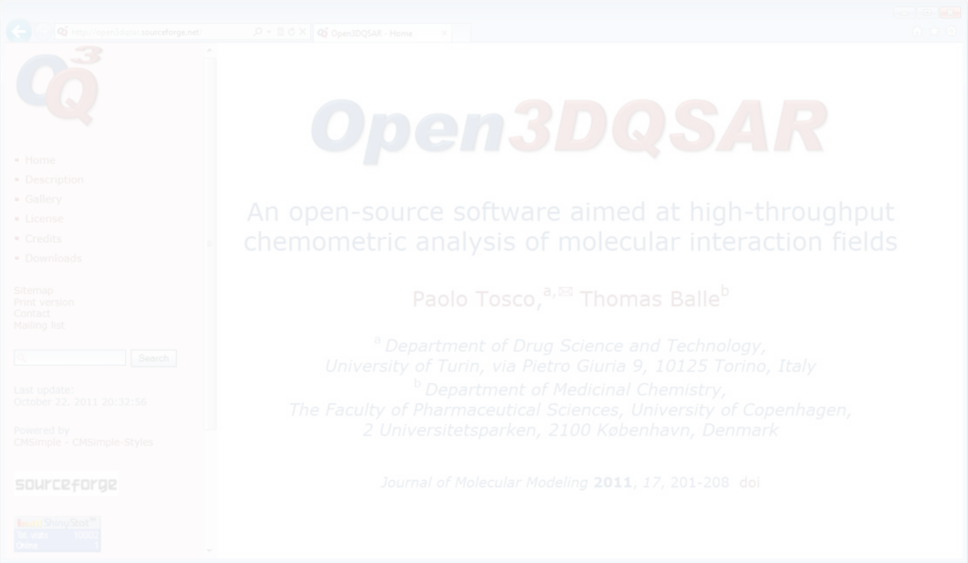
[a] *Department of Drug Science and Technology,*
University of Turin, via Pietro Giuria 9, 10125 Torino, Italy
[b] *Department of Medicinal Chemistry,*
The Faculty of Pharmaceutical Sciences, University of Copenhagen,
2 Universitetsparken, 2100 København, Denmark

Journal of Computer-Aided Molecular Design **2011**, *25*, 777-783 doi

Last update:
October 22. 2011 20:33:20

Powered by
CMSimple - CMSimple-Styles

- **Open3DALIGN**
  - molecular alignment
    - single/multiple conformations
    - atom-based alignment
    - single-run



Open3DQSAR

An open-source software aimed at high-throughput chemometric analysis of molecular interaction fields

Paolo Tosco,[a,✉] Thomas Balle[b]

[a] *Department of Drug Science and Technology,*
University of Turin, via Pietro Giuria 9, 10125 Torino, Italy
[b] *Department of Medicinal Chemistry,*
The Faculty of Pharmaceutical Sciences, University of Copenhagen,
2 Universitetsparken, 2100 København, Denmark

Journal of Molecular Modeling **2011**, *17*, 201-208 doi

Last update:
October 22. 2011 20:32:56

Powered by
CMSimple - CMSimple-Styles

- **Open3DQSAR**
  - MIF computation
  - PLS model building and validation
  - variable selection

Open3DALIGN alignment strategy: 1st step

Open3DALIGN uses a 2-step alignment strategy

1

- a cost function c_{ij} is computed for pairing heavy atoms i,j between reference (R) and probe (P) molecules

i

c_{00}	c_{01}	c_{02}	c_{03}	c_{04}	c_{05}	c_{06}	c_{07}	c_{08}	c_{09}
c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}
c_{20}	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}	c_{28}	c_{29}
c_{30}	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}	c_{36}	c_{37}	c_{38}	c_{39}
c_{40}	c_{41}	c_{42}	c_{43}	c_{44}	c_{45}	c_{46}	c_{47}	c_{48}	c_{49}
c_{50}	c_{51}	c_{52}	c_{53}	c_{54}	c_{55}	c_{56}	c_{57}	c_{58}	c_{59}

j

$$c_{ij} = w_1 \left| q(R)_i - q(P)_j \right| + w_2 s_{ij} + \boxed{\sum_{k=1}^{K} \frac{\left(h(P)_{ik} - h(R)_{jk} \right)^2}{h(P)_{ik} + h(R)_{jk}}}$$

