

ZZ Method 教程（翻译）（更新LL）

本文是翻译，原文地址在下面，本人英文并不好，但本着只翻一遍的原则，就把译文发了上来，为了以后自己看方便，也为了英文还不如我的魔友们，同时也希望英文好的同学能帮助我提高翻译水平，谢谢大家

<http://cube.crider.co.uk/>

有兴趣交流关于此方法的同学请加 59293291 群

从 F2L 开始有一些图例没有上，以后会有 WORD 版的。
（不过貌似想学这个方法的也不多）

这里提供了一些关于学习 ZZ method 的信息。ZZ 是一种新兴的速解方法，最早由 Zbigniew Zborowski 在 2006 提出。Michal Hordecki 的网页提供了很好的背景资料以及为那些不太熟悉这个方法的人提供了很好的介绍。

介绍这个算法要求一些标准的记号（U、D、R、L、F、B 等）

内容摘要：

A. EOLINE

判断棱块色相

解决棱块色相

棱块色相实例

使 E0 和 Line 同步进行

解决 EOLine 的实例

B. F2L

基本建块规则

1x1x2 块的创建

1x2x2 快的创建

特殊情况的算法

地角落位的情况
楞块落位的情况
相连型
不易处理的相连型
多重块

C. LL

OCLL/PLL
COLL/EPLL
ZBLL
F2LL

D. 实例

EOLine

这一步要把所有楞块色相调正 (EO) 同时把 DB 和 DF 楞块还原 (Line)。这一步的平均步数约为 6.127，最多需要 9 步，但是这无疑是 ZZ method 最难的一步。所以，通常这一步被分成 2 部分：调好棱块色相然后做好底层的线，也就是 EO+Line。

我们定义如果一个棱块可以通过只使用 RLUD 这几个面转动还原我们就管他叫色相正确，不能则为色相错误（错误的楞块）。一旦 DF 和 DB 的棱块还原了，就意味着整个 F2L 只需要通过转动 L、R、U 这三个面就可以完全还原。

棱块色相的判断

当你在 UD 层发现以下情况时，该棱块为错误块：

1. RL 的颜色出现在 UD 面
2. UD 的颜色出现在非 UD 面上

当你在中层发现以下情况时，该棱块为错误块：

1. RL 的颜色出现在 FB 面
2. UD 的颜色出现在 LR 面

解决棱块色相

旋转 90° 的 F 面或 B 面会导致 F 面或 B 面上所在的棱块色相改变。用 F/B 旋转 90° 来改变 F 面或 B 面上错误色相的棱块是我们解决棱块色相问题的核心技术。注意：F2 和 B2 不会改变

棱块的色相。下面是一些处理棱块色相的基本策略，根据错误色相棱块的数量分类。

0: 已完成，所有棱块色相正确

几率: $1/2048 \sim 0.05\%$

2: 先移动一下带有一个错误棱块的 F 面或 B 面，然后用另一个错误的棱块，替换刚刚改变为色相正确的那个棱块，然后再移动一下 F 面或 B 面

几率: $66/2048 \sim 3.22\%$

4: 把所有四个错误棱块移动到 F 面或 B 面然后转一下 F 面或 B 面

几率: $495/2048 \sim 24.17\%$

6: 将有三个错误棱块的 F 面或 B 面移动一下，就剩下 4 个错误棱块了。或者是先处理 4 个在处理 2 个相对于 3+3 的策略，上面的策略通常只需要移动 3 次 F 面或 B 面。

几率: $924/2048 \sim 45.12\%$

8: 看做是 4+4

几率: $495/2048 \sim 24.17\%$

10: 看做是 4+4+2

当你处理这种情况的时候，记住要跟住 2 个正确的块，而不是 10 个错误的块。第一步把两个正确的块移出 F 面和 B 面，然后同时移动 F 面和 B 面，最后再处理剩下的两个错误块。

