SOLVING THE CUBE – PT. 3

MICHAEL GEORGE

WCA ID: 2015GEOR02

THE LAST LAYER

Last Layer (**LL**) consists of 2 steps

1. Orientation (**OLL**) – Edge Orientation (**EOLL**) + Corner Orientation (**OCLL**)
2. Permutation (**PLL**) – Corner Permutation (**CPLL**) + Edge Permutation (**EPLL**)

Trivia – Your cube is likely to be unsolvable if disassembled and randomly reassembled!

- The probability that your cube can be solved after being randomly reassembled is **1/12**
- You will only realise your cube is unsolvable whilst trying to solve the last layer
- There are three possible causes of an unsolvable cube
  - **EOLL "parity"** – e.g. Flipped edge – **1/2** chance of being correct
  - **OCLL "parity"** – e.g. Twisted corner – **1/3** chance of being correct
  - **PLL "parity"** – e.g. Two edges or corners swapped – **1/2** chance of being correct

ORIENTATION OF THE LAST LAYER (OLL)

Approach

- Orientation of the Last Layer (**OLL**) will be broken into in two sub-steps
  1. Edge Orientation of the Last Layer (**EOLL**)
  2. Orienting Corners of the Last Layer (**OCLL**)
- Manipulation of the **F2L** can be used to change the orientation of the last layer
  - e.g. Extracting an **F2L pair** then re-inserting it using a different "**trigger**"
- This approach requires two simple "**algorithms**" which are essentially combinations of "**triggers**"
  1. Algorithm for edge orientation (**EOLL**) – **F R U** moves
  2. Algorithm for corner orientation (**OCLL**) – **R U** moves

EDGE ORIENTATION OF THE LAST LAYER (EOLL)

There are 4 possible cases during **EOLL** including the "**solved**" case

Cases: There can only be 2 or 4 "**flipped**" edges. An odd number of "**flipped**" edges cannot be solved!

Probabilities: The "**adjacent edge flip**" is the most common **EOLL** at **4/8** solves (i.e. **50%** of the time)

Approach: A single algorithm can be used to cycle through the **EOLL** cases

1/8

2/8

4/8

1/8

EOLL ALGORITHM

- To change the orientation of edges you must use a mixture of **F R U** moves
- The **EOLL** algorithm that will be used is **F (U R U' R') F'**
  - Notice how it includes the **U R U' R'** trigger that was used during **F2L**
  - The **F** is referred to as a "**setup**" move and the **F'** will undo the setup
  - The algorithm "**flips**" two LL edges (UF and UR) but it also has a side effect of "**permuting**" LL edges and corners
- Explanation of the algorithm
  - The **F** move is turning the front layer to create a "**pseudo slot**" at the front-right of the **F2L**
  - The **U R U' R'** trigger exchanges pieces between the U-layer and the "**pseudo slot**"
  - The **F'** move restores the front layer and thus the **F2L**
- Setup for the algorithm
  - The key to using this algorithm is knowing the appropriate "**setup**" prior to execution

EOLL DEMONSTRATION

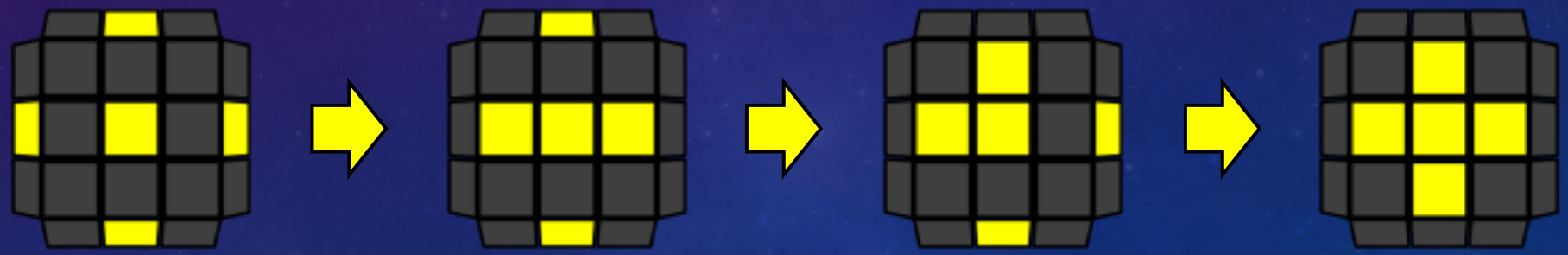
There are 4 possible cases during **EOLL** including the "**solved**" case

Approach: "**Dot case**" -> "**opposite edge flip**" -> "**adjacent edge flip**" -> "**solved**"

Setup: Ensure the "**adjacent edge flip**" is showing "**nine o'clock**" before executing the **EOLL** algorithm

Algorithm: **F (U R U' R') F'** – once, twice or thrice

e.g.