$q(R)_i$, $q(P)_j$	MMFF94 charges of atoms
s_{ij}	degree of chemical similarity between MMFF94 atom types i and j
$h(R)$, $h(P)$	K-binned histograms whose k-th bin represents the number of atoms in R and P lying in a (k-1, k) distance range from the i-th and j-th atom, respectively

Open3DALIGN alignment strategy: 1^{st} step

Open3DALIGN uses a 2-step alignment strategy

1

- The total matching cost is minimized via the Jonker-Volgenant algorithm yielding an **E** matrix which describes intermolecular atom pairings

i

c_{00}	c_{01}	c_{02}	c_{03}	c_{04}	c_{05}	c_{06}	c_{07}	c_{08}	c_{09}
c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}
c_{20}	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}	c_{28}	c_{29}
c_{30}	c_{31}	c_{32}	c_{33}	c_{34}	c_{35}	c_{36}	c_{37}	c_{38}	c_{39}
c_{40}	c_{41}	c_{42}	c_{43}	c_{44}	c_{45}	c_{46}	c_{47}	c_{48}	c_{49}
c_{50}	c_{51}	c_{52}	c_{53}	c_{54}	c_{55}	c_{56}	c_{57}	c_{58}	c_{59}

j

- The **E** matrix is then ranked and trimmed to give a **D** matrix, whose atom pairs are least-square-fitted

i	*j*
4	0
2	2
0	3
3	5
5	7
1	8

i	*j*
5	7
1	8
3	5
2	2

Open3DALIGN alignment strategy: 2nd step

Open3DALIGN uses a 2-step alignment strategy

2

- This initial, coarse alignment is then iteratively refined via the SDM algorithm and scored

- Optionally, cost function parameters may be iteratively optimized until the best scoring alignment is found

Scoring function:

$$s = \sum_{k=1}^{p} \left(\alpha + \frac{1 + \beta \cdot \left| q(R)_{\mathrm{D}[k,0]} + q(P)_{\mathrm{D}[k,1]} \right|}{1 + \left| q(R)_{\mathrm{D}[k,0]} - q(P)_{\mathrm{D}[k,1]} \right|} \right) \cdot \exp\left(-\gamma \cdot r^{2}_{\mathrm{D}[k,0]\mathrm{D}[k,1]} \right)$$

Open3DALIGN: the RDKit implementation

- The RDKit already implements least-square-fit and 3D transformation functions dedicated to molecular alignment:

  -
    ```
    double getAlignmentTransform (const ROMol &prbMol, const ROMol &refMol, RDGeom::Transform3D &trans, int prbCid = -1, int refCid = -1, const MatchVectType *atomMap = 0, const RDNumeric::DoubleVector *weights = 0, bool reflect = false, unsigned int maxIters = 50)
    ```

  -
    ```
    double alignMol (ROMol &prbMol, const ROMol &refMol, int prbCid = -1, int refCid = -1, const MatchVectType *atomMap = 0, const RDNumeric::DoubleVector *weights = 0, bool reflect = false, unsigned int maxIters = 50)
    ```

- All we need is to generate via the **Open3DALIGN** functionality a `MatchVectType` with matching atom pairs and a `RDNumeric::DoubleVector` with weights, and pass them to the least-square-fit function

O3A class: methods (C++)

```
O3A o3a(ROMol &prbMol, const ROMol &refMol, MMFF::MMFFMolProperties *prbMP,
  MMFF::MMFFMolProperties *refMP, const int prbCid = -1, const int refCid = -1,
  const bool reflect = false, const unsigned int maxIters = 50,
  const unsigned int options = O3_USE_MMFF_WEIGHTS, LAP *extLAP = NULL)
```

```
// align probe onto reference (returns RMSD)
double rmsd = o3a.align()

// get RMSD and transformation which overlays probe and reference molecules
std::pair<double, RDGeom::Transform3D *> rmsdTrans = o3a.trans()

// get the O3AScore value of the alignment (no need to actually align coordinates)
double o3aScore = o3a.score()

// get the MatchVectType vector
const RDKit::MatchVectType *matches = o3a.matches()

// get the weight vector
const RDNumeric::DoubleVector *weights = o3a.weights()
```