几率: $66/2048 \sim 3.22\%$

12: 看作是 4+4+4

这是一种非常罕见的情况，但是如果你非常幸运的遇到他，立刻转动 F 面和 B 面，然后就还剩下 4 个错误块了

几率: $1/2048 \sim 0.05\%$

下面的规则将指导你决定怎么处理错误的棱块:

1. 通常选择本来就拥有更多错误块的 F 面或 B 面来改变色相是更有效率的
2. 通常当你把一个错误块移动到 F 面或 B 面的同时把另一个本来就在 F 面或 B 面的错误块移出是低效率的
3. 当遇到 6 个或多于 6 个以上错误块的情况，同时移动 F 面和 B 面通常比只移动 F 面或 B 面更有效率
4. 当遇到 6 个或多于 6 个以上错误块的情况，你把一个错误块移动到 F 面或 B 面的同时把另

一个错误块移动到B面或F面是更有效率的

5 在移动F面或B面是只改变2个相对的错误棱块是没有意义的，但是再做E0-Line的时候却会有用。

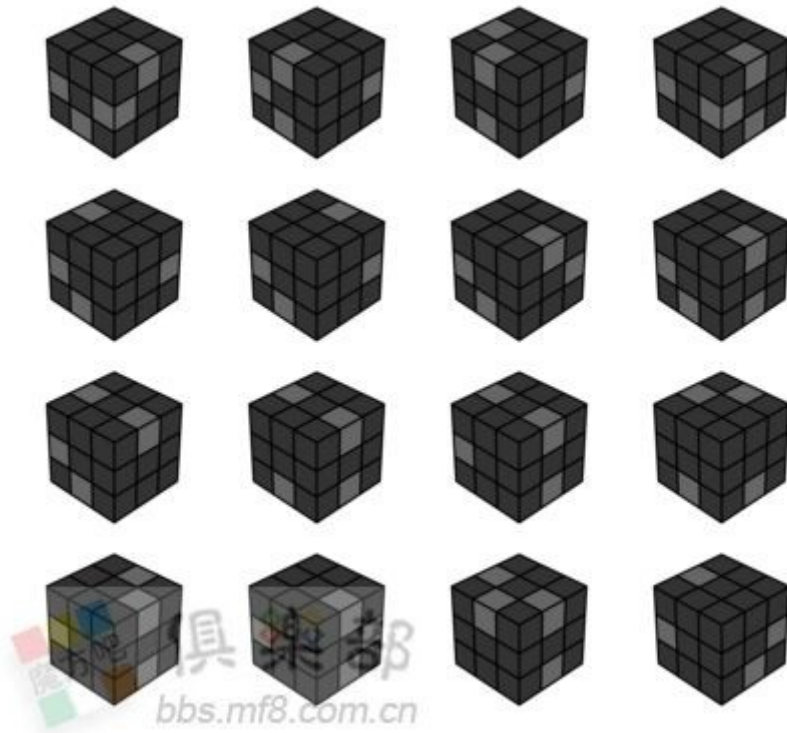
棱块色相实例

因为有超过2000种不同的E0情况，所以把所有的情况的解决办法都记下来是不切实际的，

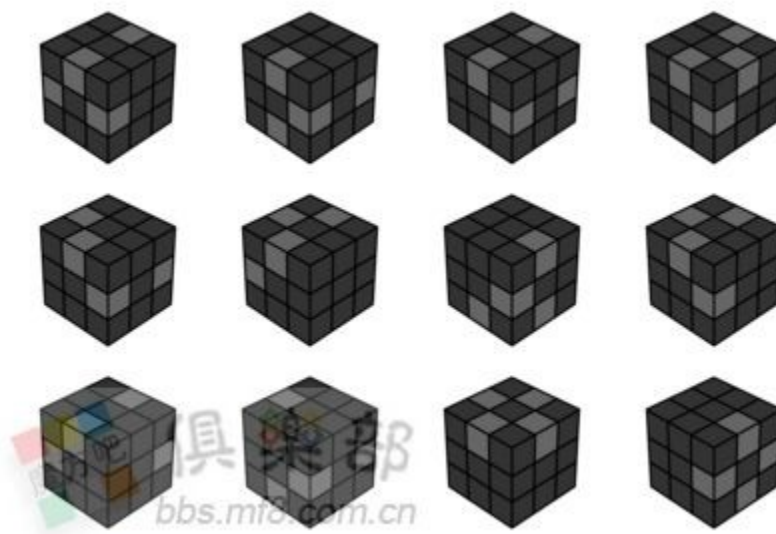
所以我们记住解决的思路已处理相应的情况。