Dot **Opposite** **Adjacent** **Solved**

ORIENTING CORNERS OF THE LAST LAYER (OCLL)

There are 8 possible cases during **OCLL** including the "**solved**" case

2/27

4/27

4/27

4/27

4/27

4/27

4/27

1/27

ORIENTING CORNERS OF THE LAST LAYER (OCLL)

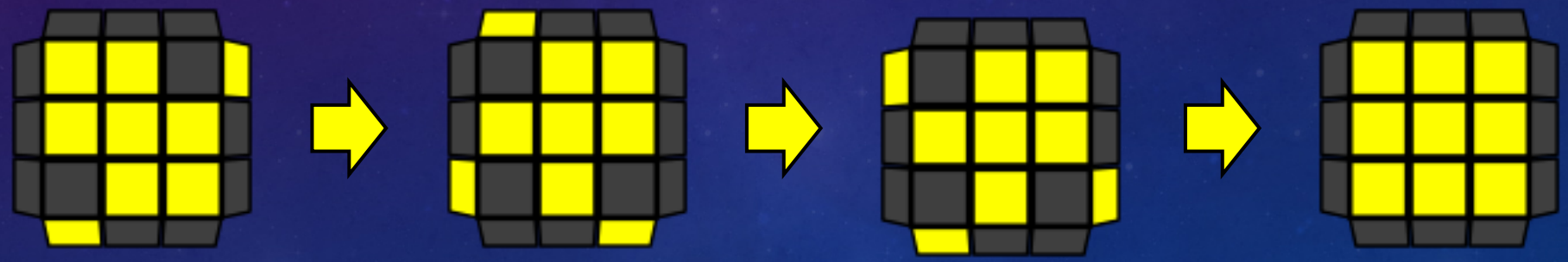
OCLL can be solved by cycling through the cases in a similar fashion to **EOLL**

Cases: There can only be 2, 3 or 4 "**twisted**" corners, corresponding to the images on the previous slide

Probabilities: Most of the **OCLL** cases have a probability of **4/27**

Approach: Cycle from **2** twisted corners or **4** twisted corners -> **3** twisted corners -> "**solved**"

e.g.



L / Bowtie **Sune** **Anti-Sune** **Solved**

OCLL ALGORITHM

- To affect the orientation of corners on the last layer you only need to use **R U** moves
- The **OCLL** algorithm that will be used is a combination of two triggers – **(R U2 R' U2') (U R U' R')**
  - The **R U2 R' U2'** trigger extracts the front-right **F2L** pair
  - The **U R U' R'** trigger re-inserts the front-right **F2L** pair
  - The algorithm will "**twist**" three LL corners but it also has a side effect of "**permuting**" LL edges
  - Note: The algorithm can be shortened to **R U2 R' U' R U' R'** due to a "**cancellation**" between the triggers
- Explanation of the algorithm
  - The algorithm extracts an **F2L pair** using the **R U2 R' U2'** trigger and re-inserts it using the **U R U' R'** trigger
  - It is worth noting that any algorithm consisting solely of **R U** moves will not affect edge orientation (**EO**)
- Setup for the algorithm
  - The key to using this algorithm is knowing the appropriate "**setup**" prior to execution
  - The setup is based on simple rules based on the number of "**twisted**" corners – see the following slides

3 TWISTED CORNERS

There are 2 **OCLL** cases with 3 twisted corners

Approach: “**Sune**” -> “**Anti-Sune**” -> “**Solved**”

Setup: Ensure the “**oriented**” corner is at the back-right (see arrows) before executing the **OCLL** algorithm

Algorithm: **(R U2 R’ U2’) (U R U’ R’)** – once or twice, remembering to **AUF** prior to execution