O3A class: methods (Python)

```
GetO3A(prbMol, refMol, prbPyMMFFMolProperties, refPyMMFFMolProperties,
  prbCid = -1, refCid = -1, reflect = False, maxIters = 50, options = O3_USE_MMFF_WEIGHTS)
```

```
# align probe onto reference (returns RMSD)
rmsd = o3a.Align()

# get RMSD and transformation which overlays probe and reference molecules
(rmsd, trans) = o3a.Trans()

# get the O3AScore value of the alignment (no need to actually align coordinates)
o3aScore = o3a.Score()

# get the MatchVectType vector
matches = o3a.Matches()

# get the weight vector
weights = o3a.Weights()
```

O3A alignment (C++)

```
// g++ -Wall -Wno-unused-function -o o3aAlign o3aAlign.cpp -I$RDBASE/Code \
//    -L$RDBASE/lib -lFileParsers -lRDGeneral -lForceFieldHelpers -lForceField -lMolAlign

#include <GraphMol/RDKitBase.h>
#include <GraphMol/FileParsers/MolSupplier.h>
#include <GraphMol/FileParsers/MolWriters.h>
#include <GraphMol/ForceFieldHelpers/MMFF/AtomTyper.h>
#include "GraphMol/MolAlign/AlignMolecules.h"
#include "GraphMol/MolAlign/O3AAlignMolecules.h"

using namespace RDKit;

int main() {
  std::string sdf = "ref_e2_scrambled.sdf";
  std::string o3aSdf = "ref_e2_O3A.sdf";
  SDMolSupplier supplier(sdf, true, false);
  int nMol = supplier.length();
  const int refNum = 48;
  ROMol refMol = *(supplier[refNum]);
  MMFF::MMFFMolProperties refMP(refMol);
  SDWriter::SDWriter *o3aMol = new SDWriter::SDWriter(o3aSdf);
  std::cout << "N\t\tSCORE\t\tRMSD" << std::endl;
  for (int prbNum = 0; prbNum < nMol; ++prbNum) {
    ROMol prbMol = *(supplier[prbNum]);
    MMFF::MMFFMolProperties prbMP(prbMol);
    MolAlign::O3A o3a(prbMol, refMol, &prbMP, &refMP);
    std::cout << prbNum + 1 << "\t\t" << std::fixed << std::setprecision(2)
      << o3a.score() << "\t\t" << o3a.align() << std::endl;
    o3aMol->write(prbMol);
  }
  o3aMol->close();
  return 0;
}
```

Align all molecules in a SDF file on a common reference (C++)