E0中大多数解法是个一组进行的。所以我们只演示4个错误棱的情况，通过一定方法使他们移动到F面。这些方法也可以很容易的用在B面上，很多情况又不只一种的最优解法，在一定程度上可以自由的把一个棱移动到对面，或者创建一个Line

只有一个需要设置的



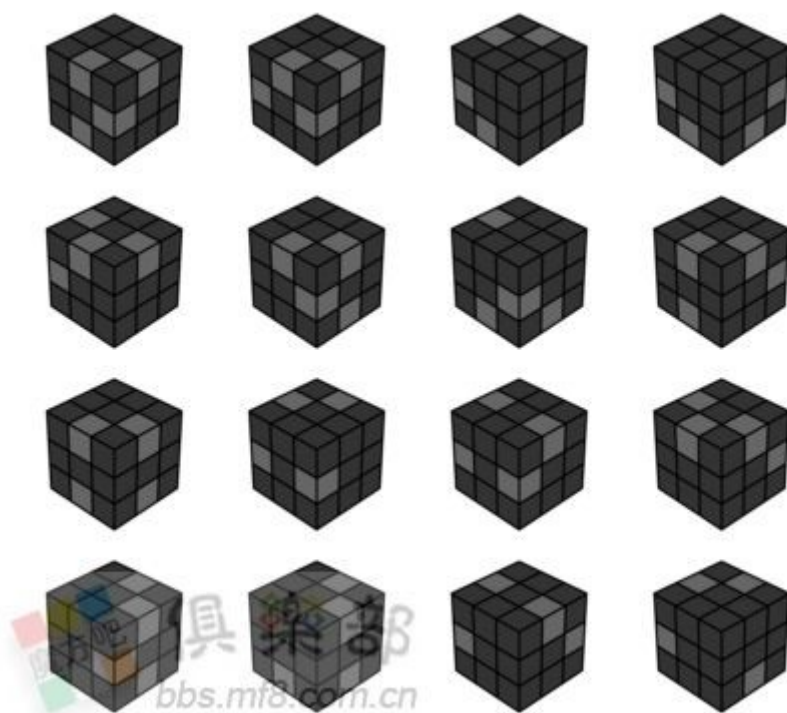
需要用到 F2B2 的



需要同时移动 3 个棱的



需要同时移动 2 个棱的



需要用到 F2 才能构成同时移动 2 棱的



需要移出一个棱的



即需要移出一个棱又需要用到 F2 的



同时做出 E0 和 Line

当调整棱块色相变的很熟悉之后，就可以开始计划如何做出 Line，并且把这个过程整合到棱块的色相调整中，达到棱块调色相和做 Line 同时进行。一开始可以先把 Line 整合到一些比较简单的 E0 情况中（比如，当只有 2 个或者 4 个错误块的时候）。随着水平的增长，逐渐尝试在更复杂的情况下做 Line。

最简单的把 line 结合到 E0 的方法是，找到要做 Line 的棱块，然后计算出在 E0 之后他们的位置，然后在这个最终的位置的基础上不用看就可以继续做出 Line。尽管这个方法无疑要比 E0+Line 快一些，但是在步数上却一摸一样。有可能通过选择正确的方法调整棱块色相使得 Line 的布置更有效，有时甚至可以完全消除做 line 这个步骤。下面将介绍怎么做到这些

