Yuxuan’s OLL Algorithm List

This document contains a list of every OLL algorithm that Yuxuan uses for normal 3x3 solves with a short description of how she learned/remembers the algorithm.

Note 1: All images and some algorithm names/categories are taken from the speedsolving wiki at https://www.speedsolving.com/wiki/index.php/OLL

Note 2: Most names are pretty arbitrary.

Note 3: The pictures will be presented in the angle you are supposed to hold the cube at to perform the algorithm. There will be a U/U2/U’ in parenthesis in front if a specific algorithm that I use happens to be from a different angle than the one shown.

Note 4: If there is more than one algorithm listed, the one that I use the most will be first. All other algorithms will be for the specific cases that are listed in the comments.

Note 5: All algorithms will also be broken down into triggers (a sequence of moves that is fast and easy to execute). Each case will have two cells in the algorithm section. The first cell will have the normal algorithm(s). The second cell will have the same algorithm(s) broken down into triggers.

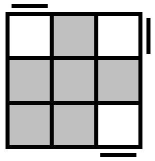
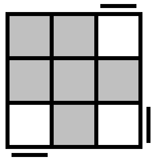
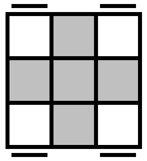
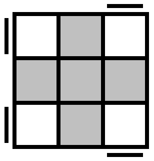
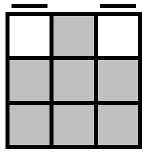
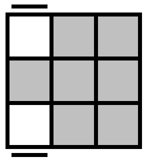
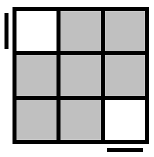
Note 6: For the most part, all algorithms will be written the way I execute them (some double turns will have a ‘ even though a 180 degree turn is the same whether the layer was turned clockwise or counter clockwise).

Note 7: There will be a separate document for 2 look OLL or 4 look last layer (4LLL), which is going to be a subset of these algorithms. There will also be a separate document for easy OLL’s that you will learn from just learning 2 look OLL.

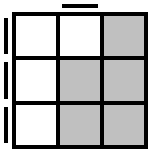
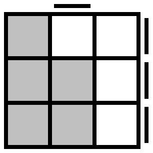
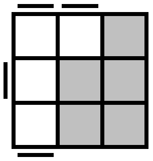
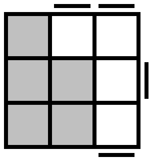
Note 8: My philosophy for learning algorithms was/is picking algorithms that are easy to learn (even if it might mean it is more moves/slower) so many of my algorithms build on each other.

Note 9: If there are any problems with the algorithms, contact Yuxuan.

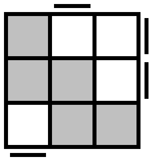
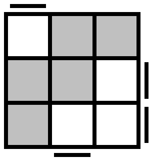
All edges oriented

OLL	Algorithm(s)	Comments
	R U R’ U R U2’ R’ With triggers: (R U R’ U) (R U2’ R’)	“Sune”
	R’ U’ R U’ R’ U2’ R With triggers: (R’ U’ R U’) (R’ U2’ R)	“Antisune”
	1. R U2’ R’ U’ R U R’ U’ R U’ R’ 2. F (R U R’ U’)3 F’ (expanded: F R U R’ U’ R U R’ U’ R U R’ U’ F’) With triggers: 1. (R U2’ R’) U’ (R U R’ U’) (R U’ R’) 2. F (R U R’ U’)3 F’	“Double Sune” 1. Happens to solve corners if UBR == UFR and UBL == UFL 2. Solves corners if UBL == UBR and UFL == UFR
	R U2’ R2’ U’ R2 U’ R2’ U2’ R With triggers: (R U2’) (R2’ U’) (R2 U’) (R2’ U2’) R	“Pi” Solves corners if UBR == UFR (and UBL and UFL are opposite colors)
	1. R2 D’ R U2’ R’ D R U2 R 2. L2’ D L’ U2 L D’ L’ U2 L’ 3. (U2) R U R’ U R U2’ R2’ U’ R U’ R’ U2’ R With triggers: 1. R2 D’ (R U2’ R’) D (R U2 R) 2. L2’ D (L’ U2 L) D’ (L’ U2 L’) 3. (U2) (R U R’ U) (R U2’ R’) (R’ U’ R U’) (R’ U2’ R)	“Headlights”, “U” 1. Solves corners if UBL == FUR 2. Solves corners if UBR == FUL 3. Solves corners if UFL and UFR are opposite colors and the back 2 corners are solved. Algorithm is basically sune + antisune
	r U R’ U’ r’ F R F’ With triggers: (r U R’ U’) (r’ F R F’)	“Chameleon” Very similar to the case below
	F’ r U R’ U’ r’ F R With triggers: F’ (r U R’ U’) (r’ F R)	“Triple Sune”, “Bowtie” Very similar to the case above (move the last move to the front)

