基于专家系统与滑动窗口的空战动作识别、意图分析与战场态势评估

队员：茹靖涵　陈　曦　田亚里笛

摘　要　本文基于机动动作知识库，使用滑动窗口法实现了一种识别飞机机动动作，并在视距内预测机动动作意图，在此基础上建立了使用优势函数量化战场双方的优劣势的直接算法. 该方法在即使没有大量训练数据来进行学习时依然可以保持较好的准确度.

首先可以把机动动作根据复杂度和基本性分成复杂动作和基本动作，所有复杂动作都可以由 5 类基本动作生成，由于基本动作的特征明显，参数明确，便于研究，因此可以采用两步走的方法，首先通过查阅文献可以得到相当准确的飞行基本动作的对应参数，据此可以建立起基本动作的参数知识库，为获得某段时间内飞机的动作数据，可以设置滑动窗口对时间序列进行分段线性拟合，并使用显著性检验，若通过检验，则认为拟合效果较好，区间内各个动作较为相似，应该划分为一段子序列，若检验不通过，则区间内有参数非线性变化，需要减少窗口长度再次尝试拟合，这样既最大程度上在综合考虑了该段时间的所有信息的同时增加区间划分的准确性，也可以容易地通过获得的线性函数的斜率得到对应参数的变化率，以此为判断标准识别机动动作.

为预测一段动作的意图，需要先对动作序列做分割，为此需要评估相邻动作之间的相似性，通过判断是否存在姿态参数的突然剧烈变化来切割区间，由于复杂动作的动作意图相对更加清晰，尝试将一段子序列中的基本动作组合成复杂动作，结合战场整体态势与当前战机的处境来预测该子动作序列的意图，为此分别从角度，高度，距离，速度，事件等方面建立优势函数并综合考虑一架飞机在当前局势下的总优势函数，以此来作为意图判断的一个指标. 另一方面，该战机当前的动作也可以反映其行为意图. 综合以上两个方面就可以建立意图判断算法.

为建立起战场整体态势函数，先对每架战机建立优势函数，之后对双方各自阵营的所有战机的优势度求和即可得到双方对应的优势度，之后做差即可得到战场整体态势函数，用该函数的符号来反映当前局势对哪一方更有利.

关键词　机动动作知识库，滑动窗口法，动作拆分，优势函数，意图分析

一、问题重述

受到俄乌战争的影响，空战游戏逐渐成为了当下的热门. 在游戏开发过程中，如何针对不同玩家、不同场景生成对应的人机对抗策略，已成为一项具有挑战性的研究内容.

游戏中，每个玩家都能够独立驾驶一架“战斗机”，并且在飞行过程中操作战斗机完成各种各样的飞行动作. 随着当今战斗机性能的不断提升，游戏中战斗机在性能指标上能够完成的飞行动作也愈加多样，如果玩家可以充分发挥战斗机的性能优势，就可以实现各种各样的作战姿势和作战策略，提高任务完成度. 除此以外，由于不同玩家的操作水平高低存在差异，所以最终呈现出的战场任务完成率也各不相同.

在这一过程中，通过对战斗机的机动动作进行识别与分析，有助于“团体”作战指挥人员准确评估每场战斗中“空中格斗”的效果、掌握每一名飞行员玩家的飞行水平及动作完成情况；同时还能帮助玩家在游戏结束之后，从飞行动作数据中找出机动动作操作过程中的不足并及时改善，缩短提升战斗力的训练周期. 因此，实现游戏中战斗机机动动作的识别，是对其进行数据分析的重中之重.

现有大量的飞行对抗模型数据；而玩家经常通过 TacView 软件进行战斗过程的回放，从中找到成功或失败的原因.

基于上述内容, 玩家 A 聘请我们作为游戏顾问，帮助他解决以下问题：

1. 通过文献调研，研究战斗机的机动动作，建立“机动动作”的描述和量化模型. 该模型应尽可能覆盖常见的战术机动动作.

2. 通过附件 1 中的 10 场飞机时序数据，结合问题 1 的模型，得到各架飞机的机动动作序列，并按附件 2 中的给定格式输出.

3. 建立机动动作序列的切割算法或者是若干机动动作的组合算法，判定各个机动动作子序列的意图，并使用 10 场数据进行验证.

4. 建立空战战场态势评估模型，并使用 10 场数据进行分析验证.

二、基本假设

（1）所有飞机的任意机动动作都可以由 5 种基本机动动作生成，基本机动动作包括平飞，上升，下滑，转弯，翻滚.

（2）在同一时间一架飞机有且仅有一种意图.

（3）所有战机都根据自身处境以及战场整体状况决定下一步行动.

（4）双方所有战机在任意时刻都可以获得对方所有战机的位置和信息.

（5）所有战斗机自子序列的意图有且仅有攻击、防御、侦察三种类型.

三、符号说明

各符号的含义说明见表 1.

表 1: 符号含义说明

符号	含义	符号	含义
l_0	初始窗口长度	l_k	第 k 次运算时的窗口长度 $(k = 1, 2, 3, \ldots)$
M	窗口长度上限	L_k	第 k 段线性拟合函数 $(k = 1, 2, 3, \ldots)$
ϕ	维度	R^2	线性相关系数
θ	经度	b_k	第 k 段线性拟合函数的斜率
Yaw	偏航角	φ	方位角
$Roll$	横滚角	q	进入角
$Pitch$	俯仰角	v	真实空速
d	相距距离	d_0	视距距离
E	飞机动能	h	海拔高度
R	地球半径	R_q	进入角优势函数
R_φ	方位角优势函数	R_a	角度优势函数
R_d	距离优势函数	R_e	能量优势函数
R_f	时间优势函数	R_v	速度优势函数
R_h	高度优势函数	BS	本机优势函数

四、问题一

（一）问题分析与准备

该问题要求我们为战斗机的 “机动动作” 建立相应的描述和量化模型. 而解决问题的关键，就在于确定模型中有哪些 “机动动作”, 以及如何利用每秒种记录下来的飞行参数将飞行动作识别出来. 基于以上要求，我们做了以下前期准备工作：

第一，对战斗机的 “机动动作” 做出明确定义. 通过查阅大量的相关文献，最终总结归纳出表 2 中的 5 个基本飞行动作和表 3 中的 7 个复杂飞行动作 [**?**]：

表 2: 基本动作说明表

机动种类	定义
平飞	除经纬度均匀变化外，其他参量不变，偏航角不变.
上升	经纬度或偏航角基本不变，但俯仰角突然增加，高度快速上升.
下滑	经纬度或偏航角基本不变，但俯仰角突然降低，高度快速下降.
转弯	水平速度方向或偏航角改变.
翻滚	横滚角变化且非零.

表 3: 复杂动作说明表

机动种类	组成方法	定义
定常盘旋	转弯	偏航角变化速度恒定且变化 ≈360°，高度基本不变，横滚角保持非零.
急盘降	转弯 + 下滑	偏航角变化速度恒定且变化 ≈360°，高度迅速下降.
俯冲	下滑	偏航角基本不变，高度迅速下降.
急拉起	上升	偏航角基本不变，高度迅速上升.
半斤斗	上升/下滑	高度略有变化，俯仰角变化 ≈180°，横滚角基本不变.
半滚倒转	上升/下滑	高度略有变化，俯仰角变化 ≈180°，横滚角变化 ≈180°.
桶滚	翻滚	高度和偏航角基本不变，横滚角变化 ≈180°.
战术转弯	转弯	偏航角快速改变，改变量 <90°.
规避急转弯	转弯	偏航角快速改变，改变量 >90°.
高速 Yo-Yo	上升 + 翻滚	首先高度上升，之后横滚角变化 ≈180°.
低速 Yo-Yo	下降 + 翻滚	首先高度下降，之后横滚角变化 ≈180°.

现规定战斗机的每个 “机动动作” 可能为上述十二个动作中的一种，而所有复杂动作都可以通过上述的五个基本动作在时间空间上的相互组合形成. 在这种处理方式下，复杂的 “机动动作” 拆分为了多个基本动作的组合，由于基本动作耗时更短、操作更单一、相关联的飞行参数更少，所以识别的精度与准确度更高，效果更好. 而实现基本动作识别算法将在建模过程中进行具体论述.

第二，定义 “窗口” 概念. 由于战斗机完成任何动作都需要一定的时间长度，所以说战斗

机“机动动作”的识别必须基于某一时间段，我们就将该时间段成为为一个“窗口”. 一般来说，该窗口取值越小，得到的片段越多，误差越小，但是过短却会不足以做出飞机整体动作的判断；窗口取值越大，得到的片段越少，稍大的窗口有利于判断飞机动作的完整性，但是过大时则会影响到飞机单个动作的判断，误差也大. 最终窗口长度根据片段多少和误差情况进行综合取值.

第三，采取合适的方法对基本动作进行识别. 对于本问题，数据集的精度较高，基于这个想法，在已知飞机的飞行参数的情况下要分析动作序列可以有下面几个思路：

（1）用聚类算法训练参数：需要训练集.

（2）对插值法得到的函数求导判断变化：数值导数.

（3）滑动窗口法分段用线性函数拟合看斜率的情况：窗口长度选择 + 线性回归显著性检验.

（4）知识库法：利用文献中的参数判断某一动作 + 根据数据计算对应参数.

由于本题中的原数据仅仅给出了飞机在某时刻的飞行参数，但没有给出它此时对应的机动动作，所以无法建立训练集，故排除思路（1）；另一方面，若使用插值法，插值节点太多，耗时太长，故排除思路（2）. 综上所述，最终采用滑动窗口法以及知识库法结合作为最终的建模思路.