我们想要在 D 面上创建一条 Line。不管这条 Line 是怎么创建的，最终他可以归结为下面的情况：

目标一：Line 在 DF/DB 的位置上

目标二：Line 在 DR/DL 的位置上，接着通过 D/D' 的旋转，只需要多移动一步

把这两个目标记在心里，我们就可以根据他们到任意一个目标的步数来衡量所有可能的方法。

到目标 1 的距离（第一组）：

0：DF、DR（色相已对时）

1：UF、UB、*DR*、*DL*（色相已对时）

1：RF、LF、RB、LB（色相不对时）

到目标 2 的距离（第二组）：

0：DL、DR（色相已对时）

1：FL、BL、FR、BR、UL、UR、*DF*、*DB*（色相已对时）

1：FU、FD、BU、BD（色相不对时）

把两个组成 Line 的棱块安置到目标一或目标二，会产生一个更有效的结果，除非其中一个的归位会导致另一个离开应在的位置。利用斜体字的位置避免这种情况发生。

如果一个棱块形成了目标一的状态，而另一个形成了目标二的状态，总会产生一个非常麻烦的状态（最少用 2 步，通常需要 3 步以上）

一个高效的解决方案可以由在处理色相的过程中想法把 Line 的两个棱块落到第一组而实现。

如果这需要很多步才能完成，那么尝试把他们落到第二组中去，但是一定要注意避开把他们一个放在第一组，另一个放在第二组。

解决 EOLine 的实例：

1.打乱步骤：F U2 D2 B2 U' F U' R' L' D R'

观察：这个 case 只有 4 个错误块，分别在 UF、UL、DL、DB。基本的策略是把这四个移动到 F 或 B 面上然后旋转一下。DF 棱现在在 DR 位上，而且色相正确。DB 棱在 RU 位上色相错误

EOLine: D' L' R' U2 B

分析：一开始 D' 让色向正确 DF 归位。L'R'U2 使所有四个棱块到 B 面上，并且使色相错误的 DB 棱到达第一组的位置。最后的 B 使所有色相调正，并且使 Line 也做好了

2.打乱步骤：F' B' U L2 D B R2 L2 U'

观察：共有 8 个错误棱块。分两组解决他们。已经有四个错误块在 F 面上了。DB 和 DF 都在 B 面上第一组的位置

EOLine: F D B U2 F2

分析：一开始的 F 直接修正了四个错误块，只留下了 4 个错误块。D 让唯一一个不在 B 面的错误块到 B 面。旋转 B 面调正 4 个错误块并使 DF 块和 DB 块到达第一组的位置，这样旋转 U2 和 F2 使得 Line 完成

3.打乱步骤：R F2 B' D' L U2 L2 D2 B

观察：共有 6 个错误块，看起来用 3+3 的程序处理这个问题会很有效。DF 和 DB 都是错误块 DF 在第一组的位置，DB 在靠近第一组的位置。

EOLine: B' U2 D' L' B F2

分析：一开始的 B' 调正了 3 个错误块，剩下 4 个错误块。接下来的 U2 完成了 2 件事情：把一个错误块就位了并且把 DF 块放到了第一组的位置。D' 和 L' 把剩下的两个块放到了 B 面，B 改变了剩下四个的色相，F2 是 DF 归位。

4 打乱步骤：R' D F2 R B' U F' D' F2

观察：共有 8 个错误块，所以基本的策略是分两次每次解决 4 个快。F 面有 3 个错误块，比 B 面得 2 个要多。DB 棱在正确的位置上而色相也正确，同时也是第一组的位置。DF 棱也在正确的位置上但是色相错误了，属于第二组。

EOLine: U F L B' D U B

分析：一开始的U使F面全是错误块然后处理了他们。现在DF棱到了第二组的位置，同时DB棱到了第一组的位置。接下来的L使得DF到了第一组的位置，但是现在这两个块在D面互相垂直。所以一个B'使DB块让出去，这样就可以让DF块可以归位了。与此同时错误块的数量和位置也不受影响。接下来的D和U把剩下的两个错误块放到了B面，而且DF也归位了。最后的B修正了所有剩下的4个错误块，并且让DB棱归位。