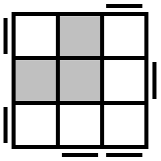
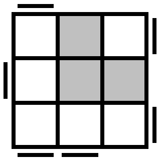
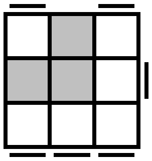
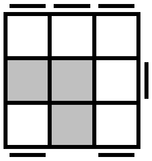
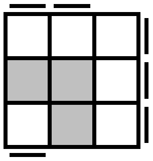
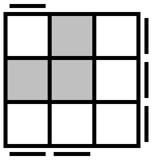
P shape

OLL	Algorithm(s)	Comments
	1. f R U R’ U’ f’ 2. (U2) F U R U’ R’ F’ With triggers: 1. f (R U R’ U’) f’ 2. (U2) F (U R U’ R’) F’	Wide fruruf 2 algs are identical (different angles)
	1. f’ L’ U’ L U f 2. (U2) F’ U’ L’ U L F With triggers: 1. f’ (L’ U’ L U) f 2. (U2) F’ (U’ L’ U L) F	Lefty wide fruruf 2 algs are identical (different angles)
	S R U R’ U’ R’ F R f’ With triggers: S (R U R’ U’) (R’ F R f’)	S move + T shape OLL
	S’ L’ U’ L U L F’ L’ f With triggers: S’ (L’ U’ L U) (L F’ L’ f)	Lefty version of the case above

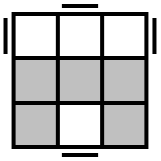
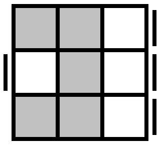
W shape

OLL	Algorithm(s)	Comments
	R’ U’ R U’ R’ U R U R B’ R’ B With triggers: (R’ U’ R U’) (R’ U R U) (R B’ R’ B)	Last 4 moves executed like l U’ R’ U
	R U R’ U R U’ R’ U’ R’ F R F’ With triggers: (R U R’ U) (R U’ R’ U’) (R’ F R F’)	

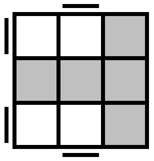
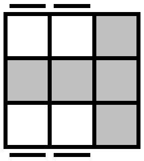
L shape

OLL	Algorithm(s)	Comments
	F R U R’ U’ R U R’ U’ F’	fruruf x2
	With triggers: F (R U R’ U’) (R U R’ U’) F’	
	F’ L’ U’ L U L’ U’ L U F	Lefty fruruf x2
	With triggers: F’ (L’ U’ L U) (L’ U’ L U) F	
	r U2’ R’ U’ R U R’ U’ R U’ r’	Wide double sune
	With triggers: r U2’ (R’ U’ R U) (R’ U’ R) U’ r’	
	r’ U2’ R U R’ U’ R U R’ U r	Wide opposite double sune
	With triggers: r’ U2’ (R U R’ U’) (R U R’) U r	
	R B’ R2’ F R2 B R2’ F’ R	Fun to execute It’s easier if you do the F’ with your left ring finger
	With triggers: (R B’) (R2’ F) R2 (B R2’) (F’ R)	
	R’ F R2 B’ R2’ F’ R2 B R’	Fun to execute Very similar to the case above It’s easier if you do the F’ with your left ring finger
	With triggers: (R’ F) (R2 B’) R2’ (F’ R2) (B R’)	

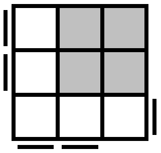
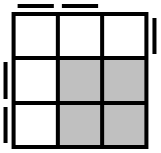
C shape

OLL	Algorithm(s)	Comments
	R U R’ U’ B’ R’ F R F’ B’	First B’ is executed with right ring finger There is a regrip between the B’ and R’ Last 2 moves (F’ B’) is executed with left index and ring fingers
	With triggers: (R U R’ U’) B’ (R’ F R F’) B’	
	R’ U’ R’ F R F’ U R	
	With triggers: R’ U’ (R’ F R F’) U R	

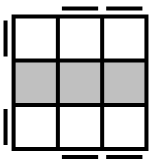
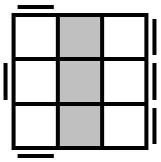
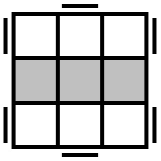
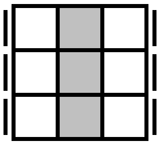
T shape

OLL	Algorithm(s)	Comments
	F R U R’ U’ F’	fruruf
	With triggers: F (R U R’ U’) F’	
	R U R’ U’ R’ F R F’	I call this the T OLL Also happens to be the second half of Y perm
	With triggers: (R U R’ U’) (R’ F R F’)	

