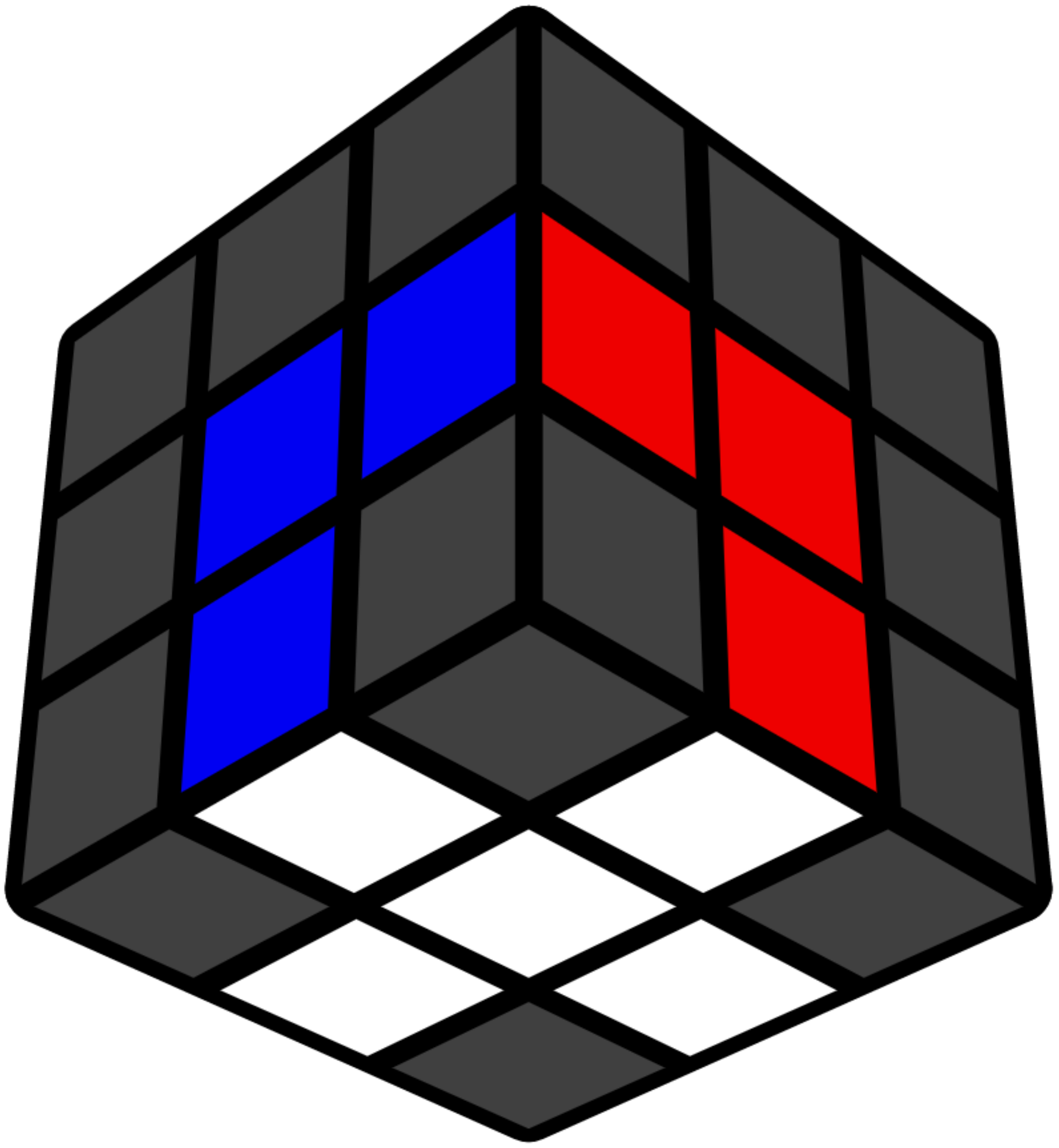
Ribbon



A Speedsolving Method by Justin Taylor

Introduction

This method was developed with the intention of providing a Two Look Last Slot + Last Layer (2LLSLL) option for CFOP and other methods that do not orient the edges first. The method has a total of 266 algorithms in its standard form. However, if one already knows full OLL and PLL, there are only 188 new algorithms to learn. While this is certainly not a light load, this is fewer algorithms than any other true 2LLSLL methods for CFOP. The method also retains the simple recognition, ergonomics, and color neutrality viability of CFOP. The concept can be viewed as its own standalone method in the form of RFTT, or it can simply be the most versatile tool in a CFOP user's toolbox.

Steps of the Method

The primary variant of the method, and the easiest one to adapt to for CFOP solvers, is known as RFTT (pronounced as Rift): Ribbon, F2L, TOLS, TTLL. The steps are as follows:

Ribbon: This is effectively solving 5 edges, planned fully in inspection. The average move count is 6 moves, with a maximum of 9 moves. The bottom cross is solved with any one F2L edge. The slot with this single edge is known as the Ribbon Slot. This can be compared to having an XCross done in a standard CFOP solve. However, as there is no regard for the corner in that slot, further lookahead is possible, and lower move count solutions are also more common. This can be extended further with the use of blockbuilding, allowing potential for XRibbons and beyond to be created. This is similar to creating an XXCross, but is much easier to plan in inspection. Another approach is to build the cross and a few F2L pairs, and simultaneously insert an edge while solving an adjacent pair. This multislotting technique can be used whenever a standard XCross is obvious and is the more efficient route. In conclusion, whenever is the easiest way to insert the edge as seamlessly as possible is the best.

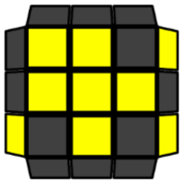
F2L: This consists of solving the other three or fewer slots still left. Most commonly, this will be the next step after the Ribbon Slot is solved. Sometimes, the Ribbon slot is not solved as the first slot, and is inserted while solving another pair. Occasionally, the Ribbon Slot will have an incorrect F2L corner paired with it. The best solution to this is to simply move it out of the way with a D move, replace with a U layer corner, and bring it back. However, by predicting which corner with go with the edge in the Ribbon Slot, this can be avoided in most cases.