Sune **Anti-Sune** **Solved**

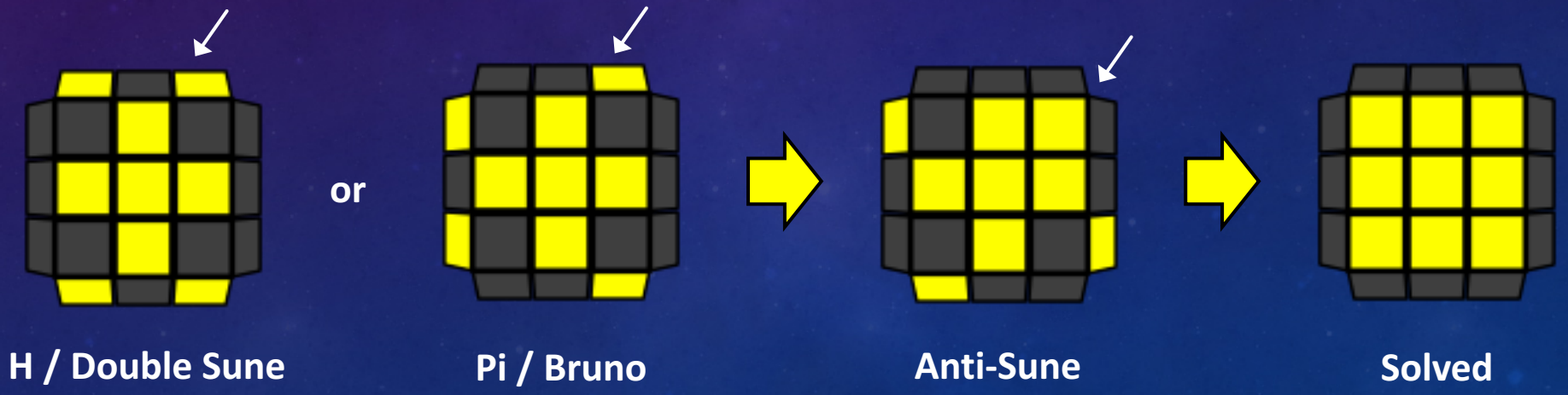
4 TWISTED CORNERS

There are 2 **OCLL** cases with 4 twisted corners

Approach: "**H / Double Sune**" or "**Pi / Bruno**" -> "**Anti-Sune**" -> "**Solved**"

Setup: Ensure the back-right corner has its U-sticker on the back before executing the **OCLL** algorithm

Algorithm: **(R U2 R' U2') (U R U' R')** – twice, remembering to **AUF** prior to execution



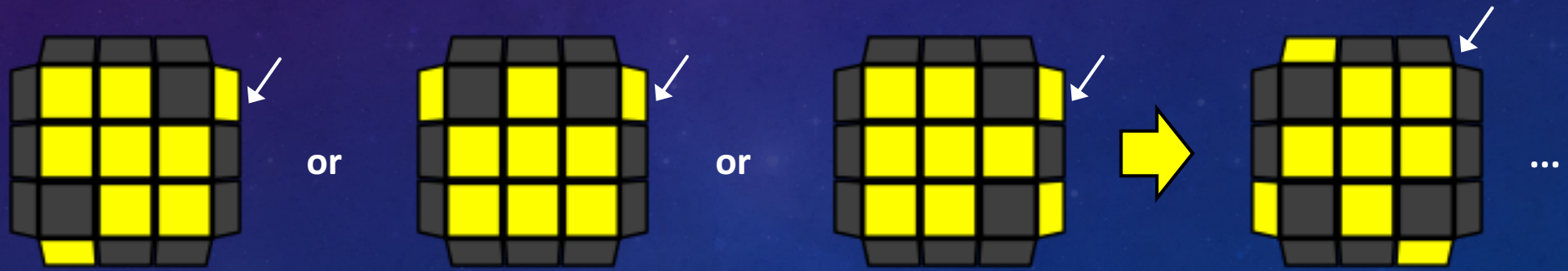
2 TWISTED CORNERS

There are 3 **OCLL** cases with 2 twisted corners and they take the most effort to solve

Approach: “**L / Bowtie**” or “**T / Chameleon**” or “**U / Headlights**” -> “**Sune**” -> “**Anti-Sune**” -> “**Solved**”

Setup: Ensure the back-right corner has its U-sticker on the right before executing the **OCLL** algorithm

Algorithm: **(R U2 R’ U2’) (U R U’ R’)** – thrice, remembering to **AUF** prior to execution



L / Bowtie **T / Chameleon** **U / Headlights** **Sune**

FINGER TRICKS

F (U R U' R') F'

1. Use your index finger(s) to turn the U-layer for the "**setup**" instead of rotating the cube around the y-axis
2. Re-grip so that your right thumb is underneath the R-face and fingers are on top before executing the algorithm
3. The **F** move should be executed with your right index finger, pushing the top-right corner downwards
4. Re-grip before executing the **U R U' R'** trigger which in itself should not require any further re-grips
5. The **F'** move should be executed with your right thumb, pushing the bottom-right corner upwards

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

1. Use your index finger(s) to turn the U-layer for the "**setup**"
2. Re-grip so that your right thumb is underneath the R-face and fingers are on top
3. Execute the whole algorithm without re-gripping
4. The **U2** can be executed as a "**double flick**" – i.e. index finger followed by middle finger

NEARLY DONE!