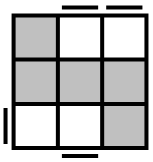
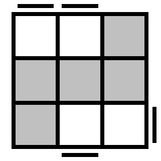
Square shape

OLL	Algorithm(s)	Comments
	r U2’ R’ U’ R U’ r’	Wide opposite sune
	With triggers: r U2’ (R’ U’ R U’) r’	
	r’ U2’ R U R’ U r	Wide opposite antisune
	With triggers: r’ U2’ (R U R’ U) r	

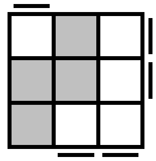
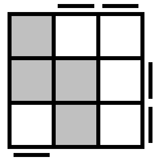
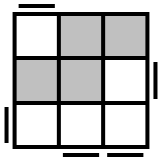
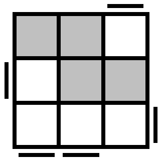
I shape

OLL	Algorithm(s)	Comments
	f R U R’ U’ R U R’ U’ f’	Another fruruf x2
	With triggers: f (R U R’ U’) (R U R’ U’) f’	
	R U R’ U R U’ y R U’ R’ F’	
	With triggers: (R U R’ U) (R U’) y (R U’ R’ F’)	
	F R U R’ U’ F’ r U R’ U’ r’ F R F’	fruruf + chameleon
	With triggers: F (R U R’ U’) F’ (r U R’ U’) (r’ F R F’)	
	r U2’ R2’ F R F’ U2’ r’ F R F’	Wide version of the first dot OLL
	With triggers: (r U2’) (R2’ F R F’) U2’ (r’ F R F’)	

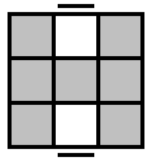
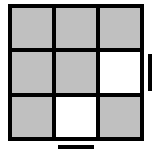
Big lightning bolt shape

OLL	Algorithm(s)	Comments
	R’ F R U R’ U’ F’ U R	R’ + fruruf + solve back right block
	With triggers: R’ F (R U R’ U’) F’ U R	
	L F’ L’ U’ L U F U’ L’	Lefty version of previous case
	With triggers: L F’ (L’ U’ L U) F U’ L’	

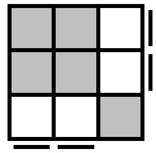
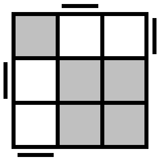
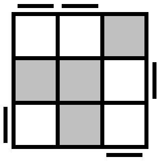
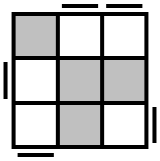
Small lightning bolt shape

OLL	Algorithm(s)	Comments
	r U R’ U R U2’ r’ With triggers: (r U R’ U) (R U2’ r’)	Wide sune
	r’ U’ R U’ R’ U2’ r With triggers: (r’ U’ R U’) (R’ U2’ r)	Wide antisune
	r U R’ U R’ F R F’ R U2’ r’ With triggers: (r U R’ U) (R’ F R F’) (R U2’ r’)	
	l’ U’ L U’ L F’ L’ F L’ U2 l With triggers: (l’ U’ L U’) (L F’ L’ F) (L’ U2 l)	Lefty version of previous case

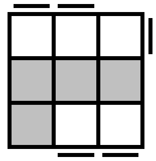
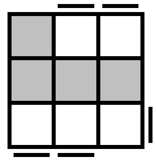
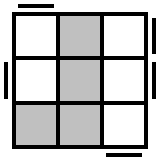
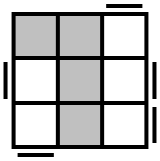
All corners oriented

OLL	Algorithm(s)	Comments
	R U R’ U’ M’ U R U’ r’ With triggers: (R U R’ U’) M’ (U R U’ r’)	H OLL Extremely rare case: if everything else is solved (2 flip), do (M’ U’)4 U’ (M’ U’)4
	r U R’ U’ M U R U’ R’ With triggers: (r U R’ U’) M (U R U’ R’)	Opposite H OLL (I use this a lot in my other OLL’s) M usually executed as r’ R Extremely rare case: if everything else is solved, do (R U R’ U) (M’ U’)4 U’ (M’ U’)4 (R U’ R’)

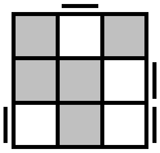
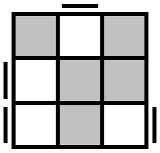
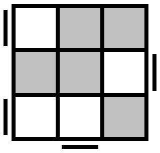
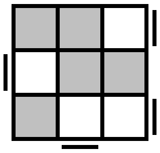
Fish shape