TOLS: This stands for Twisted Orientation and Last Slot, and is the major development of the method. This step averages approximately 10 moves, but can take as few as 6, and contains a total of 173 algorithms. The step orients the DFR corner and the entire last layer in one algorithm. TOLS is divided into three sets of algorithms: TOLS+ (U or D sticker of DFR corner is facing forwards), TOLS- (U or D sticker of DFR corner is facing to the right), and TOLSo (U or D sticker of DFR corner is facing down). The first two have 58 algorithms each, and the last has 57. Below, the algorithms and corresponding images for the + and – sets are included. TOLSo is not included, as OLL algorithms can be used instead. The thousands of collective hours poured in to developing OLL algorithms have been proven repeatedly to be extremely quick, so there is no need to learn an additional 57 unnecessary algorithms. Recognition can either be done in two easy steps, or one slightly more difficult step. For the two-step recognition, corner orientation of the top layer is the first to be determined. The first layer corner is ignored, as its orientation can be deduced from the information from the top layer. Then, the edge orientation is recognized from a certain angle. The algorithms below are organized in this style of recognition, for ease of learning. The alternative recognition is to look at the shape of the oriented pieces of the top layer. This is how OLL recognition is commonly taught, and the idea carries on to TOLS. While having triple the cases per shape will make recognition difficult at first, this step can flow as seamlessly as OLL does at its highest level.

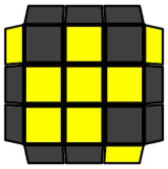
TTLL: This stands for Tran Thompson Last Layer. This step is borrowed from ZZ-CT, a method developed by Chris Tran with the intention of creating a low algorithm count 2LLSLL for the ZZ method. There is an average move count of 15, with the shortest algorithms having only 7 moves. The step contains 72 algorithms that solve the entire cube when every piece is oriented and the DFR corner is replaced with any top layer corner. This is broken down into 6 corner permutations, each with 12 possible edge permutations. Additionally, there is a 1/5 chance of a PLL occurring, due to the possibility of the corner solving itself during TOLS. TTLL is recognized by the block patterns, opposite and adjacent stickers, and any other useful information available, just as PLL is. A set of TTLL algorithms is also included further down in the document.

The method has an average move count of around 47-50 moves in a speedsolve, with one fewer look than traditional CFOP. Additionally, each step has several cases that are very short and fast, leading to excellent lucky single times. Color neutrality is as easy with this method as it is with CFOP. Many of the algorithms are excellent, and over time, will be changed and further improved. This is the first iteration of algorithms published for TOLS, but the majority are already great as they stand.

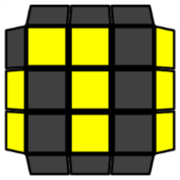
On the next few pages, the algorithms are sorted in order as TOLS+, then TOLS-, and finally TTLL.



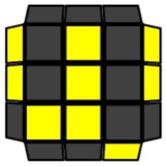
(U) R U R' U2 R U R'



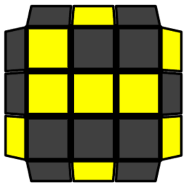
R U2 R' U R U2 R' U2 R U R'



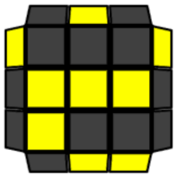
R U' R' F U F' R' F R2 U' R' F'



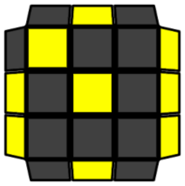
R U2 R' F' U F R U R'



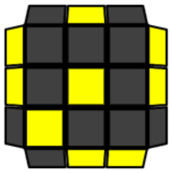
(U') R U R' U' R U2 R2 F R F'



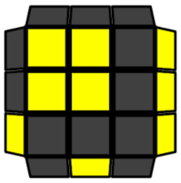
(U') R U' R' F R' F' R2 U R'



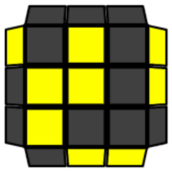
D R' F R U R' U' F2 U' F U2 R D'



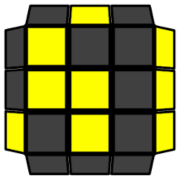
F R' F' R2 U2 R2 F R F'



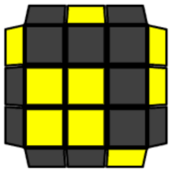
(U) R U' R' U' F' U F R U R'



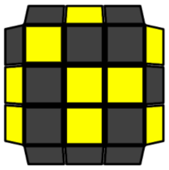
(U2) R U R' F' U2 F U2 R U R'



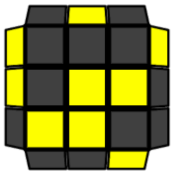
R U2 R' F R' F' R2 U' R' U2 R U R'



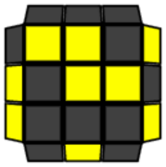
(U2) M' U R U' M U2 R' F' U2 F



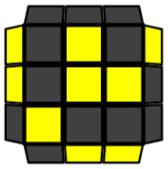
(U') R U R' U R2 U2 F R F' U2 R2



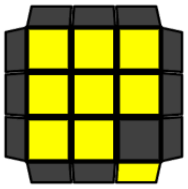
(U') F R U' R' F U F2 U' R' F2 R

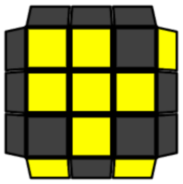


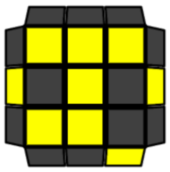
R' F R F' R U2 R' U F' U F

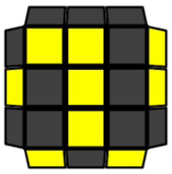


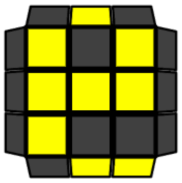
(U2) R' F R F' R U R2 F R F2 U' F

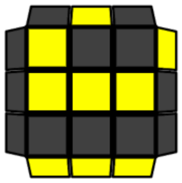

(U) D' R' D R U' R' D' R D

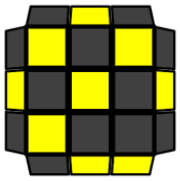

(U2) R U' R' U' R U R' U' R U2 R'