F2L 完成前两层

基本建块策略

这一步通过构建两个在Line左右的1x2x3的块来完成前两层。我们按顺序把这个步骤分成更小的步骤，这样就更容易被识别并快速执行。我把这一部分为四个部分：

1. 左手的1x2x2
2. 右手的1x2x2
3. 左手的1x1x2
4. 右手的1x1x2

执行的顺序依照不同的情况而定，通常来说是先完成1x2x2然后在完成1x1x2，否则的话会是整体的步骤变多。

1x1x2 块

构建1x1x2块通常是在U面完成，使用的方法和CFOP的方法很相似。但是不能旋转魔方，只能用L、R、U来完成，三个基本型是：

如果要连接的角块和棱块没有都在U面上那你就需要先把他们移动到U面上，再做公式。如果他们已经在U面上，但是错误的链接到了一起，那就需要先把他们拆开，再做公式。

1x2x2 块

构建一个1x2x2你需要一个角块2个棱块（一个D面得棱块和一个中间层的楞块）。构建这个块需要由一个角块和一个棱块组成的1x1x2块，以及一个棱块。把单独的楞块放到正确的中间层（FL/BL/FR/BR）然后通过移动U把它们连接到一起。

特殊情况的算法

建筑块最核心的技术是直觉的用最少的步骤解决前两层。和 F2L 相似的是提高塔可以更有效的解决比较复杂的情况。例如需要解决的几个块被错误的链接上的时候。建议完全理解之后再背熟 F2L 算法。

解决这些情况最理想的方法（除非指定其他方法），并且不影响棱块的色相。适用于所有的对应情况

底角落位的情况

棱块落位的情况

相连型

所有的相连型被分成三组

1. 相连的块在 U 层，在他们的槽上面
2. 和 1 一样 但是对面的槽是空的
3. 相连的块已经在他们的槽里面了

以下公式最初的步骤有的是要把快移到做公式的位置所以有可能是没有必要的

错误连接的块

多重解

有时候会遇到多个 $1 \times 1 \times 2$ 出现在 U 层上。如果 LR 两边都是空的，那么很容易的先将一个临时放在一个槽中，先处理剩下的，然后再处理刚才存储的。但是如果有一边已经完成了，那么就只有很少的余地让你不破坏已经完成的块而完成剩下的步骤。下面介绍了一些已经优化的这类算法。如果一组已完成的块已经 L 或者 R 面，那么旋转这个面，让他们到 U 面上，就会和下面的某个情况相吻合。

两个平行的块

两个相邻的块

F2L 和 D 层相连的：D 层可以动的

F2L 和 D 层相连的：F2L 可以动的

F2L 和 D 层斜对着的：剩下的棱块在 U 面

F2L 和 D 层斜对着的：剩下的棱块在 R 面

如果你非常偏好使用一只手，那么旋转魔方会让算法更容易执行。最后请注意，可以用 ZBF2L 来完成最后一组 $1 \times 1 \times 2$ 。只需要 ZBF2L 中 LL 棱块已经调好色相的那些算法

C.

LL – The Last Layer

在 ZZ 的最后一层的解法中有很多不同的方法可以选择。一下几种值得注意：

OCLL/PLL：想调整最后一层角的方向（OCLL），然后在同时调整最后一层棱块和角块的位置（PLL）。这和 CFOP 的 OLL/PLL 很相似，但是由于顶层棱块已经调成好色相了所以 OLL 这一步需要的公式更少，OCLL 最少只需要 6 个公式，PLL 最少只需要 14 个公式，加起来一共 20 个公式。OCLL 平均为 7.93 步，PLL 平均为 11.21 步，一共是 19.14 步。通过在嵌入最后一个 1x1x2 的时候不完全控制角块，来消除 H 和 Pi 的 OCLL，可以让 OCLL 和 PLL 的步数和公式更少