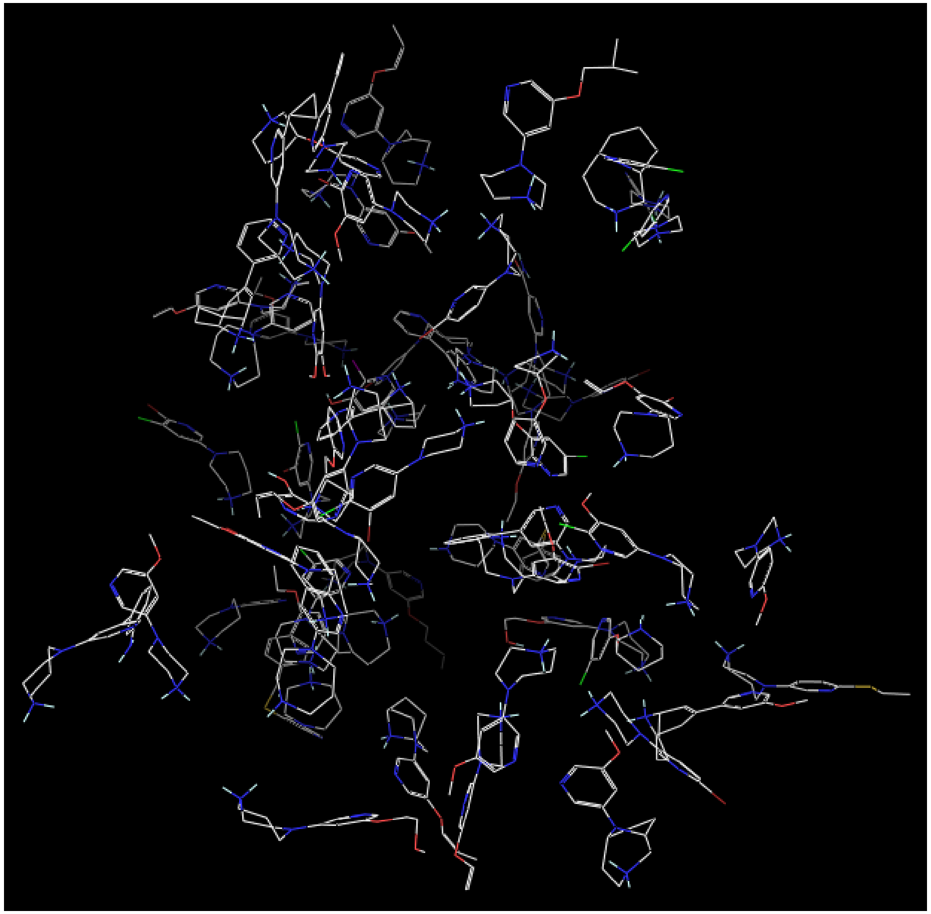
O3A alignment (Python)

Align all molecules in a SDF file on a common reference (Python)

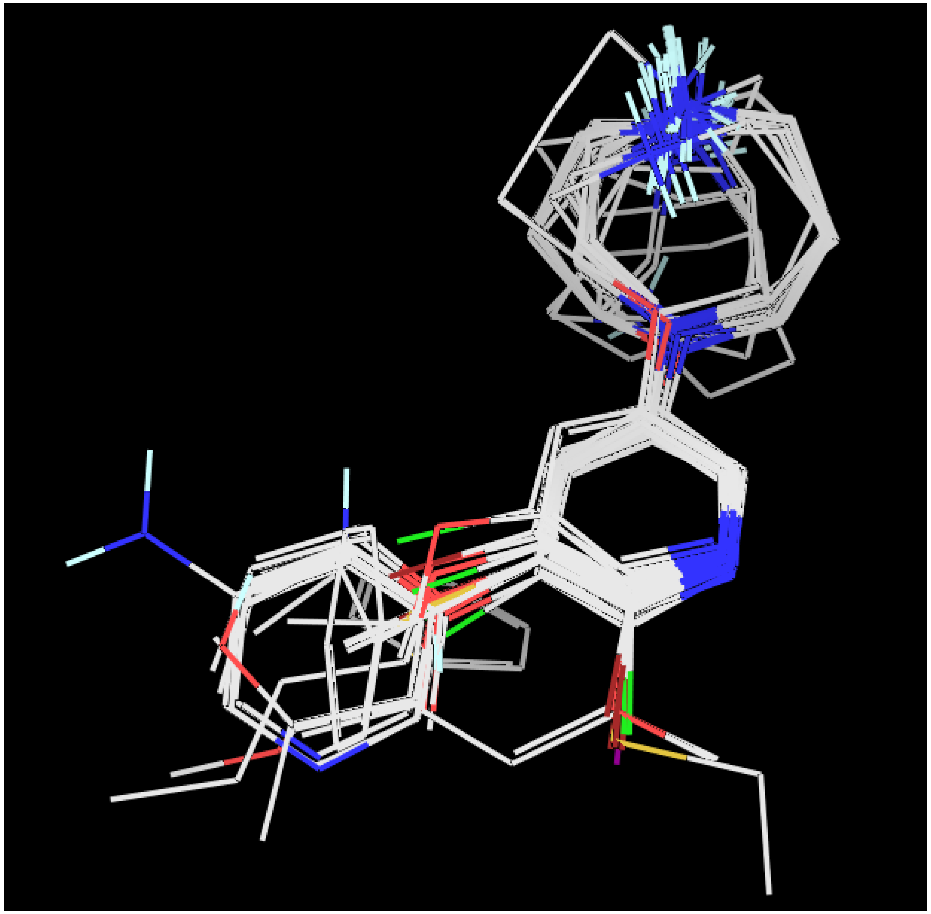
```
from rdkit import Chem
from rdkit.Chem import AllChem

sdf = 'ref_e2.sdf'
o3aSdf = 'ref_e2_PyO3A.sdf'
supplier = Chem.SDMolSupplier(sdf, True, False)
nMol = len(supplier)
refNum = 48
refMol = supplier[refNum]
refPyMP = AllChem.MMFFGetMoleculeProperties(refMol)
o3aMol = Chem.SDWriter(o3aSdf)
print 'N\t\tSCORE\t\tRMSD'
for prbNum in range(0, nMol):
  prbMol = supplier[prbNum]
  prbPyMP = AllChem.MMFFGetMoleculeProperties(prbMol)
  pyO3A = AllChem.GetO3A \
    (prbMol, refMol, prbPyMP, refPyMP)
  print '{0:}\t\t{1:.2f}\t\t{2:.2f}'.format(prbNum + 1, pyO3A.Score(), pyO3A.Align())
  o3aMol.write(prbMol)
o3aMol.close()
```