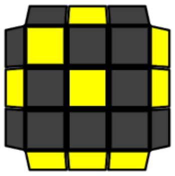

(U) R F R U R' U' F' U R'

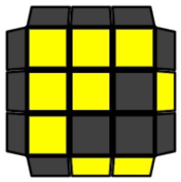

R U2 R' U' F R U R' U' F2 U F

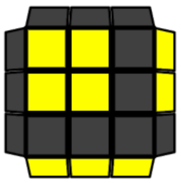

(U) R U' F R2 U R2 U' R2 F' R'

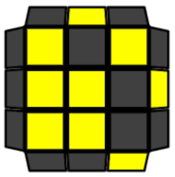

(U) R U' F R2 U R U' R' U R' U' F' R'

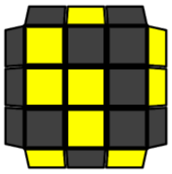

R' U' F U R2 U' R' F2 U' F U R U R'

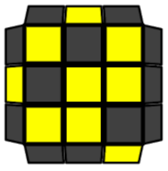

R U' R2 F R F' U2 R' F R F2 U F

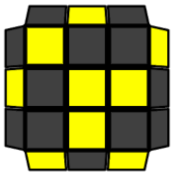

F R' F' R U' R U R'

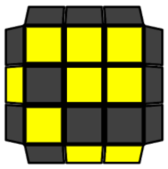

F R' F' R2 U R' U R U' R'

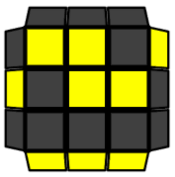

R U2 R' F R' F' R U R U R'

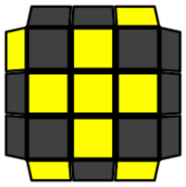

(U2) M' U R U' M U2 R' F' U2 F

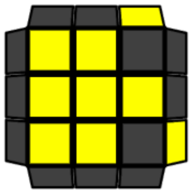

(U) R U2 R' U2 R U2 R2 F R F'

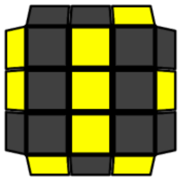

(U') R U' R' U' F' U F U' R U2 R'

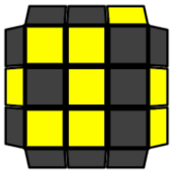

R U' R' U2 F R U R' U' F2 U F

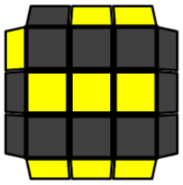

R U' R' U R' U' F U R U' F'

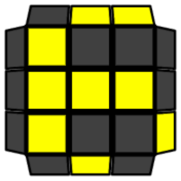

(U) R U2 R' U' R U R' U R U R'

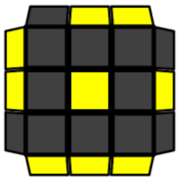

(U) R U2 R’ U2 R U R’ U R U’ R’

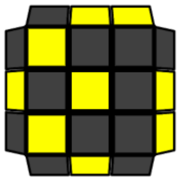

R U2 R2 U2 R U R2 F R2 F' R' U R

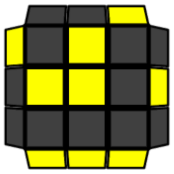

(U') R U R' F' U' F U2 R U R'

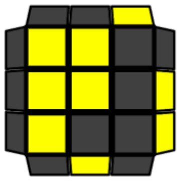

(U') R' U' R2 F R2 F' U' R2 U2 R2 U2 R'

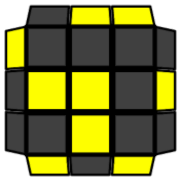

R U2 R' U2 F U R U' R2 F' R

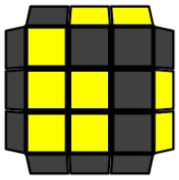

(U') R U' R2 F R F2 U' F U R U R'

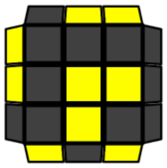

(y' U') R' U2 R' F R F2 U2 F R

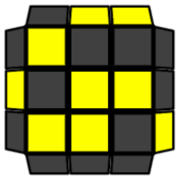

(U) F' U' F2 R' F' R U2 R U R'

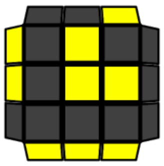

(U2) R U2 R' U2 R U R2 F R F'

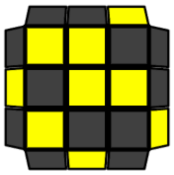

(U') R U2 R2 F R F' U R U R'

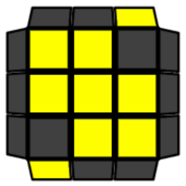

R U2 R' F U R U' R' F' R U2 R'


(y) L' U L U F' L F L2 U2 L

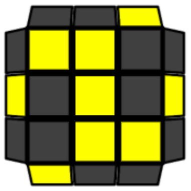

R U R' F' U2 F R U R' U R U R'


R U2 R' F' U2 F U R U' R'

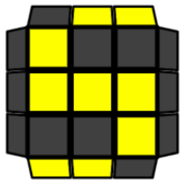

R U R' U' R U2 R' F' U2 F U R U R'



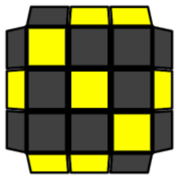
R U2 R' U R U R'