完成以上准备工作后，就可以顺利进入到建模环节.

（二）模型的建立

基于上述分析，首先需要确定基本动作对应的的相关参数与参数范围. 在查阅文献资料后，确定了表 3 中所示的各个基本动作飞行参数特征以及范围. 其中，“——”代表该类动作与此处参数的取值无关.

表 4: 基本动作参数特征表

基本动作	航向变化率 $(^\circ/s)$	俯仰角变化率 $(^\circ/s)$	倾斜角变化率 $(^\circ/s)$	高度变化率 $(^\circ/s)$
平飞	$(-1,1)$	——	——	$(-2,2)$
上升	——	——	——	>2
下滑	——	——	——	<-2
转弯	>1	——	——	——
翻滚	——	——	>10	——

由上表可知，所有基础动作的识别，都可以通过一段时间内参数特征的线性拟合斜率进

行识别与筛选. 例如判断上升类动作，只需要将一段时间段内所有高度数据的离散点进行线性拟合得到斜率，然后判断该斜率的值 > 2 即可.

现依据上述思路进行建模，则对时间窗口长度的选择需要重点考量. 首先是选择合适的初始时间窗口长度，以保证离散点进行良好线性拟合，得到准确的拟合斜率；其次是需要进一步调整窗口的长度，以确保每一个窗口内求得的线性拟合斜率，都可以匹配到表 3 中的基本动作. 如过不能匹配成功，则继续对窗口长度进行调整，知道能够识别出动作为止.

对于窗口长度的选择，采用滑动窗口法对飞行时序数据进行切割.

首先，将初始时间窗口长度设置为 l_0. 需要注意的是，在设置初始窗口时，为防止窗口长度过长而参数变化较小，从而造成某时刻的飞行动作判断错误，应该提前设置窗口的长度阈值

$$l_k \leq M, \forall k \tag{1}$$

在参考相关文献后，最终取 $l_0 = M = 10$ 作为窗口初始长度.

在选定初始窗口长度后，取出当前飞行时间序列数据集 $\{X_k\}_k^{N_k}$ 的前 l_k 位数据进行最小二乘拟合，得到第 k 段线性拟合函数

$$L_k : \hat{y} = a_k + b_k x \tag{2}$$

为了保证 L_k 的拟合度较高，需要对前一次（即第 k 次）获得的 L_k 函数做 R^2 检验. 对于一元线性回归方程 $\hat{y} = a + bx$ 而言，可求出以下平方和公式

回归平方和

$$SSR = ||\hat{\boldsymbol{y}} - \bar{y} \cdot \boldsymbol{I}||^2 \tag{3}$$

残差平方和

$$SSE = ||\boldsymbol{y} - \hat{\boldsymbol{y}}||^2 \tag{4}$$

总离差平方和

$$SST = SSR + SSE \tag{5}$$

结合以上公式，可以推则 R^2 统计量为

$$R^2 = \frac{SSR}{SST} \tag{6}$$

由以上公式可得，R^2 的区间位于 $[0, 1]$ 之内，并且 R^2 的值越接近 1，线性函数的拟合优度越好. 现将 R^2 的取值临界值设置为 0.95，当 $R^2 \geq 0.95$ 时，可认为拟合优度良好，检测通过；否则检测失败. 具体的处理方式如下：

（1）若检测通过，保留当前的 L_k，并从当前数据集中删去该次拟合使用的数据.

（2）若检测失败，说明当前窗口内的数据的拟合优度差，需要丢弃此时的 L_k，并缩小窗口长度 $l_k = l_k - 1$ 来提高拟合度. 然后重新求得新的 L_k 并进行 R^2 检验，直到检测通过.

通过以上操作，就能够得到此时的窗口长度 l_k，以及各个主要参数对应的线性拟合函数 $L_k : \hat{y} = a_k + b_k x$. 此时，再将斜率 b_k 与表 4 中的基本动作参数特征范围进行比较，具体的处理如下：

（1）若能够在表 4 中找到匹配的基本动作，则成功识别出该窗口内的战斗机 “机动动作” 类型.

（2）如果得到的结果无法与表 3 中的任意一种情况匹配，在不考虑误差的情况下，则说明该窗口内包含了多个基本动作，导致得到的参数特征不明确. 此时的处理方法同样是缩小窗口长度 $l_k = l_k - 1$，以分离出可以精确识别的单个基本动作. 然后继续重复上述的比较操作，直到在表 4 中找到对应的飞行动作.

循环以上步骤，将所有的基本飞行动作进行识别. 最后再将基本动作按照表 3 中的组合规则，把部分基本动作合成为一个大的复杂动作即可.

（三）模型的合理性与实用性说明

（1）合理性：由于本题所给数据未提供标签，因此很难检测两个序列所包含机动动作的相似度，自然不能使用聚类算法对序列进行分析. 对于数值导数算法，由于具有的样本点已经足够多，再使用插值增加伪数据点，会很大程度增加计算的时间复杂度，而对最终结果的影响较小. 因此，在查阅文献后，该模型选择滑动窗口算法利用参数变化率确定基本动作，效果较好. 通过所得到的结果文件与同一时间段内真实情况进行对比，发现大多数情况下飞机的动作确实可以与预测动作相符合.

（2）实用性：本模型所选用的参数来源于文献，经实验验证效果较好，证明参数的鲁棒性较高，可以在多种场合下对多种飞机进行良好的预测，可以迁移到多种场景中使用.

五、问题二

该问题要求使用问题一建立好的机动动作识别模型，将 10 场战斗的飞行时序数据导出为各架飞机的机动动作序列. 在对所给的 10 场战斗飞行时序数据表进行分析后，可以总结得出以下特点：

（1）每场战斗中共有两个战队，每个战队都有若干架战斗机参与战斗，每架战斗机都有唯一的编号进行标记.

（2）每架战斗机可记录的参数及指标多达 56 种，且此题提供的数据精度都较高.

（3）每间隔 1s，就对战场上的每架战机的飞行状态参数进行一次记录；在 10 场战斗中，战斗的持续时间从 10min 至 50min 不等.

现基于上述特点，并结合输出格式要求进行综合考量，做出以下的数据处理以及规定：

（1）通过飞机编号的识别，将每架战斗机进行区分并且单独识别. 在结束时，能够得到每架战斗机唯一的机动动作序列.

（2）虽然战斗机记录下的状态参数种类繁多，但实际上只需要 “航向变化率”、“俯仰角变化率”、“倾斜角变化率” 以及 “高度变化率” 这四个参数就可以对战斗机的机动动作进行识别. 结合附录 1 中所给的表格格式，需要使用到的四个指标就是：“偏航角 (Yaw)”、“俯仰角 (Pitch)”、“倾斜角 (Roll)” 以及 “海拔高度 (Altitude)”.

（3）为了简化最终的输出结果，我们使用不同的数字代表特定的飞行动作，其对应关系如表 5 所示.

表 5: 飞行动作——数字对应表

飞行动作	对应数字	飞行动作	对应数字
坠毁	-1	俯冲	8
未起飞（待命）	0	急拉起	9
平飞	1	半斤斗	10
上升	2	半滚倒转	11
下滑	3	滚筒	12
转弯	4	战术转弯	13
翻滚	5	规避急转弯	14
定常盘旋	6	高速 Yo-Yo	15
急盘旋	7	低速 Yo-Yo	16

在经过以上处理以及格式规范后，用 Python 进行程序编写，将附录 1 中的 10 场战斗数据以此代入问题 1 中得到的模型中进行计算. 最后，我们得到了 10 张符合附录 2 格式的表格，其中包含了每架战斗机每秒的飞行动作序列. 所有的表格全部存放在了命名为《问题 2 结果》的文件夹内.

接下通过实例，对本文对几类典型的飞行动作进行验证. 首先以第一组样本中的 102 号飞机为例，通过图 1 可看出它远离战场且飞行动作为先爬升再直飞后盘旋，图片中的颜色代表飞机的横滚角大小.

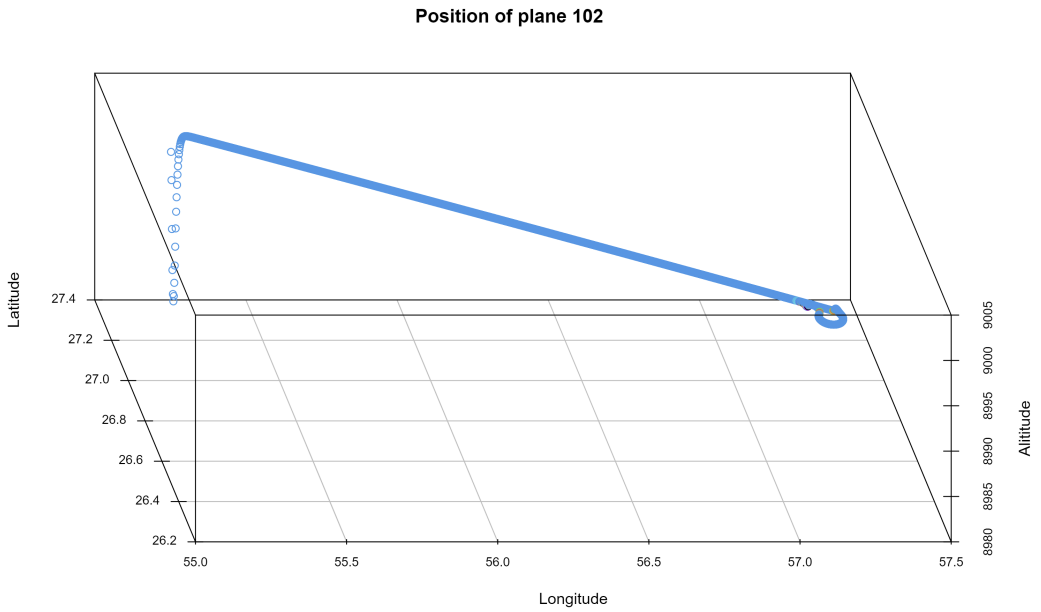

图 1: 102 战机飞行轨迹

对照输出的动作序列对比分析,可知前半段飞行基本吻合,而后半段,即 Unix.time>1306904185 时主要以盘旋为主，盘旋段的飞行状态如图 2 所示:

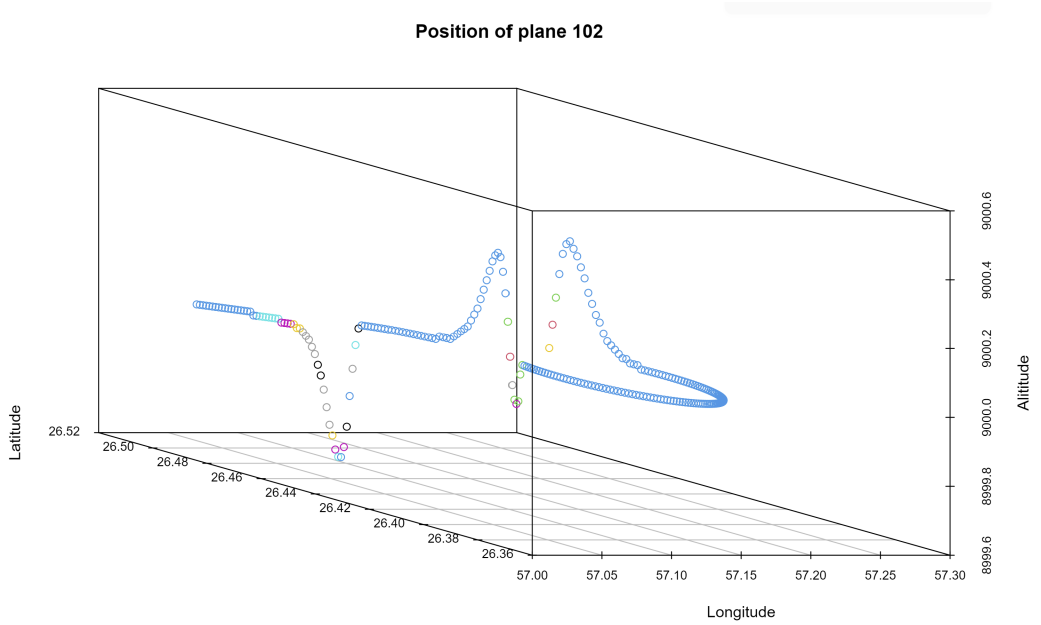

图 2: 102 战机盘旋轨迹

同样和输出序列相吻合，同时可以看到高度的微小变化并没有导致动作序列的识别出现问题，说明本模型的稳定性较好. 另一方面，由于上面的例子的动作相对单一，下面选取一架动作较剧烈的战机进行分析，这里选取 108 号飞机，它的主要参数及变化如图 3 所示.

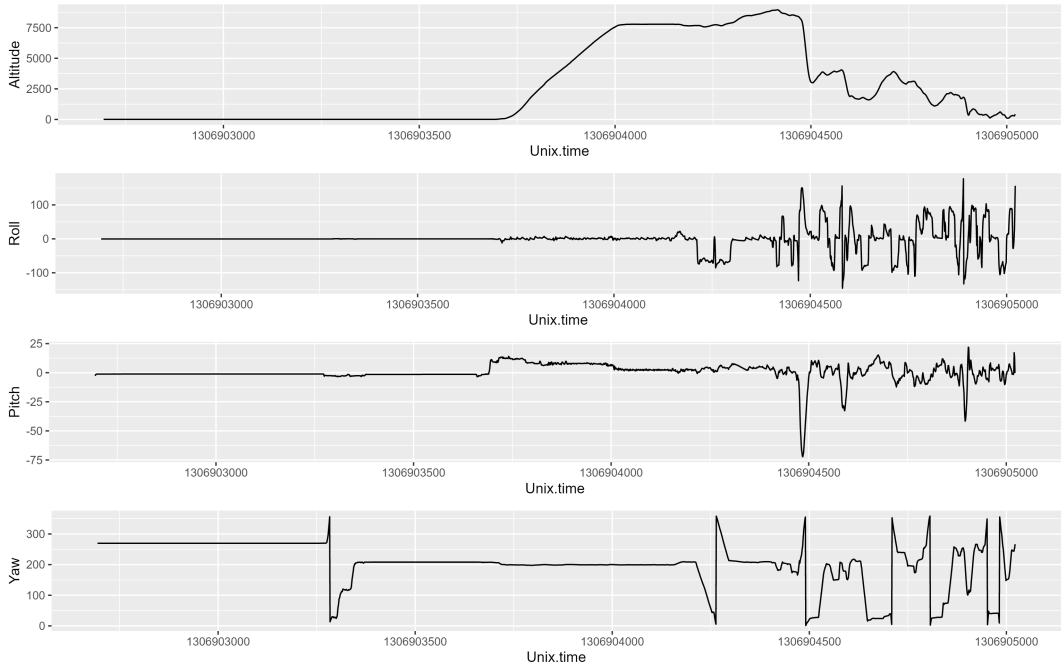

图 3: 108 战机参数变化图

可以看到在后半段它进行了复杂的机动，经可视化得到下面的三维散点图，其中颜色变化代表角度变化.

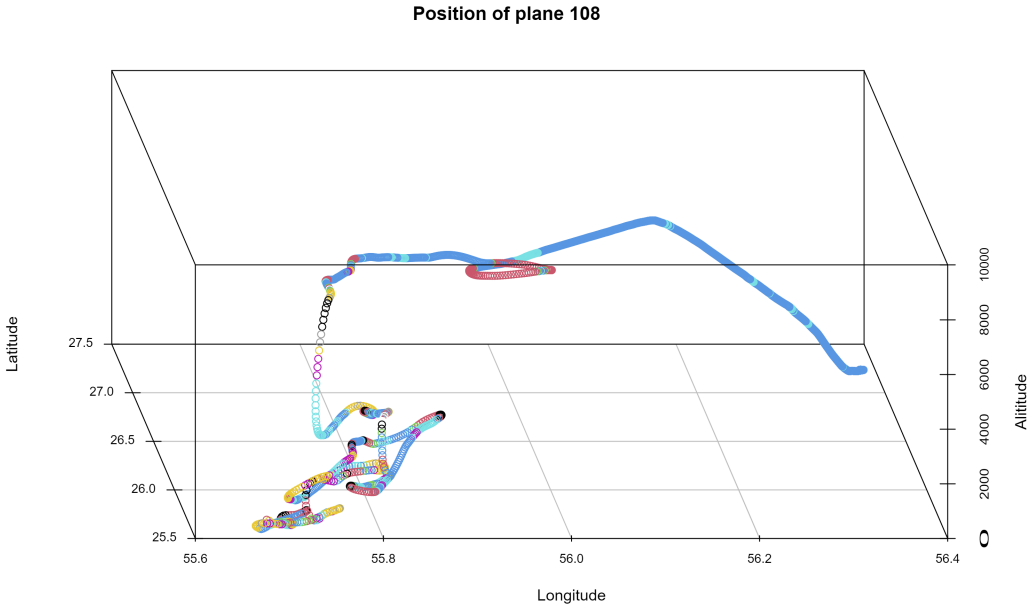

图 4: 108 战机动作轨迹

可以看出它的飞行动作大概是先爬升，在一段平飞后进行盘旋，在一系列转弯后进行俯冲然后是一些复杂的机动，这同输出序列基本一致.

最后我们在输出序列中选取一段机动动作来观察原数据是否确实代表该动作，这里选择

10E 号战机, 根据输出序列它在 1306904370<Unix.time<1306904440 时在进行下盘旋，经过可视化后确实如此，如下图所示：

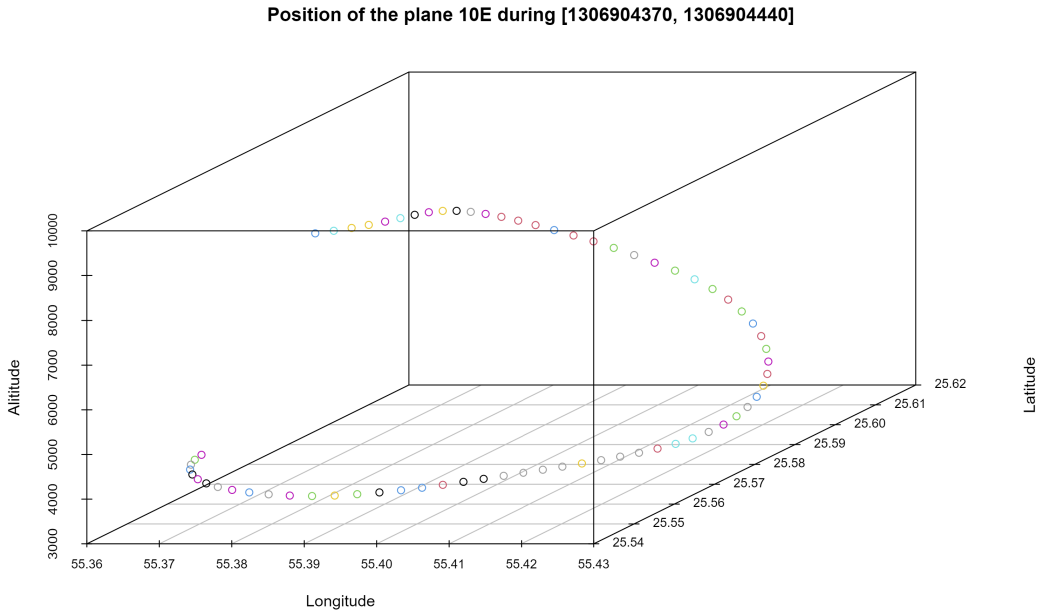

图 5: 10E 战机动作轨迹

以上三个例子基本说明了在一般情况下，本模型识别的飞行动作基本与实际动作一致，具备较高稳定性，且由于是使用一段较稳定序列分析动作种类，结合线性拟合分析整体特征，因此具备较好的稳定性. 此外若可以提供验证数据可以进一步使用学习算法进行优化并做更准确的模型检验.

