Yo Notation:

A turn-based encoding notation system for memorizing 3x3 algs (non-trivial algs only)

Yo notation is a system I developed in 2018. The main aim is to make algs easier to encode and chunk into letter pairs and letter quads. The face turns and slice turns are chunks that we form images of and memorize.

Singmaster notation is good to generate scrambles and stuff, but not good enough for memorization. Yo notation is a way to encode face turn, slice turns and wide turns of a 3x3 uniquely as a letter. The main objective of creating such an encoding system is that it makes learning algorithms much easier. In most of the speedsolving methods, the algorithm can be broken down into triggers like R U R' U', R' F R F' etc, and most of the cases in blindsolving can be broken down into commutator notation [A: [B, C]], which makes it easier to remember as the algorithm makes sense and there is a pattern that we get out of it. For 5-style edges, a lot of the algs can be decomposed into commutators and nested commutators, but it cannot be expressed in a few triggers as there are just too many types of algs. So, there was a need to create a system which makes memorizing algorithms easier, as each face turn can be a unique letter U=a, U'=b, U2=c and so on, and the entire algorithm string can be broken down into groups of 4 to be memorized as images. So for example, R U2 R' D, becomes jckd in Yo notation, which is just the letter quad for justin kid image.

Video link: Yo Notation Playlist

U - A



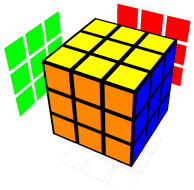
U' - B



U2 - C



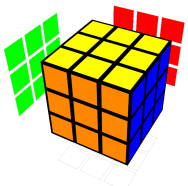
D - D



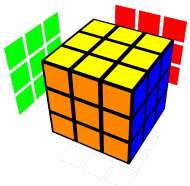
D' - E



D2 - F



L - G



L' - H



L2 - I



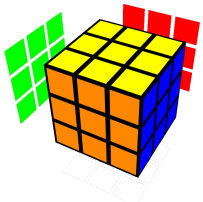
M - M



M' - N



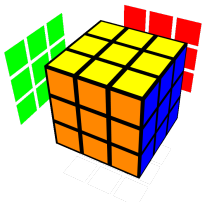
M2 - O



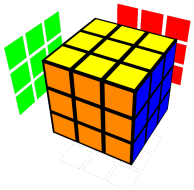
R - J



R' - K



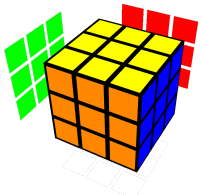
R2 - L



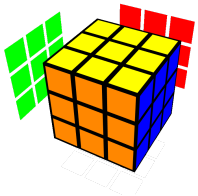
S - S



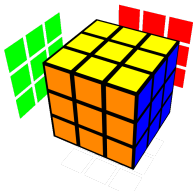
S' - T



S2 - U



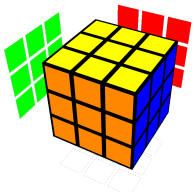
E - P



E' - Q



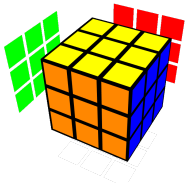
E2 - R



F - V



F' - W



F2 - X



B - Y

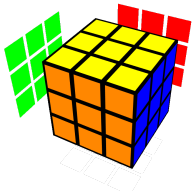


B' - Z

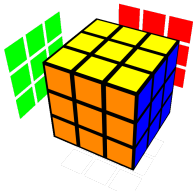


And for wide moves,

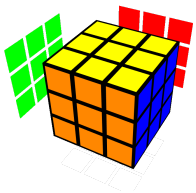
u - QA



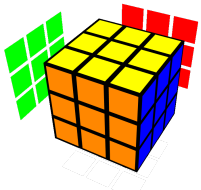
u' - BP



u2 - RC



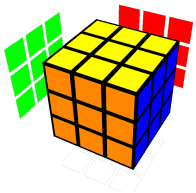
f - SV



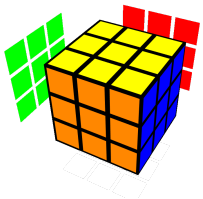
f' - WT



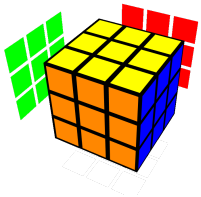
f2 - XU



l' - NH



l - MG



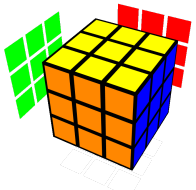
l2 - OI



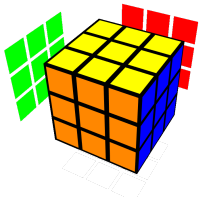
r - NJ



r' - MK



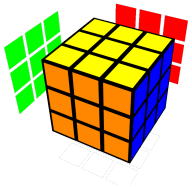
r2 - OL



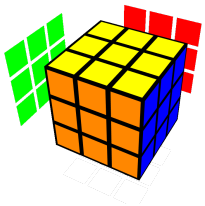
x - LI



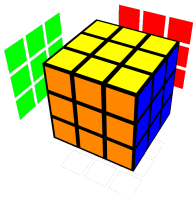
x' - MI



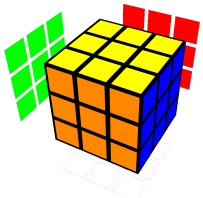
y - GI



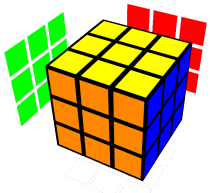
y' - KJ



z - QP



z' -DP



(I have chosen these letter pairs as these were rarer in the algorithm string and had strong imagery in my letter pair scheme. You can choose otherwise according to your personal preference.)There are also an insane amount of cancellations I have come up with, but it will be pretty advanced if you are trying this for the first time. This system takes time to get accustomed to. For fun sake, I have decided to name this system as Yo notation.