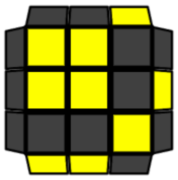
F' U F U2 R U' R' F' U F



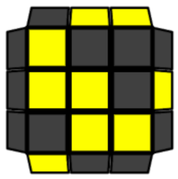
R U2 R' U' D R2 U' F R F' U R2 D'



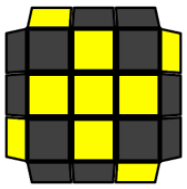
F2 U' R U' M' U M U R' U F2



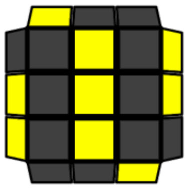
R' U' D' F' U F D R



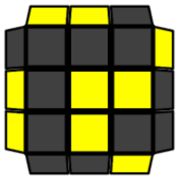
R U' R' U' R U R' F' U' F R U2 R'



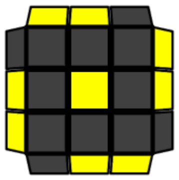
R U' R' U' R U' R' U R U2 R'



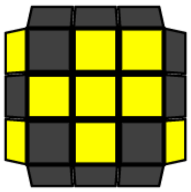
R U' R' U' F' U2 F R U2 R2 F R F'

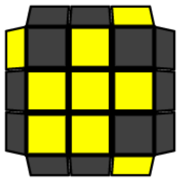


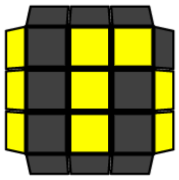
R U R' U' F' U' F U' R U2 R'

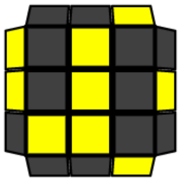


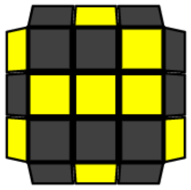
R U2 R' U F' U F R' F R2 U R' U' F'

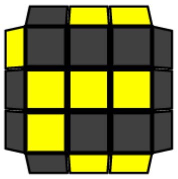

(U2) R U' R' U R U2 R' U R U' R'

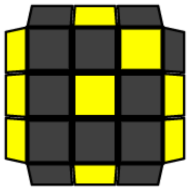

(U') R U' R' U R U' R' U' R U R'

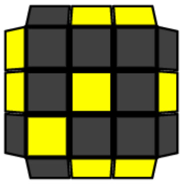

(U2) R' U' R' U' R U R U F R F'

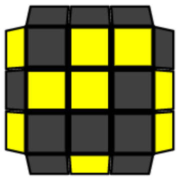

(U') R U' R' F' U' F R U2 R'

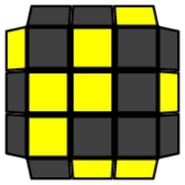

R U' R' F' U' F U R U R'

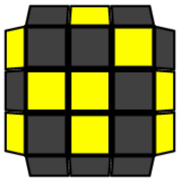

(U2) R' F R U R U' R2 F' R2 U R'

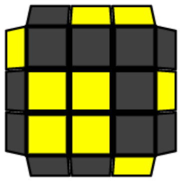

R' F R F' R' F R F' R U2 R2 F R F'

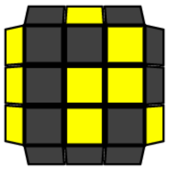

(y U2) F' L F L2 U2 L2 F' L' F

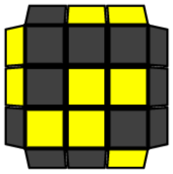

(U) R U2 R' U R' F R F' U R' F R F'

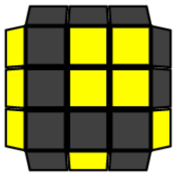

(U') R U' R' U' R U2 R' F R U R' U' F'

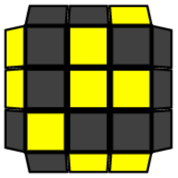

(U2) R U' R' F' U' F R' F R F' R U2 R'

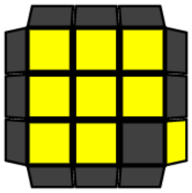

R U R2 U' F' U F R y' R' U R

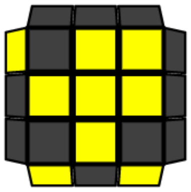

R2 U2 R2 F R F' R' U' R2 U' R' U R'

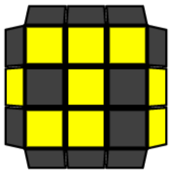

R U R' U' R U2 R2 U' F U R U' F'

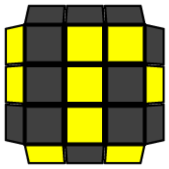

(U) R U R' U2 R U' R' U2 R' F R F'

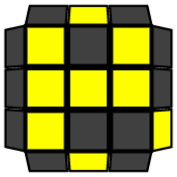

R U' R2 F R2 U R' U' R' F' R

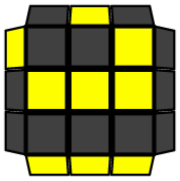

(U) R' D' R U' R' D R

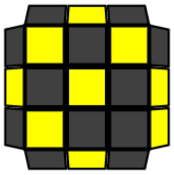

(U') R U' R' U2 R U' R'

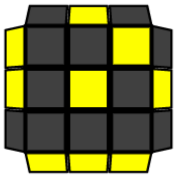

(U) R U' R' U R U R2 F R F'

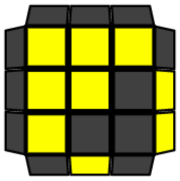

(U') R' F R2 U R' U' F' U R U' R'

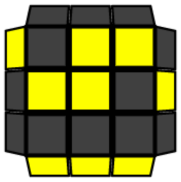

F R U R' U' F' R U' R' U2 R U' R'