- [speedsolving.com wiki: OCLL](https://speedsolving.com/wiki/OCLL)
- [speedsolving.com wiki: PLL](https://speedsolving.com/wiki/PLL)
- [Dan Harris's OLL Page \(corners only\)](#)
- [speedsolving.com: Partial Corner Control](https://speedsolving.com/Partial-Corner-Control)

COLL/EPLL (ZZ-VH)：先一步调整最后一层角块的色相和位置（同时不改变棱块的色相），然后一步调整最后一层棱块的位置。相比起 OCLL/PLL 他的步数更少，而且有很多更容易被识别的情况。学习 COLL/EPLL 同时也是学习 ZZLL 或 ZBLL 很好的中间步骤。COLL 有 40 种不同的情况，最少需要 25 个公式。EPLL 只有 4 种情况（只有 3 个公式），最后一层一共 28 个公式。COLL 平均 9.78 步，EPLL 平均 8.75 步，加起来一共 18.53 步，比 OCLL/PLL 稍微少一点

- [Rob Burton's COLL Page](#)
- [Jason Baum's COLL Page](#)
- [Dan Harris's COLL Page](#)
- [Lars Vandenberg's COLL Page](#)
- [speedcubing.com: COLL Page](#)

ZBLL (ZZ-a): 就像魔方界的圣杯一样, 这个算法可以一步解决最后一层的角块方向和位置以及楞块的位置。它包括了 494 种不同的情况, 并且要求至少 177 个公式和它们的镜像以及逆公式。完成最后一层只需要平均 12.08 步, 在 2-look 的情况中这个很让人满意了

- [Bernard Helmstetter's ZBLL algs](#)
- [Chris Hardwick's ZBLL Page](#)
- [Jason Baum's ZBLL Page](#)
- [Lars Petrus's ZBLL Page](#) (warning: many java applets!)

F2LL

在处理最后一个 F2L 的时候, 是有可能通过一些算法减少最后一层的可能性。这种在 F2L 的时候部分处理最后一层的方法叫 F2LL。下面列举了一下 F2LL 的选择:

Phasing (ZZ-b or ZZLL): 在完成最后一个 1x1x2 的时候交换相对的 2 个 LL 棱块。相比较而言这是一种初级技术, 大约 2 步左右塔使得最后一步的数量从 494 种降低到 167, 让最后一步只需要 80 个公式。同时, 这也是学习 COLL/EPLL 和全部 ZBLL 过程中很好的过渡

- [Michal Hordecki's ZZLL algorithms](#)
- [Michal Hordecki's Phasing algorithms](#)
- [James Stuber's ZZLL algorithms](#)
- [speedsolving.com: Phasing Explained](#)

Winter Variation (WV):在完成最后一个 1x1x2 的时候调整最后一层角块的色相，让接下来的最后一层只剩下 PLL。只需要 27 个公式并且比 OCLL/PLL 的平均步数更少。

- [Sebastien Felix's WV Page](#)
- [Ostracod's WV Page](#)
- [Dragon Cube WV Page](#)

MGLS: 在最后一个 F2L 的时候只有棱块是完成的。MGLS 中 CLS 部分然后调整最后一层的角块色相同时让 D 面得那个角块完成。这样就让最后一层只剩下了 PLL。这一步总共有 105 种情况，最少需要 56 个公式，平均 9.5 步。如果将 D 面的角块落位了也能减少总情况数（色相没有必要也正确）。这种方法叫 EJF2L，总共需要 16 个公式

- [Lucas Garron's MGLS Page](#)
- [Cubefreak MGLS Section](#)
- [YouTube: EJF2L Tutorial](#)
- [YouTube: EJF2L Cases](#)

Blah's Method: 在嵌入最后一个 $1 \times 1 \times 2F2L$ 的时候调整 LL 的角块。使得最后一步的 ZBLL 数量变少