OLL	Algorithm(s)	Comments
	F R U’ R’ U’ R U R’ F’ With triggers: F (R U’) (R’ U’ R U) R’ F’	First half of Y perm
	R U2’ R2’ F R F’ R U2’ R’ With triggers: (R U2’) (R2’ F R F’) (R U2’ R’)	
	R U R’ U R’ F R F’ R U2’ R’ With triggers: (R U R’ U) (R’ F R F’) (R U2’ R’)	
	L’ U’ L U’ L F’ L’ F L’ U2 L With triggers: (L’ U’ L U’) (L F’ L’ F) (L’ U2 L)	Lefty version of previous case

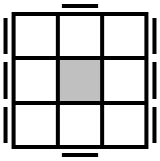
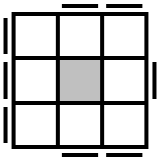
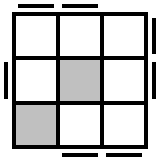
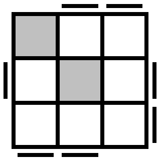
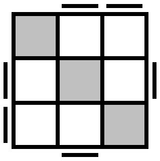
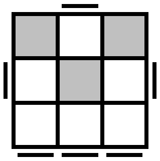
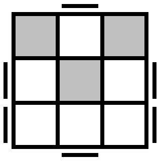
Knight move shape

OLL	Algorithm(s)	Comments
	R U R’ U R U2’ R’ U’ r U R’ U’ M U R U’ R’ With triggers: (R U R’ U) (R U2’ R’) U’ (r U R’ U’) M (U R U’ R’)	Sune + opposite H OLL
	R’ U’ R U’ R’ U2’ R U2’ r U R’ U’ M U R U’ R’ With triggers: (R’ U’ R U’) (R’ U2’ R) U2’ (r U R’ U’) M (U R U’ R’)	Antisune + opposite H OLL
	R U R’ U R U2’ R’ U r U R’ U’ M U R U’ R’ With triggers: (R U R’ U) (R U2’ R’) U (r U R’ U’) M (U R U’ R’)	Sune + opposite H OLL
	R’ U’ R U’ R’ U2’ R r U R’ U’ M U R U’ R’ With triggers: (R’ U’ R U’) (R’ U2’ R) (r U R’ U’) M (U R U’ R’)	Antisune + opposite H OLL

Awkward shape

OLL	Algorithm(s)	Comments
	M U R U R’ U’ R’ F R F’ M’ With triggers: M U (R U R’ U’) (R’ F R F’) M’	M U + T OLL
	M U’ L’ U’ L U L F’ L’ F M’ With triggers: M U’ (L’ U’ L U) (L F’ L’ F) M’	M U’ + lefty T OLL
	R’ F R F’ R’ F R F’ R U R’ U’ R U R’ U’ With triggers: (R’ F R F’) (R’ F R F’) (R U R’ U’) (R U R’ U’)	Lots of moves but it’s fast double sledge + double sexy
	L F’ L’ F L F’ L’ F L’ U’ L U L’ U’ L U With triggers: (L F’ L’ F) (L F’ L’ F) (L’ U’ L U) (L’ U’ L U)	Lefty version of previous case

No edges flipped correctly

OLL	Algorithm(s)	Comments
	R U2’ R2’ F R F’ U2’ R’ F R F’ With triggers: (R U2’) (R2’ F R F’) U2’ (R’ F R F’)	
	F R U R’ U’ S R U R’ U’ f’ With triggers: F (R U R’ U’) S (R U R’ U’) f’	fruruf + other fruruf (F’ f turns into an S) F R U R’ U’ F’ f R U R’ U’ f’
	r U R’ U R U2’ r’ U r U R’ U’ M U R U’ R’ With triggers: (r U R’ U) (R U2’ r’) U (r U R’ U’) M (U R U’ R’)	Wide sune + opposite H
	r’ U’ R U’ R’ U2’ r2 U R’ U’ M U R U’ R’ With triggers: (r’ U’ R U’) R’ U2’ (r2 U R’ U’) M (U R U’ R’)	Wide antisune + opposite H
	R U R’ U R’ F R F’ U2’ R’ F R F’ With triggers: (R U R’ U) (R’ F R F’) U2’ (R’ F R F’)	
	r U R’ U R U2’ r2’ U’ R U’ R’ U2’ r With triggers: (r U R’ U) (R U2’) (r2’ U’ R U’) (R’ U2’ r)	Wide opposite sune + wide opposite antisune
	r’ U2’ R U R’ U r2’ U2’ R’ U’ R U’ r’ With triggers: (r’ U2’ R U) (R’ U) (r2’ U2’ R’ U’) (R U’ r’)	Wide opposite antisune + wide opposite sune