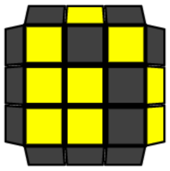

(U) R U2 R' U R U' R' U2 R' F R F'

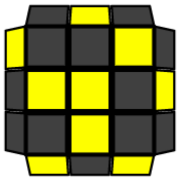

(U2) R U' R2 U' F' U F2 R2 U R' U' R' F' R

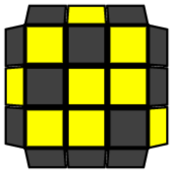

(U2) R2 U2 F R F' U2 x' R U R U'

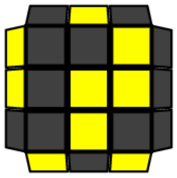

R' F R y' R' U R' U' R

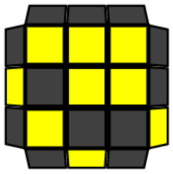

(U) R U' R2 F R F' R U R'

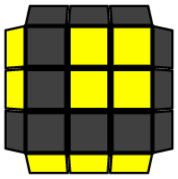

(y' U2) R' U' R U R' F R' F' R2

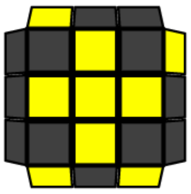

R U' R' U R U2 R2 F R2 U R' U' F'


(y' U') R' U2 R U2 R' U2 R2 x' U' R' U

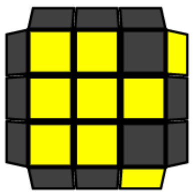

R' D' R U' R' D R U' F U R U' R' F'


(U) R U R' U2 R' F R F' U2 R U' R'

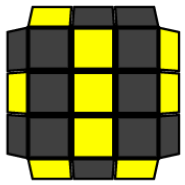

D' R U R' U R' F' R2 U2 R2 F R D



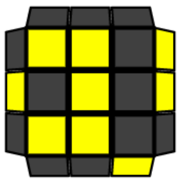
(U') R U R' U R U R' U' R U2 R'



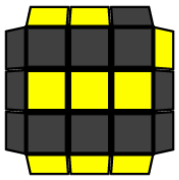
(U') R U' R' U' R U2 R'



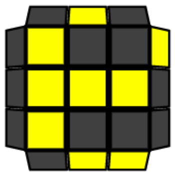
R U R' U2 R U2 R' F' U2 F R U' R'



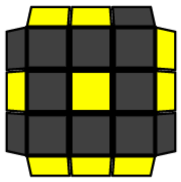
(U2) R U2 R' U R' F R F' R U R'



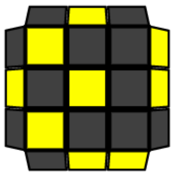
R U2 R2 F R2 U R' U' F' R U2 R'



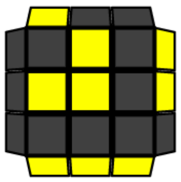
(U) R U2 R' U R' F R2 U R' U' R' F' R



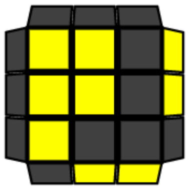
R U R' U' F U R U' R2 F2 U' F U R



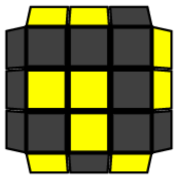
F R U' R' F U2 R U2 R2 F' R U F'



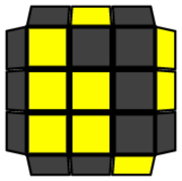
(U') R U R' F R' F' R2 U R' F R' F' R



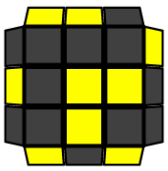
F R' F' R2 U' R' U2 R U2 R'



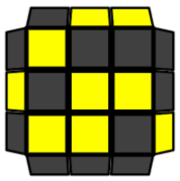
R U R' U' F' U2 F R U2 R'



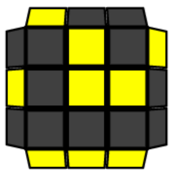
R U2 R' F U R U' R' F' R U2 R'



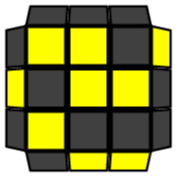
(y) L' U2 L2 F' L' F U' L' U' L



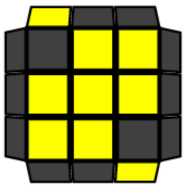
(y' U') R' U2 R U2 R' F R' F' R2



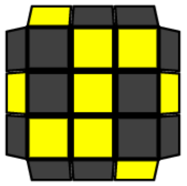
(U2) R U R2 F R F' U2 F' U' F



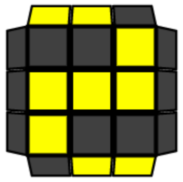
R U R' U' F' U' F U2 R U' R'



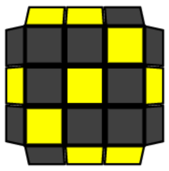
R U R' U' R U' R' U2 R U2 R'



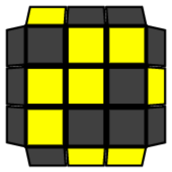
(U) R U' R' F' U' R U2 R' U2 R' F R



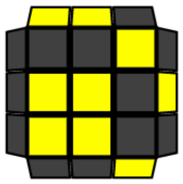
R' F R F' U R U' R' F R' F' R



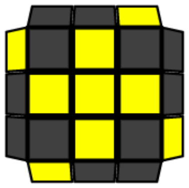
(y) R' D' F R U2 R' F D R2 U' R' y' R U' R'



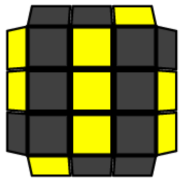
(U2) R U' R' U' F R' F' R U R U R'



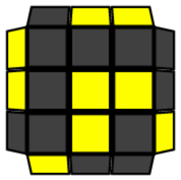
(U) R U' R' F' U F U2 R U2 R'