六、问题三

（一）问题分析与准备

该问题需要基于问题二的结果，将机动动作序列切割/组合得到子序列，并以此判定背后隐含的意图.

在查阅相关文献后可知，战斗机的的目标意图集合包括攻击、侦察、监视、突防、防御、电子干扰共 6 种类型 [?]. 为了简化模型并进行合理的假设和建模，我们将突防、攻击统一归类为攻击；侦察、监视统一归类为侦察；并且忽略题目所给指标无法检测出的电子干扰意图，将目标意图集合简化为：攻击、防御、侦察三大类. 至此，我们规定所有的动作序列都必须围绕以上三种目的意图进行展开.

在上述规定下，战斗机在战场中做出的飞行动作一定是为达成攻击、防御、侦察中的任一目的而完成的. 所以说，如果能够识别出当前时间段内战斗机的战术动作的类型，在一定程度就可以反映出当前飞机的真实意图. 基于以上思路，子序列算法的核心就在于如何将零

散的动作序列组合，得到一段连续并可以判定的战术动作，用于意图推断. 考虑到战斗机在做同一个战术动作时连续性好，必然会产生大量的过渡，故动作前后相似度较高；而在切换不同战术动作时，动作序列的前后相似度必然降低. 所以我们设计了启发式相似度计算函数来对动作序列进行组合，将相似度高的动作序列合并、完全不同的动作序列分离，使得每个子序列都能对应一种能够反映意图的战术动作.

部分复杂机动动作所对应的战略意图如下所示：

（1）平飞机动：用于追赶、截获敌方；挣脱敌方，退出战斗；进入另一机动的预备状态.

（2）转弯：躲避敌方导弹的规避激动.

（3）半斤斗：用于攻击超过遭遇航线的敌机.

（4）战术转弯：用于攻击敌机；作为防御机动.

（5）急拉起：攻击超高度飞行的敌机；在有速度优势的情况下摆脱敌机；躲避导弹攻击.

在实战中，飞行员会根据战斗时战场的整体情况和自身条件安排下一步的动作，当动作的意图出现歧义时，就需要结合飞机当前情况和战场整体态势，衡量当前子序列中战机的攻击性，若当前飞机优势大，攻击性强，则动作意图更偏向于攻击；若当前飞机处于劣势，且攻击性弱，则动作意图更倾向于防御；若当前飞机处境安全，攻击性不高也不低，则动作意图更倾向于侦察.

为量化某一飞机在一段序列 Δ_i 中的意图偏向，现定义一个优势度 $I, I \in \{0, 1\}$. 分析可得，I 的值越接近 1，越可能攻击；反之越可能防御. 至此，完成了建立模型前的一系列准备工作.

（二）模型建立

1. *机动动作组合算法*

基于以上分析，我们设计出启发式相似度计算函数来对机动动作序列进行组合. 设当前战斗的时间序列（即 unixtime）共 m 个，集合 Ω_i 为第 i 个 $(i = 1, 2, 3\ldots)$ 子序列中包含的所有基本动作；集合 A_k 表示第 k$(k = 1, 2, 3\ldots, k <= m - 1)$ 时刻时战斗机被识别出的所有基本动作. 若此时 A_k 属于子序列 Ω_i，而集合 $A_k + 1$ 满足以下公式

$$A_k \subseteq A_{k+1} \tag{7}$$

或：

$$A_{k+1} \subseteq A_k \tag{8}$$

那么我们就认为 k 时刻的机动动作和 $k + 1$ 时刻的机动动作相似度较高，属于同一个动作子序列，并将第 $k + 1$ 时刻中的基本动作纳入到集合 Ω_i 中；如果不满足以上的条件函数，那么我们就认为这两个动作前后不连贯，$k + 1$ 时刻的机动动作需要与 k 时刻的动作分开，并

划分到集合 Ω_{i+1} 中. 利用以上算法模型对所有的机动动作序列进行遍历，就可以得到完整的子序列以及每个子序列中的基本动作集合 Ω_i.

在得到了子序列的基础上，我们需要进一步判断出每个子序列所对应的战术动作，以此来初步推断子序列的战术意图. 由于我们在上一步中已经得到了集合 $\Omega_i, i = 1, 2, 3, 4\ldots$ 的结果，所以就将集合 Ω_i 中的基本动作全部列出，若集合内的基本动作只有一种类型，则对照基本动作表 2 进行判定；若集合 Ω_i 内存在多种基本动作，如 $\Omega_i = \{2, 3, 4\}$，则需要遍历复杂动作表 3，只要 Ω_i 中的任一动作参与了某复杂动作的组成，那么就要用该复杂动作的定义，对该子序列进行判定. 若判定结果符合，那么就将该子序列判定为该复杂动作；若判定结果不符合，则继续遍历直到找到对应战术动作为止. 通过分析可得，以上所有的复杂动作定义全部互斥，不相互包含，故最中可以得到所有子序列唯一对应的战术动作.

结合文献参考以及表 3 中定义，所有复杂动作反映出的目标意图如表 6 所示.

表 6: 机动动作意图对照表

目标意图	动作类型
攻击	急盘降，俯冲，急拉起，战术转弯，高速 Yo-Yo, 低速 Yo-Yo，上升，下滑，翻滚
防御	俯冲，急拉起，半斤斗，半滚倒转，桶滚，战术转弯，规避急转弯，上升，下滑，转弯，翻滚
侦察	平飞，定常盘旋

在对上表进行合理分析后，可以得出以下结论：

（1）唯一指明攻击意图的动作：急盘降、高速 Yo-Yo,、低速 Yo-Yo.

（2）唯一指明防御意图的动作：半斤斗、半滚倒转、桶滚、规避急转弯、转弯.

（3）唯一指明侦察意图的动作：定长盘旋.

（4）多种意图公用的动作：俯冲、急拉起、战术转弯、平飞、上升、下滑、翻滚.

如果说子序列对应的战术动作能够唯一指明意图，那么通过对照表 6 就可以直接判断出目标意图；若该战术动作可能对应多种意图，则需要进一步建立本机优势函数模型对当前战机状态进行判断，从而推导出目标意图.

2. 本机优势模型

通过分析可知，本机的优势需要紧密结合当前战场上的战斗情况进行判断. 一般认为，空战中机距离小于 37km 称为视距内作战；机距大于 37km 时则称为视距外作战 [**?**]. 在对附录 1 中提供的飞行数据进行统计处理后，我们得出所有飞机相距的最大距离达到了 222.3447km.

因此该作战既包含视距内作战，也包含视距外作战. 考虑到战斗机在视距内和视距外的攻击手段有很大差别，所以这里需要进行分类讨论.

（1）超视距作战

1）角度优势

角度优势的定义如下：若作战时，目标飞机在本机攻击角度以内或处于易于攻击的角度范围内，则认为本机在此时的空战中占据角度优势. 角度优势的建立依靠本机和目标间的进入角 q 和方位角 φ. 当敌我双方在迎头接近时，雷达探测范围最大，并且可以在最远的距离发射空空导弹，所以优势最大；同理, 在敌机尾部时，雷达探测范围变小，角度优势较小 [**?**]. 基于该思路，设进入角优势为 R_q，方位角优势为 R_φ, 可以得到两架战机交战时

$$R_q = \begin{cases} e^{-\frac{|180-\varphi-q|}{180}\pi}, & \varphi \geq 0 \\ e^{-\frac{|180+\varphi+q|}{180}\pi}, & \varphi < 0 \end{cases} \tag{9}$$

已知本机和敌机位置点及经纬度分别为 $P_1 = (\theta_1, \phi_1), P_2 = (\theta_2, \phi_2)$，这里将直线距离作为真实距离，并引入一个辅助点 $P_3 = (\theta_1, \phi_2)$，因此可得方位角为

$$\varphi = \arccos \frac{d(P_1, P_3)}{d(P_1, P_2)} \tag{10}$$

其中 $d(P_1, P_2)$ 为 P_1, P_2 之间的欧氏距离.