2. Memorizing commutators

Translating the entire move sequence doesn't work well in the case of a commutator.
Commutator Doc reference

For example,
AB - [R' D' U' : [R' D R, U']]
Extended Yo notation translation- kebk djbk ejcdj
Shortcut - (keb,kdjb) : from the first element inside the bracket we get to know the setup moves, and the second element becomes the insertion and interchange move. With a little bit of training, we can find out 'kdj' is an insertion and 'a' is the interchange move, and the whole sequence reads [R' D R, U'].

3. Memorizing 5-style edge algs

Remembering algorithms via triggers will work in the 5-cycle case (oiag) : [U : [M, F]]
but not in the case (dula): F2 M' F' E2 F' L' F E2 F' l, which have some 3 move insertion in its sequence but no triggers or straightforward [A, B] commutator form inside it. So, it is best to memorize 'dula' as 'xnwr whvr wmg', from which we can form 3 images and memorize the sequence without having a mental note to take care of. According to my personal experience, reading off 'Yo notation' is not natural at first and is roughly two times slower than reading off the standard R U R' U' notation. We just need to be patient and stick with 'Yo Notation' until we become bilingual in reading off Cube Face Turns!

4. Yo Notation for bigger puzzles (5x5)

Smaller letters are used for inner slices, and for wide moves, the small letters and Capital letters are appended together. For example, D2 F2 2D r d' 2B d r' 2D' R D' 2B' D R' F2 D2 becomes
FXd JjEe yDd Kk eJEz DKXF, quite tough to memorize as there are a lot of capital and non-capital letters in the string. No quad is consistent.

More information in this SS thread:
https://www.speedsolving.com/threads/yo-notation-for-big-cubes.79871/
Key Move (For Tao Yu's alg trainer, modified for 5-style)

```
["j", "R"],
["k" , "R'"],
["a" , "U"],
["b" , "U'"],
["v" , "F"],
["w" , "F'"],
["y" , "B"],
["z" , "B'"],
["g" , "L"],
["h" , "L' "],
["d" , "D"],
["e" , "D'"],
["u" , "r"],
["," , "r'"],
["c" , "l"],
["r" , "l'"],
["m" , "M"],
["m" , "M"],
["n" , "M'"],
["o" , "x"],
["." , "x'"],
["z" , "y"],
["[" , "z"],
[";" , "z'"],
["/" , "y'"],
["s", "S F'"],
["t", "S' F"],
["p", "E"],
["q", "E'"]];
```

It will take a while before you can get used to it and comfortably solve the virtual cube.
I have also added ways to memorize wide moves and rotations, which was not covered in the video. On the alg trainer the keys are remapped as per the Yo notation mostly. Note that double turns will have to be done as two quarter turns and they are not programmed differently. (eg. R2 is R R, or on kb jj)

About Yo

I have used this system to memorize ZBLL algset as well (sighted advanced CFOP algset). I use the Roux method and new other methods (MethNeu) to solve a 3x3. Although I am currently creating a set called YBLL in Yo Notation, it will be more focussed towards solving 3BLD parity rather than LL of 3x3 speedsolve.
If you have any further doubts or need help or clarification, drop a mail to 5stylerepertoire@gmail.com

Check out the Mega Manual for more details on LQs, Yo notation usage and UF5 algs. SS threads dedicated to this topic: Link 1 Link 2 Link 3.