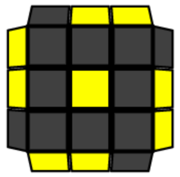
R U2 R' U R U R' U R U2 R' U' R U' R'



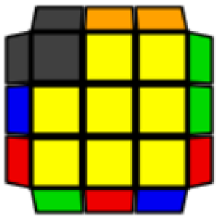
R U2 R' U R' F R F' U2 R U' R'

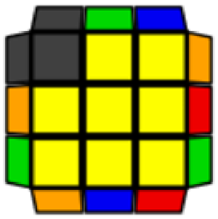


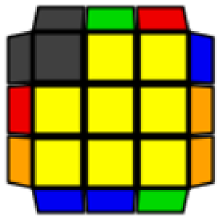
(U') R' U' R2 U' R' U R' U F R F'

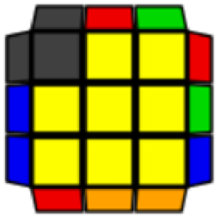


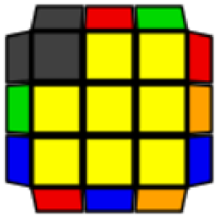
R' F R F' U2 R' F R F' d' L' U' L

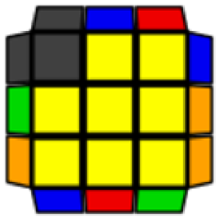

R' D' R U' R2 D' R U' R'
D R U D R

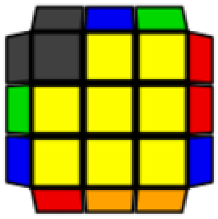

(U') R U R' U R U R' U R
U' R' U R U R' U R U R'

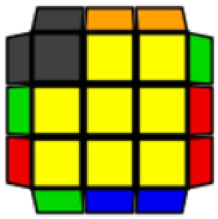

(U) R U' R' U R U2 R' U'
R U R'

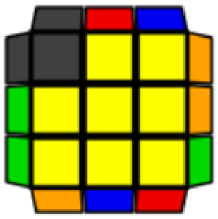

R U' R' U' R U2 R' U2 R
U2 R' U R U R'

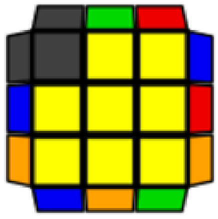

(U') R2 U2 R U' R' U R'
U2 R2 U R U R'

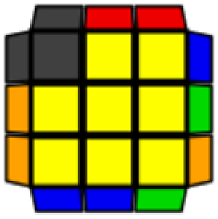

(U2) R U2 R' U' R' U2 R2
U R2 U R

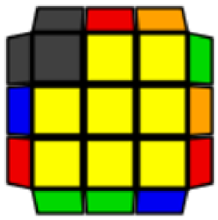

(U2) R2 U R2 U R2 U2
R2

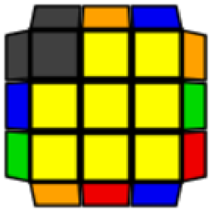

(U) R U' R' U R2 U R' U
R' U' R2 U' R' U2 R'


R2 U2 R2 U' R2 U' R2

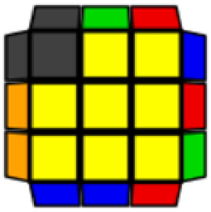

(U) R U' R' U R2 U R' U
R' U' R2 U' R' U2 R'


R' U2 R U2 R U R' U2 R
U R' U' R' U2 R

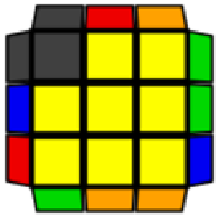

(U) R' U' R2 U' R2 U2 R
U R U2 R'



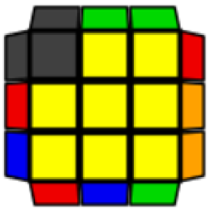
(y') R2 U2 R U' R2 U R U R' U2 R2 U2 R' U R2



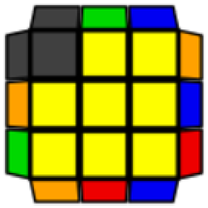
R U' R D R' U R U D' R D R' U' R D' R



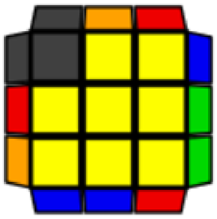
(y' U') R' U R U' R' U2 R U R' U' R



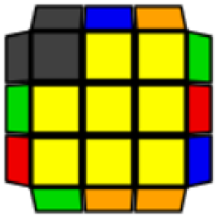
(U') R U R' U' R U' R D R' U2 R D' R' U R' U R U' R'



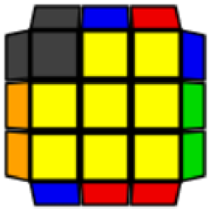
(y' U) R2 U2 R' U R U' R U2 R2 U' R' U' R



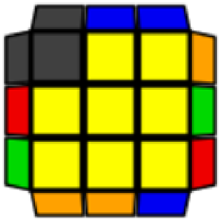
(y' U2) R2 U' R2 U' R2 U2 R2



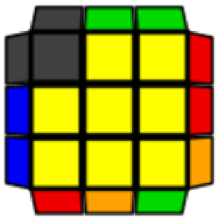
(y' U2) R' U2 R U R U2 R2 U' R2 U' R'



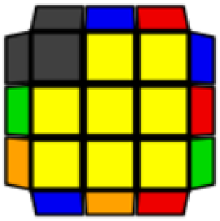
(y' U') R U R U' R' U2 R' U2 R2 U R' U' R'



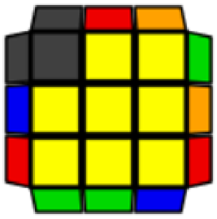
(U') D R2 U2 R2 U R2 U R2 D'