另一方面，设 Φ_1 为机载雷达得最大搜索方位角、Φ_2 为我方战机载弹的最大离轴发射角、Φ_3 为我方战机的不可逃逸角度，则可以建立方位角优势函数为

$$R_\varphi = \begin{cases} 0.1 - \dfrac{|\varphi| - 85}{10 \times (180 - 85)}, & |\varphi| \geq 85 \\ 0.2 - \dfrac{|\varphi| - \Phi_1}{10 \times (180 - \Phi_1)}, & \Phi_1 \leq |\varphi| \leq 85 \\ 0.3 - \dfrac{|\varphi| - \Phi_1}{10 \times (\Phi_1 - \Phi_2)}, & \Phi_2 \leq |\varphi| \leq \Phi_1 \\ 0.8 - \dfrac{|\varphi| - \Phi_2}{2 \times (\Phi_2 - \Phi_3)}, & \Phi_3 \leq |\varphi| \leq \Phi_2 \\ 1 - \dfrac{|\varphi|}{5\Phi_3}, & 0 \leq |\varphi| \leq \Phi_3 \end{cases} \tag{11}$$

2）距离优势

在超视距作战中，距离对态势的影响主要是雷达探测概率和空空导弹的杀伤概率上 [**?**]. 对于 RCS 一定的敌机，我机与敌机的距离越大，雷达的发现概率越小；而导弹的杀伤概率则是在某一距离内较大，但在可发射区域的最远和最近处杀伤概率小，且变化较快.

类似于角度优势，设 D_1, D_2, D_3, D_{min} 分别为机载雷达最大探测距离、我机载弹最大有效

距离、我机导弹不可逃逸距离、以及导弹最小有效距离，则可建立两机的距离优势函数

$$R_d = \begin{cases} 0.1839 \exp\left(-\dfrac{d - D_1}{D_1}\right), & d \geq D_1 \\ 0.5 \exp\left(-\dfrac{d - D_1}{D_1 - D_2}\right), & D_2 \leq d \leq D_1 \\ 0.2 \exp\left(-\dfrac{d - D_3}{D_2 - D_3}\right), & D_3 \leq d \leq D_2 \\ 1, & D_{min} \leq d \leq D_3 \\ 2^{-\frac{d - D_3}{2 - D_3}}, & 0 \leq d \leq D_{min} \end{cases} \tag{12}$$

当存在多机空战时，若有 n 架敌机在视距外，则可以建立本机的距离优势函数为

$$R_d = \frac{1}{n} \sum_{i=1}^{n} R_d^{(i)} \tag{13}$$

其中，$R_d^{(i)}$ 为本机相对于第 i 架敌机的距离优势.

3）能量优势一般来说，在超视距空战中，战机速度越大，其携带的空空导弹对敌方的威胁越大. 该优势由高度优势和速度优势共同组成，由此建立能量优势函数 R_e

$$R_e = \begin{cases} 1, & E \geq 2E' \\ 1 + \dfrac{0.9(E - 2E')}{\frac{3}{2}E'}, & \dfrac{1}{2}E' \leq E \leq 2E' \\ 0.1, & \dfrac{1}{2}E' \leq d \leq 2E' \end{cases} \tag{14}$$

当存在多机空战时，若有 n 架敌机在视距外，则可以建立本机的距离优势函数为:

$$R_e = \frac{1}{n} \sum_{i=1}^{n} R_e^{(i)} \tag{15}$$

其中，$R_e^{(i)}$ 为本机相对于第 i 架敌机的能量优势.

4）事件优势

一般发射本机携带的导弹的战机是在该段时间中占优势的一方，而且导弹的发射会进一步提高发射方的优势度. 因此, 从导弹的发射情况也可以来描述战场形势，这里为简化模型只考虑了诱饵弹和导弹，因此建立两机的事件优势 R_f 为

$$R_f = \begin{cases} 0, & \text{无动作} \\ 0.5, & \text{发射诱导弹} \\ 1, & \text{发射导弹} \end{cases} \tag{16}$$

当存在多机空战时，若有 n 架敌机在视距外，则可以建立本机的事件优势函数为

$$R_f = \frac{1}{n}\sum_{i=1}^{n} R_f^{(i)} \tag{17}$$

其中,$R_f^{(i)}$ 为本机相对于第 i 架敌机的事件优势.

综合以上的各个优势，可建立视距外空战中本机优势函数为

$$BS_1 = \lambda_a \times R_a + \lambda_d \times R_d + \lambda_e \times R_e + \lambda_f \times R_f \tag{18}$$

其中，$\lambda_a + \lambda_d + \lambda_e + \lambda_f = 1$.

（2）视距内空战

1）角度优势

角度在视距内空战中同样很重要，设敌机的方位角和进入角分别为 φ 和 q，角度优势为 R_a，则可以构建两机角度优势函数

$$R_a = 1 - \frac{|\varphi| + |q|}{180} \tag{19}$$

当存在多机空战时，若有 n 架敌机在视距内，则本机角度优势函数为:

$$R_a = \frac{1}{n}\sum_{i=1}^{n} R_a^{(i)} \tag{20}$$

其中,$R_a^{(i)}$ 为本机相对于第 i 架敌机的角度优势.

2) 距离优势

由于在视距内空战中攻击手段主要为机载机枪和近距炮，这些炮弹的杀伤性呈现先上升后下降的态势，在最佳攻击距离的杀伤性最好，两侧逐渐降低，设距离优势为 R_d，则两机距离优势函数为

$$R_d = \exp(-(\frac{d - D_0}{2(D_{max} - D_{min})})^2) \tag{21}$$

其中 D_0, D_{max}, D_{min} 分别为最佳攻击距离，最大攻击距离，和最小攻击距离.

当存在多机空战时，若有 n 架敌机在视距内，则本机角度优势函数为

$$R_d = \frac{1}{n}\sum_{i=1}^{n} R_d^{(i)} \tag{22}$$

其中，$R_d^{(i)}$ 为本机相对于第 i 架敌机的距离优势.

由于导弹的射程数据较难获得，且一般最多只能获得最大攻击距离，现对上式做了如下

简化

$$R_d = \exp(-(\frac{d - \frac{D_{max}}{2}}{D_{max}})^2) \tag{23}$$

其中 D_{max} 取 30km.

3）事件优势

设时间优势为 R_f, 与超视距相同建立事件优势函数:

$$R_f = \begin{cases} 0, & \text{无动作} \\ 0.5, & \text{发射诱导弹} \\ 1, & \text{发射导弹} \end{cases} \tag{24}$$

当存在多机空战时，若有 n 架敌机在视距内，则可以建立本机的事件优势函数为

$$R_f = \frac{1}{n}\sum_{i=1}^{n} R_f^{(i)} \tag{25}$$

其中,$R_f^{(i)}$ 为本机相对于第 i 架敌机的事件优势.

对于视距内空战，设速度优势为 R_v，高度优势为 R_h，可知对应的速度优势函数和高度优势函数如下

4）速度优势

$$R_v = \begin{cases} 0.1, & v < 0.6v' \\ -0.5 + \dfrac{v}{v'}, & 0.6v' \le v \le 1.5v' \\ 1, & v > 1.5v' \end{cases} \tag{26}$$

其中，v, v' 分别为我机和敌机的速度. 对于多机空战，若有 n 架敌机在视距内，则可以建立本机的速度优势函数为

$$R_v = \frac{1}{n}\sum_{i=1}^{n} R_v^{(i)} \tag{27}$$

其中，$R_v^{(i)}$ 为本机相对于第 i 架敌机的速度优势.

5）高度优势

$$R_h = \begin{cases} 0, & H - H' \leq -5km \\ 0.5 + 0.1(H - H'), & -5km \leq H - H' \leq 5km \\ 1, & v > 5km \end{cases} \tag{28}$$

其中 H, H' 分别为我机和敌机的高度. 对于多机空战，若有 n 架敌机在视距内，则可以建立本机的高度优势函数为

$$R_h = \frac{1}{n} \sum_{i=1}^{n} R_h^{(i)} \tag{29}$$

其中，$R_h^{(i)}$ 为本机相对于第 i 架敌机的速度优势.

综合以上的各个优势, 设视距内空战中本机优势为，可建立其优势函数为

$$BS_2 = \lambda_a \times R_a + \lambda_d \times R_d + \lambda_v \times R_v + \lambda_h \times R_h + \lambda_f \times R_f \tag{30}$$

其中，$\lambda_a + \lambda_d + \lambda_v + \lambda_h + \lambda_f = 1$.

若敌方当前共有 N 架飞机，设 $BS_1^{(i)}, BS_2^{(i)}$ 分别为本机相对敌方第 i 架飞机的优势函数. 综合视距内和视距外的优势函数，可以建立本机优势函数

$$\overline{BS} = \frac{1}{N} \sum_{i=1}^{N} (\frac{1}{2}(1 + \text{sgn}(d_i - d_0))BS_1^{(i)} + \frac{1}{2}(1 - \text{sgn}(d_i - d_0))BS_2^{(i)}) \tag{31}$$

其中，$\overline{BS}$ 为本战机面对多架敌机时的战斗优势，d_i 是两机相对距离，d_0 是视距距离. 据分析可知，$\overline{BS}$ 的值越大，战斗机优势越明显.