O3A alignment: before and after

Before

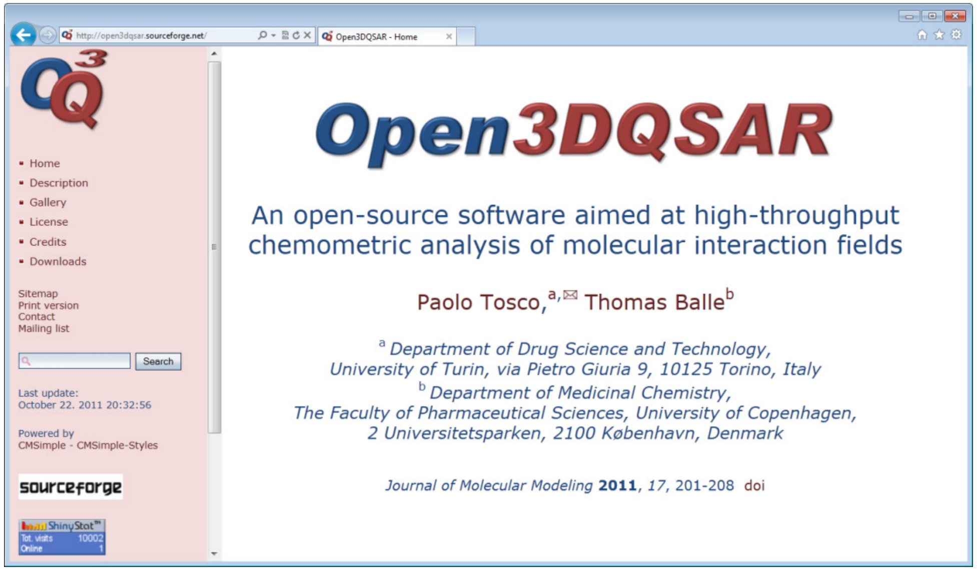


After



What next?

- Methods to perform OOTB multi-conformational and multi-threaded alignment
- Iterative refinement of overlays
- Implementation of O3Q functionality



Last update:
October 22. 2011 20:32:56

Powered by
CMSimple - CMSimple-Styles

Open3DQSAR

An open-source software aimed at high-throughput chemometric analysis of molecular interaction fields

Paolo Tosco,[a,✉] Thomas Balle[b]

[a] *Department of Drug Science and Technology,*
University of Turin, via Pietro Giuria 9, 10125 Torino, Italy
[b] *Department of Medicinal Chemistry,*
The Faculty of Pharmaceutical Sciences, University of Copenhagen,
2 Universitetsparken, 2100 København, Denmark

Journal of Molecular Modeling **2011**, *17*, 201-208 doi

- **Open3DQSAR**
  - MIF computation
  - PLS model building and validation
  - variable selection

Acknowledgments

Thanks

- To Nik and Greg
- To NIBR for funding
- To all scientists who publish their science in such an open, clear and truly reproducible form as Halgren did with MMFF