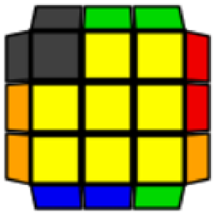
(y' U') R' U R U R2 U2 R' U R' U' R' U R2 U R2



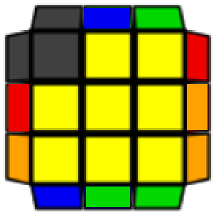
(y') R' U R U R2 U2 R' U R' U' R U2 R2



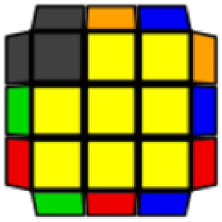
y' U' R U R2 U R2 U2 R' U' R' U2 R



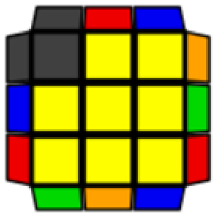
R' D R D' R' U R D2 R' U' R D' R' D' R



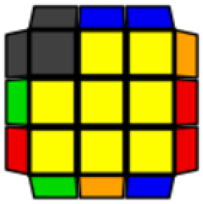
(U2) x R' U R U' R' U R U' R' U R U'



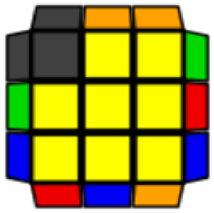
R' U' R' U R2 D' R U2 R' U2 R U' R' D R'



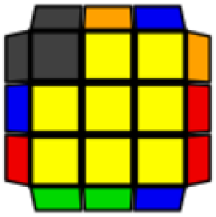
(U') R U R' U' R U2 R D R' U R' U' R U R2 D' R



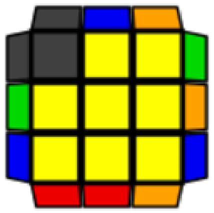
R2 D R2 U2 R2 U R2 U R2 U2 D' R2



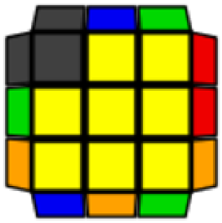
(U') R U2 R' U2 R U2 R' U' R U' R D R' U R D' R' U R'



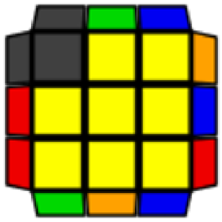
R2 D U2 R2 U' R2 U' R2 U2 R2 D' R2



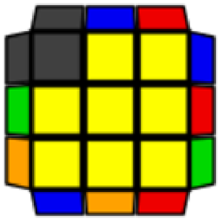
(y') R' U R U R' U R U' R D R' U R D' R' U2 R' U' R



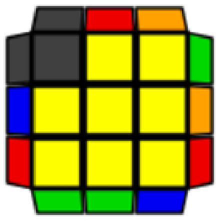
R U' R' U' R U' R D R' U' R D' R' U R' U2 R U R'



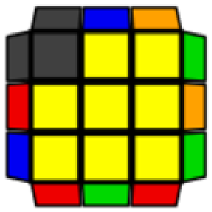
(U2) R U' R' U2 R U' R D R' U R D' R' U R' U R U R'



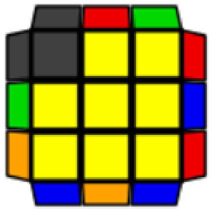
(U y') R2 D' R2 U2 R2 D R2 D' U' R2 U' R2 D



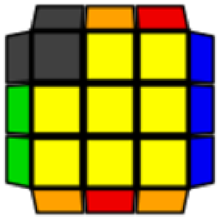
R2 U R2 D U R2 D' R2 U2 R2 D R2 D'



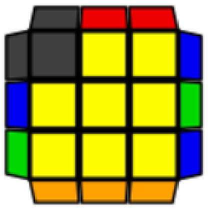
R' D R D' R' U R D2 R'
U' R D' R' D' R



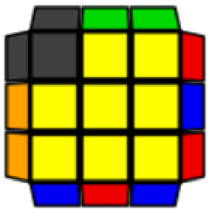
R U' R U' R' U R' U R2 D'
R U R' U' D R



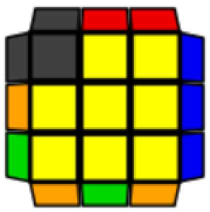
R2 y R U R' U' R' F R2
U' R' U' R U R' F



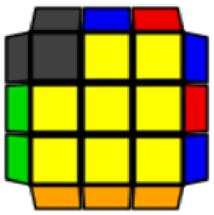
R U R' U' R U' R' U' R U
R D R' U' R D' R' U R' U



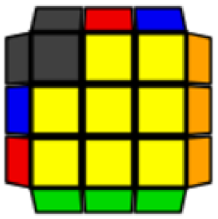
(U) R U F R U R' U' F'
R2 F R F' R U' R'



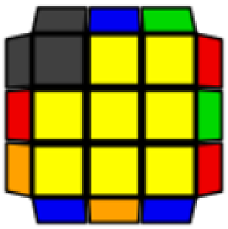
R2 y R U R' F' R U2 R'
U2 R' F R U R U2 R' U'
F2



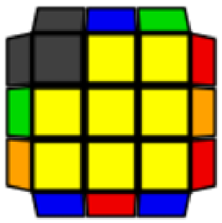
(y) L2 R' U2 R U2 R' F R
U R' U' R' F' R2 U' L2



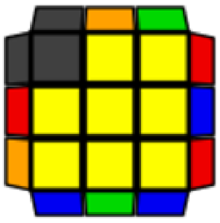
(z’ x) U2 R2 F R F’ R U2
r’ U L



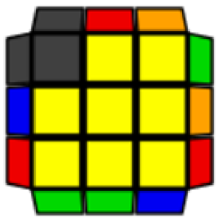
R U2 R U2 D R2 U' R' U
R2 D' R' U' R' U' R2



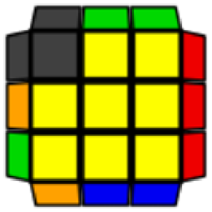
R U' R' D R' U R U' R' U'
R D' R2 U R U' R2