由于超视距建模需要较多参数，而从题目中可以获得的信息不足以支持超视距情况下的建模，所以我们最终采用视距内的情况进行模型的建立，此时 $\overline{BS} = BS_2$. 所以此时需要确定 $BS_2 = \lambda_a \times R_a + \lambda_d \times R_d + \lambda_v \times R_v + \lambda_h \times R_h + \lambda_f \times R_f$ 中的参数取值. 在查询文献，并结合大量数据统计后，我们最终设定了以上 5 个参数的具体取值：$\lambda_a = 0.25, \lambda_d = 0.1, \lambda_v = 0.15, \lambda_h = 0.25, \lambda_f = 0.25$. 最终得到本机优势 BS 的最终函数表达式

$$BS = 0.25 \times R_a + 0.1 \times R_d + 0.15 \times R_v + 0.25 \times R_h + 0.25 \times R_f \tag{32}$$

上式中 R_a, R_d, R_v, R_h, R_f 的取值全部依据视距内的取值方法进行运算，$BS \in \{0, 1\}$. 至此，我们求出了战斗中任一架战机的优势值 I，$I = BS$. 优势度的取值范围与最终的目的意向间的关系如表 7 所示.

表 7: 战机优势度——意向关系表

优势值	目标意向
$[0, 0.4)$	防御
$[0.4, 0.6)$	侦察
$[0.6, 1)$	攻击

（二）模型验证

为了验证模型合理性，选择文件 51st Bisons vs CNF Rd 1 中 Unix time 在 1306904380 和 1306904457 中的飞行数据进行分析. 在这段时间内，Red 阵营发射一枚导弹 205. 导弹的轨迹和飞机的轨迹如图 1 所示. 其中，红色轨迹代表战斗机航线，蓝色代表导弹轨迹.

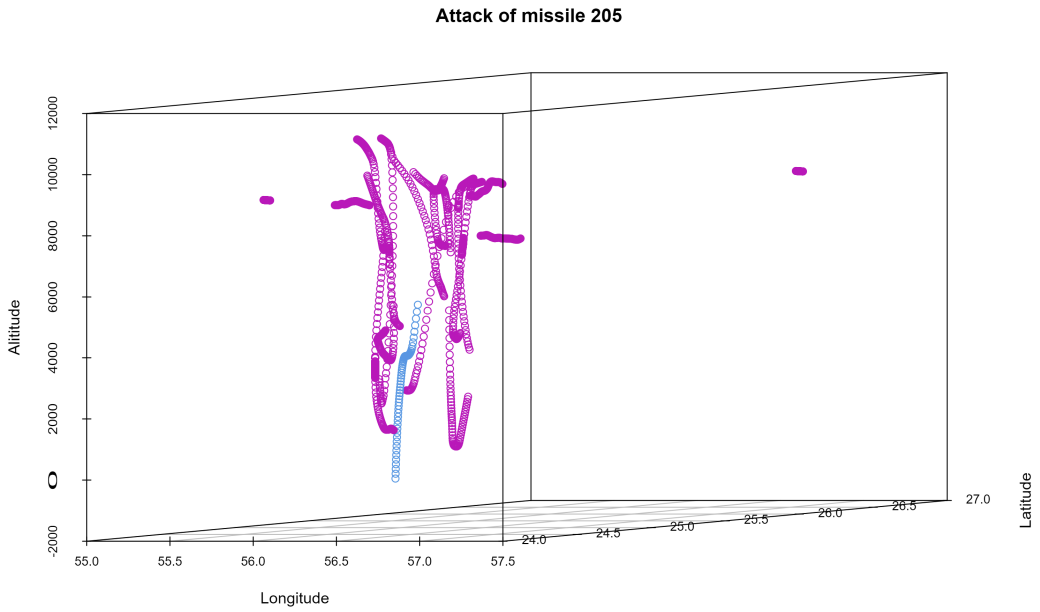

图 6: 导弹飞行轨迹

根据上述分析，当 Red 阵营发射导弹时，其阵营飞机应大部分具有攻击状态，对应飞机的 BS 值应该在阈值 0.5 之上. 通过所得到的结果文件可以看到，在 1306904380 到 1306904457 对应时间内，飞机 106、10C、105、10F 具有 0.5 以上的 BS 值，而这四架飞机均为 Red 阵营的飞机，其余飞机 BS 值在 0.5 之下，即 Blue 阵营大多处于防御状态，与实际情况相符合，证明所得到的模型具有较高的精确度.

七、问题四

(一) 模型建立

战场整体态势需要考虑战场上敌我双方的所有战机的优势程度，若敌我双方在某一时刻的飞机数量分别为 N_1, N_2，敌方第 i 架飞机的优势函数为 $\overline{BS}_i^{(1)}$，我方第 i 架飞机的优势函数为 $\overline{BS}_i^{(2)}$，则战场态势函数为

$$\Gamma = \sum_{j=1}^{N_2} \overline{BS}_j^{(2)} - \sum_{i=1}^{N_1} \overline{BS}_i^{(1)} \tag{33}$$

分析上述式子可知，在 $\Gamma > 0$ 时我方占优势，在 $\Gamma < 0$ 时敌方占优势.

若已知两点 P_1, P_2 的经纬度和所在高度分别为 $(\theta_1, \phi_1, h_1), (\theta_2, \phi_2, h_2)$，则它们之间的距离函数为

$$d(P_1, P_2) = \sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2 + \Delta_h^2} \tag{34}$$

其中

$$\begin{aligned} \Delta_x &= R(\cos(\theta_1)\cos(\phi_1) - \cos(\theta_2)\cos(\phi_2)) \\ \Delta_y &= R(\sin(\theta_1)\cos(\phi_1) - \sin(\theta_2)\cos(\phi_2)) \\ \Delta_z &= R(\sin(\phi_1) - \sin(\phi_2)) \\ \Delta_h &= |h_1 - h_2| \end{aligned} \tag{35}$$

方向矢量 $\boldsymbol{P_1P_2}$ 为 P_1 指向 P_2 的矢量，经分析可得

$$\boldsymbol{P_1P_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) \tag{36}$$

其中对 $i = 1, 2$，有

$$\begin{aligned} x_i &= (h_i + R)\cos(\theta_i)\cos(\phi_i) \\ y_i &= (h_i + R)\sin(\theta_i)\cos(\phi_i) \\ z_i &= (h_i + R)\sin(\phi_i) \end{aligned} \tag{37}$$

用以 P_2 为起始点垂直于它与地心的连线，且方向表征偏航角的单位向量 $\boldsymbol{v} = (x_v, y_v, z_v)$ 近似 P_2 的速度向量，由几何关系，引入一个点 $Q = (0, 0, \frac{h_2+R}{\cos(\phi_2)})$ 则有下列方程组

$$\begin{aligned} \boldsymbol{v} \cdot \boldsymbol{OP_2} &= 0 \\ < \boldsymbol{v}, \boldsymbol{P_2Q} > &= \text{Yaw} \\ |\boldsymbol{v}| &= 1 \end{aligned} \tag{38}$$

首先可以显式地解出 $z_v = \frac{|QP_2|\cos(\text{Yaw})}{(h_2+R)}\cos(\phi_2)$，这样原方程组就变为

$$\begin{aligned} x_2 \times x_v + y_2 \times y_v &= -z_2 \times z_v \\ x_v^2 + y_v^2 &= 1 - z_v^2 \end{aligned} \tag{39}$$

则进入角为

$$q = \arccos(v \cdot \frac{P_1P_2}{|P_1P_2|}) \tag{40}$$

(二) 模型验证

同样选择文件 51st Bisons vs CNF Rd 1 中 Unix time 在 1306904380 和 1306904457 中的飞行数据进行验证本模型的合理性. 经计算红方各架飞机的优势函数之和明显大于蓝方，即战场态势函数显著大于 0，因此模型推测此时红方在战场中占优势，这与实际情况相吻合，这说明本文的模型可以较真实地反映双方的交战情况.

参考文献

[1] 倪世宏, 史忠科, 谢川, 王彦鸿. 军用战机机动飞行动作识别知识库的建立 [J]. 计算机仿真,2005(04):23-26.

[2] 王玉伟, 高永. 基于飞行基本动作对复杂动作识别方法研究 [J]. 舰船电子工程,2018,38(10):74-76+168.

[3] 张曙光, 韩彦东, 王永正, 等. 高性能战斗机机动动作优化控制研究 [J]. 飞行力学,1997(1):50-55.

[4] 刘钻东, 陈谋, 吴庆宪, 陈哨东. 非完备信息下无人机空战目标意图预测 [J]. 中国科学: 信息科学,2020,50(05):704-717.

[5] 曹慧敏, 黄安祥, 雷祥. 空战临战态势评估方法研究 [J]. 系统仿真学报,2019,31(02):257-262+274.DOI:10.16182/j.issn1004731x.joss.17-0218.

[6] 王德鑫, 刘忠, 黄金才. 空战中基于威胁评估的任务规划 [J]. 火力与指挥控制,2007(12):24-27.