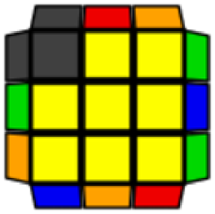
R2 U F R U' R' U R U
R2 F' U' R' U R'



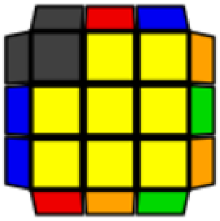
R' U2 R' U R U' D' R U2
R U R U' R2 D R



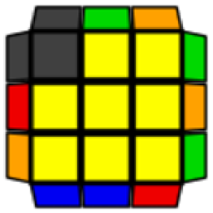
(x' y') D2 U R U' R' D2 R
U R' U' y x



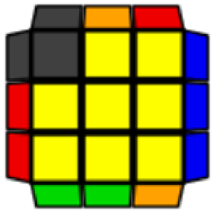
R U2 R' U2 R U' R' U2
R2 D R' U2 R D' R2



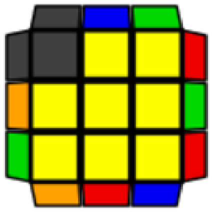
R U R' U2 R2 D R' U R
D' R' U2 R' U R U R'



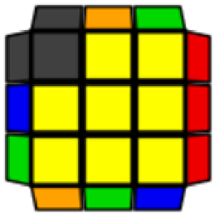
R' D' R U' R' D U' R2 U'
R U R U' R U2 R



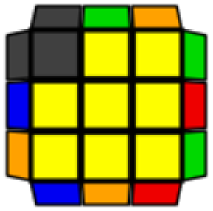
R' U' R' U' R U R D R' U'
R U D' R U R2 U' R'



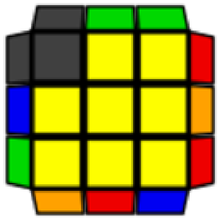
R2 y R U R' F' R U2 R'
U2 R' F R U R U2 R' U'
F2



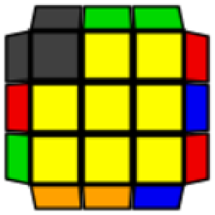
R' U R' y' R2 u' R U' R'
U R' u R2 y R2



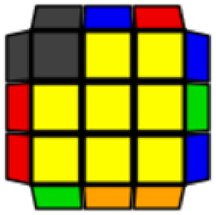
R U2 R' U2 R U' R' U2
R2 D r' U2 r D' R2



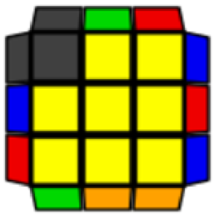
R' U' R D R' U' R D' R'
D' R U R' U D R



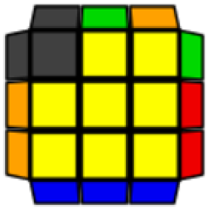
R U' R D R' U2 R D' R'
U2 R' U' R U2 R'



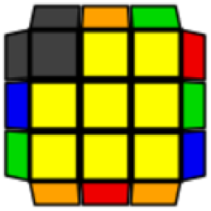
R' L U' R2 U' R' U' R U2
R2 L' U R U



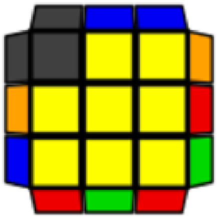
(U) R U2 R' U R U' R U
R' U D' R U R' D R U R2



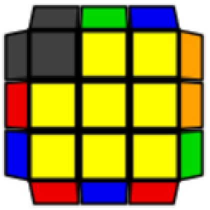
(y x') D2 R U R' D2 R U' R' x y'



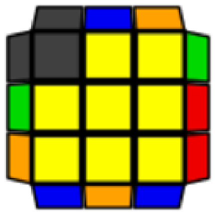
(U') R' D R2 D' R' U2 R D R2 D' R U' R' D R U' R' D' R



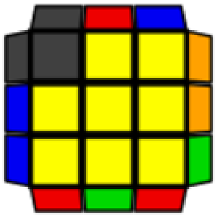
(y') R' U2 R2 U' D R' U' R D' R2 U R' U R U' R2



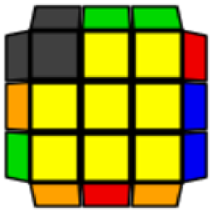
R U R' U' R U' R' U' R U R' U' R' D' R U R' D R



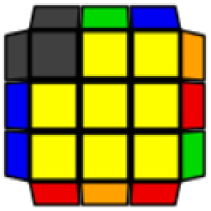
(U') R' U' D' R U' R' D R D R' U R D' R' U R U



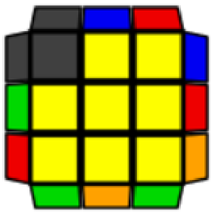
(y') R U R' D' R U R' D R D R' U' R D' U' R'



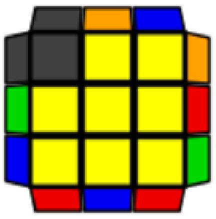
(U') R U2 R' U R U R' U' L U' R U L' U' R'



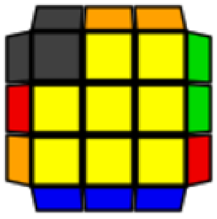
R U2 R' U2 R U' R' U2 R2 D r' U2 r D' R2



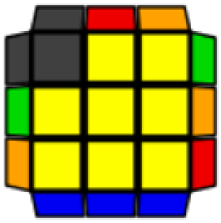
y' R' U2 R U2 R' U R U2 R2 D' r U2 r' D R2



R U2 R' U2 R U' R' U2 L' U R U' L U2 R'



F2 U' R U R' F' R U R' U' R' F R2 U' R' U' F2



F U' F R2 u R' U R U' R u' R2 F2