附件

附件清单：
附件 1：第一问 Python 代码
附件 2：第二问 Python 代码
附件 3：第三问 Python 代码
附件 4：第四问 Python 代码

附件 1：第一问 Python 代码

```
import pandas as pd

original_file = pd.read_csv('data/51st Bisons vs CNF Rd 1.csv', encoding='gbk')

Ids = original_file['Id'].values

names = []

for Id in Ids:
    found = False
    type_name = original_file.loc[original_file['Id'] == Id]['Type'].values[0]
    if type_name.find('Air') == -1:
        continue
    for name in names:
        if Id == name:
            found = True
            break
    if not found:
        names.append(Id)

print(names)
```

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np

def decompose(a):
    ans = []
    while a >= 1:
        ans.append(int(a % 10))
        a = int(a / 10)
    return ans
```

```
def calc_b(y):
	reg = LinearRegression()
	len_ = len(y)
	y = y.reshape(-1, 1)
	x = [i for i in range(1, len_ + 1)]
	x = np.array(x)
	x = x.reshape(-1, 1)
	reg.fit(x, y)
	return reg.coef_, reg.score(x, y)

def calc_similarity(a, b):
	a_digits = decompose(a)
	b_digits = decompose(b)

	all_in_a = True
	for d in a_digits:
		if d not in b_digits:
			all_in_a = False
			break

	all_in_b = True
	for d in b_digits:
		if d not in a_digits:
			all_in_b = False
			break

	return all_in_a or all_in_b

def solve_3_1(data):
	start = 0
	while data[start] == 0:
		start += 1

	end = len(data)
	intervals = []
	sss = True

	while start < end - 1 and sss:
		cur = start
		flag = True
		while flag:
			if data[cur] == -1:
				sss = False
				break
```

```
                if data[cur + 1] == -1:
                    sss = False
                    break
                flag = calc_similarity(data[cur], data[cur + 1])
                cur += 1
            interval = str(start) + '-' + str(cur - 1)
            # print(interval)
            intervals.append(interval)
            start = cur
    return intervals
```

附件 2：第二问 Python 代码

```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np


# y = b * x + a
def calc_b(y):
    reg = LinearRegression()
    len_ = len(y)
    y = y.reshape(-1, 1)
    x = [i for i in range(1, len_ + 1)]
    x = np.array(x)
    x = x.reshape(-1, 1)
    reg.fit(x, y)
    return reg.coef_, reg.score(x, y)


def decide_basic_action(yaw, pitch, roll, ground_dist):
    yaw_b, yaw_r2 = calc_b(yaw)
    pitch_b, pitch_r2 = calc_b(pitch)
    roll_b, roll_r2 = calc_b(roll)
    ground_dist_b, ground_dist_r2 = calc_b(ground_dist)

    threshold = 0.95
    if yaw_r2 < threshold or pitch_r2 < threshold or roll_r2 < threshold or
ground_dist_r2 < threshold:
        return 0

    ans = []
    if (yaw_b > -1) and (yaw_b < 1) and (ground_dist_b > -2) and (ground_dist_b <
2):
        ans.append(1)
```

```
        if ground_dist_b > 2:
                ans.append(2)
        if ground_dist_b < -2:
                ans.append(3)
        if yaw_b > 1:
                ans.append(4)
        if roll_b > 10:
                ans.append(5)
        if len(ans) == 0:
                return 0
        else:
                return ans


original_filename = '51stKIAP_vs_107th_Round_1'
ttmp = 'data/' + original_filename + '.csv'
original_file = pd.read_csv(ttmp)
Ids = original_file['Id'].values
names = []

time_min = original_file['Unix time'].min()
time_max = original_file['Unix time'].max()


time_file_name = original_filename + '_time' + '.txt'

df = pd.DataFrame(index=range(time_max - time_min + 30))

for Id in Ids:
        found = False
        type_name = original_file.loc[original_file['Id'] == Id]['Type'].values[0]
        if type_name.find('Air') == -1:
                continue
        for name in names:
                if Id == name:
                        found = True
                        break
        if not found:
                names.append(Id)

for name in names:
        acts = []
        file = original_file[original_file.Id == name]
        start_time = file.iloc[0, 1] - time_min
```

```
    roll = file['Roll'].values
    roll = np.array(roll)
    pitch = file['Pitch'].values
    pitch = np.array(pitch)
    yaw = file['Yaw'].values
    yaw = np.array(yaw)
    ground_distance = file['Altitude'].values
    ground_distance = np.array(ground_distance)

    start = 0
    end = len(roll)
    cur = 0
    default_sliding_window_len = 10

    while cur < end:
        begin = cur
        sliding_window_len = default_sliding_window_len
        act = decide_basic_action(yaw=yaw[begin:begin + sliding_window_len],
                                  pitch=pitch[begin:begin + sliding_window_len],
                                  roll=roll[begin:begin + sliding_window_len],
                                  ground_dist=ground_distance[begin:begin + sliding_window_len]
                                  )
        while act == 0:
            sliding_window_len -= 1

            act = decide_basic_action(yaw=yaw[begin:begin + sliding_window_len],
                                      pitch=pitch[begin:begin + sliding_window_len],
                                      roll=roll[begin:begin + sliding_window_len],

ground_dist=ground_distance[begin:begin + sliding_window_len]
                                      )
        for i in range(sliding_window_len):
            tmp = ''
            for a in act:
                tmp += str(a)
            acts.append(tmp)
        cur += sliding_window_len
```

```
    for i in range(df.shape[0] - len(acts) - start_time):
        acts.append('-1')

    df = df.astype('object')
    df[name] = '0'
    df.loc[start_time:, name] = acts

print(df)
tttmp = 'ques2_' + original_filename + '.csv'
df.to_csv(tttmp)
```

附件 3：第三问 Python 代码

```
import pandas as pd
import numpy as np


def decompose(a):
    ans = []
    while a >= 1:
        ans.append(int(a % 10))
        a = int(a / 10)
    return ans


def calc_similarity(a, b):
    a_digits = decompose(a)
    b_digits = decompose(b)

    all_in_a = True
    for d in a_digits:
        if d not in b_digits:
            all_in_a = False
            break

    all_in_b = True
    for d in b_digits:
        if d not in a_digits:
            all_in_b = False
            break

    return all_in_a or all_in_b
```

```
file = pd.read_csv('ques2_51st Bisons vs CNF Rd 1.csv')


def solve_3_1(data):
    start = 0
    while data[start] == 0:
        start += 1

    end = len(data)
    intervals = []
    sss = True

    while start < end - 1 and sss:
        cur = start
        flag = True
        while flag:
            if data[cur] == -1:
                sss = False
                break
            if data[cur + 1] == -1:
                sss = False
                break
            flag = calc_similarity(data[cur], data[cur + 1])
            cur += 1
        interval = str(start) + '-' + str(cur - 1)
        # print(interval)
        intervals.append(interval)
        start = cur
    return intervals
```

```
import pandas as pd
import numpy as np
from tmp import solve_3_1
from tmp import calc_b


def calc_yaw(yaw):
    len_ = len(yaw)
    sum = 0
    for i in range(1, len_):
        if np.abs(yaw[i] - yaw[i - 1]) > 300:
            sum += 5
        else:
            sum += np.abs(yaw[i] - yaw[i - 1])
```

```
    return sum


def decide_action(yaw, pitch, roll, altitude, actions):
    len_ = len(yaw)
    acts = [0, 0, 0, 0, 0]

    # 统计区间内基本动作
    for a in actions:
        while a >= 1:
            acts[int(a % 10) - 1] += 1
            a = int(a / 10)

    # 判定定常盘旋
    threshold = 0.5
    height_threshold = 20
    roll_threshold = 10
    yaw_threshold = 10

    if acts[3] > 0 and calc_yaw(yaw=yaw) > 360 * threshold and np.max(altitude) - np.min(
            altitude) < height_threshold and np.min(
            np.abs(roll)) > roll_threshold:
        return 1

    # 判定急盘降
    altitude_b, _ = calc_b(altitude)
    if acts[3] > 0 and acts[2] > 0 and calc_yaw(yaw=yaw) > 360 * threshold and np.max(altitude) - np.min(
            altitude) > height_threshold and altitude_b < 0:
        return 2

    # 判定急盘升
    if acts[3] > 0 and acts[1] > 0 and calc_yaw(yaw=yaw) > 360 * threshold and np.max(altitude) - np.min(
            altitude) > height_threshold and altitude_b > 0:
        return 3

    # 判定俯冲
    if acts[2] > 0 and calc_yaw(yaw) < yaw_threshold and np.max(altitude) - np.min(
            altitude) > height_threshold and altitude_b < 0:
        return 4

    # 判定急拉起
```

```
    if acts[1] > 0 and calc_yaw(yaw) < yaw_threshold and np.max(altitude) - np.min(
            altitude) > height_threshold and altitude_b > 0:
        return 5

    # 判定半斤斗
    if acts[1] + acts[2] > 0 and np.max(pitch) - np.min(pitch) > 180 * threshold and
np.max(roll) - np.min(
            roll) < roll_threshold:
        return 6

    # 判定半滚倒转
    if acts[1] + acts[2] > 0 and np.max(pitch) - np.min(pitch) > 180 * threshold and
np.max(roll) - np.min(
            roll) > 180 * threshold:
        return 7

    # 判定滚筒
    if acts[4] > 0 and np.max(altitude) - np.min(altitude) < height_threshold and
calc_yaw(
            yaw) < yaw_threshold and np.max(roll) - np.min(roll) > 180 *
threshold:
        return 8

    # 判定战术转弯
    yaw_slope_threshold = 5
    yaw_b, _ = calc_b(yaw)
    if acts[3] > 0 and yaw_b > yaw_slope_threshold and calc_yaw(yaw) < 90 *
threshold:
        return 9

    # 判定规避急转弯
    if acts[3] > 0 and yaw_b > yaw_slope_threshold and calc_yaw(yaw) > 90 *
threshold:
        return 10

    # 判定低速 yoyo
    if acts[1] + acts[4] > 0 and altitude_b > 0 and np.max(roll) - np.min(roll) > 180 *
threshold:
        return 12

    # 判定高速 yoyo
    if acts[2] + acts[4] > 0 and altitude_b < 0 and np.max(roll) - np.min(roll) > 180 *
threshold:
        return 11
```

```
    return np.argmax(acts) + 13


name = '51st vs 36th R2.csv'
file_name = 'ques2_' + name
origin_name = './data/' + name
file = pd.read_csv(file_name)
origin = pd.read_csv(origin_name)

YAW = origin['Yaw'].values
ROLL = origin['Roll'].values
PITCH = origin['Pitch'].values
ALTITUDE = origin['Altitude'].values


time_min = origin['Unix time'].min()
time_max = origin['Unix time'].max()
df = pd.DataFrame(index=range(time_max - time_min + 30))
df = df.astype('object')

flag = 0

for ids in list(file.columns.values):
    flag += 1
    if flag == 1:
        continue
    data = file[ids].values
    intervals = solve_3_1(data)
    actions = []

    for interval in intervals:
        start, end = interval.split('-')
        start, end = int(start), int(end)
        action = decide_action(yaw=YAW[start:end+1], roll=ROLL[start:end+1],
pitch=PITCH[start:end+1], altitude=ALTITUDE[start:end+1],
actions=data[start:end+1])
        for i in range(end - start + 1):
            actions.append(action)

    df[ids] = '0'
    ff = origin[origin.Id == ids]
    start_time = ff.iloc[0, 1] - time_min
    for i in range(df.shape[0] - len(actions) - start_time):
```

```
            actions.append(-1)
        df.loc[start_time:, ids] = actions

tttmp = 'ques3_1_' + name
df.to_csv(tttmp)
```

附件 4：第四问 Python 代码

```
import numpy as np
import pandas as pd
import math
import sympy

R = 6371004



def calc_descartes_jd(jd, wd, h):
    jd = jd / 180 * math.pi
    wd = wd / 180 * math.pi
    return (R + h) * math.cos(jd) * math.cos(wd), (h + R) * math.sin(jd) * math.cos(wd), (h + R) * math.sin(jd)



def calc_descartes_hd(jd, wd, h):
    return h * math.cos(jd) * math.cos(wd), h * math.sin(jd) * math.cos(wd), h * math.sin(jd)



def calc_phi(jd1, wd1, jd2, wd2):
    x1, y1, z1 = calc_descartes_jd(jd1, wd1, 1)
    x2, y2, z2 = calc_descartes_jd(jd2, wd2, 1)
    x3, y3, z3 = calc_descartes_jd(jd1, wd2, 1)
    dis1 = math.sqrt(math.pow(x1 - x3, 2) + math.pow(y1 - y3, 2) + math.pow(z1 - z3, 2))
    dis2 = math.sqrt(math.pow(x1 - x2, 2) + math.pow(y1 - y2, 2) + math.pow(z1 - z2, 2))
    return math.acos(dis1 / dis2)



def calc_v_zv(jd, wd, gd, yaw):
    jd = jd / 180 * math.pi
    wd = wd / 180 * math.pi
    yaw = yaw / 180 * math.pi
    q = (gd + R) / math.cos(wd)
    x, y, z = calc_descartes_hd(jd, wd, gd)
```

```
    dis = math.sqrt(math.pow(0 - x, 2) + math.pow(0 - y, 2) + math.pow(q - z, 2))
    zv = dis * math.cos(yaw) / (gd + R) * math.cos(wd)
    return zv, x, y, z


def calc_v_sympy(jd, wd, gd, yaw):
    zv, x2, y2, z2 = calc_v_zv(jd, wd, gd, yaw)
    xv = sympy.Symbol('xv')
    yv = sympy.Symbol('yv')
    eq1 = x2 * xv + y2 * yv + z2 * zv
    eq2 = xv * xv + yv * yv + zv * zv - 1
    root = sympy.solve([eq1, eq2], [xv, yv])
    return root[0][0], root[0][1], zv


def calc_v_quick(jd, wd, gd, yaw):
    zv, x2, y2, z2 = calc_v_zv(jd, wd, gd, yaw)
    tmp = math.pow(z2 * x2 * zv, 2) - (math.pow(x2, 2) + math.pow(y2, 2)) *
((math.pow(zv, 2) - 1) * math.pow(y2, 2) + math.pow(z2, 2) * math.pow(zv, 2))
    if tmp <= 0:
        return 0.33, 0.33, 0.34
    xv = (- z2 * x2 * zv + math.sqrt(tmp)) / (math.pow(x2, 2) + math.pow(y2, 2))
    yv = - (z2 * zv + x2 * xv) / y2
    return xv, yv, zv


def calc_q(jd1, wd1, gd1, jd2, wd2, gd2, yaw):
    xv, yv, zv = calc_v_quick(jd2, wd2, gd2, yaw)
    x1, y1, z1 = calc_descartes_jd(jd1, wd1, gd1)
    x2, y2, z2 = calc_descartes_jd(jd2, wd2, gd2)
    t1 = (x2 - x1) / math.sqrt(math.pow(x2 - x1, 2) + math.pow(y2 - y1, 2) +
math.pow(z2 - z1, 2))
    t2 = (y2 - y1) / math.sqrt(math.pow(x2 - x1, 2) + math.pow(y2 - y1, 2) +
math.pow(z2 - z1, 2))
    t3 = (z2 - z1) / math.sqrt(math.pow(x2 - x1, 2) + math.pow(y2 - y1, 2) +
math.pow(z2 - z1, 2))
    tmp = xv * t1 + yv * t2 + zv * t3
    return math.acos(tmp)


def calc_ra(phi, q):
    return 1 - (math.fabs(phi) + math.fabs(q)) / math.pi
```

```
def calc_rd(jd1, wd1, gd1, jd2, wd2, gd2):
    x1, y1, z1 = calc_descartes_jd(jd1, wd1, gd1)
    x2, y2, z2 = calc_descartes_jd(jd2, wd2, gd2)
    dis = math.sqrt(math.pow(x1 - x2, 2) + math.pow(y1 - y2, 2) + math.pow(z1 - z2,
2))
    d_max = 30000
    return math.exp(- dis * math.log(2, math.e) / d_max)


def calc_rf(color, time):
    data = origin.loc[origin['Unix time'] == time]
    cnt = 0
    for i in range(data.shape[0]):
        if data.iloc[i, 13] == color and 'Weapon' in data.iloc[i, 12]:
            cnt += 1
    return cnt


def calc_rv(v1, v2):
    if v1 < 0.6 * v2:
        return 0.1
    elif 0.6 * v2 <= v1 <= 1.5 * v2:
        if v1 <= 0.0001:
            return 1
        return -0.5 + v1 / v2
    else:
        return 1


def calc_rh(h1, h2):
    if h1 - h2 <= -5000:
        return 0
    elif -5000 <= h1 - h2 <= 5000:
        return 0.5 + 0.1 * (h1 - h2) / 1000
    else:
        return 1


name = '51stKIAP_vs_107th_Round_1.csv'
file_name = 'ques2_' + name
origin_name = './data/' + name
file = pd.read_csv(file_name)
origin = pd.read_csv(origin_name)
enemies = {'Red': 'Blue', 'Blue': 'Red'}
```

```
ids = list(file.columns.values)
print(ids)
ids = ids[1:]
colors = {}
for id in ids:
	cor_file = origin.loc[origin['Id'] == id]
	colors[id] = cor_file['Color'].values[2]
coloms = {}
sdsadsd = 0
for col in list(origin.columns.values):
	coloms[col] = sdsadsd
	sdsadsd += 1



def solve(id, time):
	color = colors[id]
	enemy = enemies[color]
	all_enemy_data = origin.loc[(origin['Unix time'] == time) & (origin['Color'] == enemy)]
	my_data = origin.loc[(origin['Unix time'] == time) & (origin['Id'] == id)]
	if my_data.shape[0] == 0:
		return 0
	jd1 = my_data.iloc[0, coloms['Longitude']]
	wd1 = my_data.iloc[0, coloms['Latitude']]
	gd1 = my_data.iloc[0, coloms['Altitude']]
	v1 = my_data.iloc[0, coloms['TAS']]
	ra = 0
	rd = 0
	rf = 0
	rv = 0
	rh = 0
	for i in range(all_enemy_data.shape[0]):
		jd2 = all_enemy_data.iloc[i, coloms['Longitude']]
		wd2 = all_enemy_data.iloc[i, coloms['Latitude']]
		gd2 = all_enemy_data.iloc[i, coloms['Altitude']]
		v2 = all_enemy_data.iloc[i, coloms['TAS']]
		yaw = all_enemy_data.iloc[i, coloms['Yaw']]

		phi = calc_phi(jd1, wd1, jd2, wd2)
		q = calc_q(jd1, wd1, gd1, jd2, wd2, gd2, yaw)

		ra += calc_ra(phi, q)

		rd += calc_rd(jd1, wd1, gd1, jd2, wd2, gd2)
```

```
            rf += calc_rf(colors[id], time)

            rh += calc_rh(gd1, gd2)

            rv += calc_rv(v1, v2)

    return (ra * 0.25 + rd * 0.1 + rf * 0.15 + rh * 0.25 + rv * 0.25) /
all_enemy_data.shape[0]

time_min = origin['Unix time'].values[0]
time_max = origin['Unix time'].values[-1]
df = pd.DataFrame(index=range(time_min, time_max + 30))
df = df.astype('object')

cnt = -1
for id in ids:
    print(id)
    df[id] = 0
    cnt += 1
    filee = file[id].values
    for time in range(time_min, time_max):
        if filee[time - time_min] == 0:
            continue
        if filee[time - time_min + 15] == -1:
            break
        cur = time - origin['Unix time'].values[0]
        ans = solve(id, time)
        df.iloc[cur, cnt] = ans

tttmp = 'ques4_' + name
df.to_csv(tttmp)
```