**BB籃球策略註解檔**

**[Strategy\_main.cpp](#_Toc54107317)** [3](#_Toc54107317)

[**\_main()** 3](#_Toc54107318)

[**Strategy\_main.cpp** 4](#_Toc54107319)

[**\_strategymain()** 4](#_Toc54107320)

[**Strategy\_main.cpp** 7](#_Toc54107321)

[**\_Draw()** 7](#_Toc54107322)

[**Strategy\_main.cpp** 8](#_Toc54107323)

[**\_MoveHead()** 8](#_Toc54107324)

[**Strategy\_main.cpp** 9](#_Toc54107325)

[**\_MoveContinuous()** 9](#_Toc54107326)

[**Strategy\_main.cpp** 11](#_Toc54107327)

[**\_InversePerspective()** 11](#_Toc54107328)

[**Strategy\_main.cpp** 13](#_Toc54107329)

[**\_Triangulation()** 13](#_Toc54107330)

[**Strategy\_main.cpp** 14](#_Toc54107331)

[**\_image()** 14](#_Toc54107332)

[**Strategy\_main.cpp** 15](#_Toc54107333)

[**\_AreaSizeMeasure()** 15](#_Toc54107334)

[**Strategy\_main.cpp** 16](#_Toc54107335)

[**\_ComputeSpeed()** 16](#_Toc54107336)

[**Strategy\_main.cpp** 18](#_Toc54107337)

[**\_SelectBaseLine()** 18](#_Toc54107338)

[**Strategy\_main.cpp** 19](#_Toc54107339)

[**\_FindballInitial()** 19](#_Toc54107340)

[**Strategy\_main.cpp** 20](#_Toc54107341)

[**\_FindballHead()** 20](#_Toc54107342)

[**Strategy\_main.cpp** 21](#_Toc54107343)

[**\_TraceballHead()** 21](#_Toc54107344)

[**Strategy\_main.cpp** 23](#_Toc54107345)

[**\_TraceballBody()** 23](#_Toc54107346)

[**Strategy\_main.cpp** 26](#_Toc54107347)

[**\_FindbasketHead()** 26](#_Toc54107348)

[**Strategy\_main.cpp** 27](#_Toc54107349)

[**\_TracebasketHead()** 27](#_Toc54107350)

[**Strategy\_main.cpp** 28](#_Toc54107351)

[**\_TracebasketBody()** 28](#_Toc54107352)

[**Strategy\_main.cpp** 30](#_Toc54107353)

[**\_UPbasket()** 30](#_Toc54107354)

[**Strategy\_main.cpp** 31](#_Toc54107355)

[**\_SlamDunk()** 31](#_Toc54107356)

[**Loadparameter.cpp** 32](#_Toc54107357)

[**\_initparameterpath()** 32](#_Toc54107358)

[**Loadparameter.cpp** 33](#_Toc54107359)

[**\_LoadParameters()** 33](#_Toc54107360)

[**Walkparameter** 34](#_Toc54107361)

[**\_setWalkParameterInit()** 34](#_Toc54107362)

[**WalkContinuouse** 34](#_Toc54107363)

[**\_setWalkParameterMax()** 34](#_Toc54107364)

[**WalkContinuouse** 34](#_Toc54107365)

[**\_setWalkParameterMin()** 34](#_Toc54107366)

[**WalkContinuouse** 34](#_Toc54107367)

[**\_setWalkParameterExp()** 34](#_Toc54107368)

[**WalkContinuouse** 35](#_Toc54107369)

[**\_startContinuous()** 35](#_Toc54107370)

[**WalkContinuouse** 35](#_Toc54107371)

[**\_stopContinuous()** 35](#_Toc54107372)

[**WalkContinuouse** 35](#_Toc54107373)

[**\_checkDataValue()** 35](#_Toc54107374)

**Strategy\_main.cpp**

**\_****main()**

1. 初始化ROS並預設節點名稱：ros::init()

* 本策略預設節點名稱為BBthrow，節點名稱可更改

1. 創建節點的代理物件：ros::NodeHandle

* 本策略物件名稱命名為nh

1. 創建Rate的物件並設定節點的更新速率：ros::Rate loop\_rate(30)

* Rate的物件名稱命名為loop\_rate設定loop\_rate物件，使節點的更新速率為30Hz

1. 讀取參數路徑：Load->initparameterpath()

* 讀取strategy裡的參數(Parameter)路徑並存到變數裡
* 詳細請參照[initparameterpath()](#initparameterpath)

1. nh代理物件狀態是否正常：nh.ok()

* 當ros::ok()回傳false，代表發生以下其中一件事件
* SIGINT被觸發(Ctrl-C)執行了ros::shutdown()
* 被另一個同名節點踢出ROS網路
* ros::shutdown()被其他部分的程式執行
* 節點中所有的代理物件(ros::NodeHandles)都已經被銷毀

1. 訊息回撥處理函式：ros::spinOnce()

* 呼叫所有有變動Topic的回撥(callback)函式

1. 主策略：strategymain()

* 程式主要執行策略的成員函式
* 詳細請參照[strategymain()](#strategymain)

1. 休息函式：loop\_rate.sleep()

* 根據策略loop\_rate物件中所設定節點的更新速率，如不到1/30秒迴圈一次，則在此休息直到1/30秒

1. 回傳0：return 0

* 一般用在主函式結束時，按照程式開發的一般慣例，表示成功完成本函式

**Strategy\_main.cpp**

**\_****strategymain()**

1. 策略指撥：strategy\_info->getStrategyStart()
   * 當strategy\_info->getStrategyStart() = true：判斷初始旗標
   * 當strategy\_info->getStrategyStart() = false：判斷連續步態是否開啟
2. 連續步態是否開啟：walk\_con->isStartContinuous()
   * 用於判斷連續步態是否開啟之副函式
   * 當連續步態開啟時則回傳true，反之則回傳false
3. 關閉連續步態：walk\_con->stopContinuous()
   * 用於關閉連續步態之副函式
4. 讀檔旗標：BasketInfo->LoadFlag
   * 預設值為false
   * 當BasketInfo->LoadFlag = true：不執行任何指令
   * 當BasketInfo->LoadFlag = false：讀檔
5. 參數初始化：BasketInfo->Init()
   * 程式變數與參數初始化
6. 讀檔：Load->LoadParameters()
   * 讀檔之副函式，建立在LoadParameter類別中
   * 讀檔的內容有：
     + 29.ini：機器人站姿的參數
     + Throw.ini：籃球策略中的參數
     + Speed.ini：固定距離之投球力道數值
     + ContinuousMove.ini：連續步態參數
   * 詳細請參照[LoadParameters()](#LoadParameters)
7. 站立旗標：sendbodystandflag
   * 預設值為false
   * 當sendbodystandflag = true：不執行任何指令
   * 當sendbodystandflag = false：執行站立動作
8. 站立動作：ros\_com->sendBodySector(Robot\_StandUp)
   * Robot\_StandUp為站立動作的磁區(29)
   * ros\_com->sendBodySector(int Sector)：執行存放在Sector中的動作
9. 初始旗標：DIOSTRATAGAIN
   * 預設值為true
   * 當DIOSTRATAGAIN = true：不執行任何指令
   * 當DIOSTRATAGAIN = false：執行參數初始化、讀檔並執行找球初始動作
10. D1小指撥：strategyinfo->DIOValue.Switch.D1
    * strategyinfo->DIOValue有2種資料型態
      + unsigned char DInput
      + struct Switch：成員有D0, D1, D2, D3
    * strategyinfo->DIOValue.Switch.D1 = true：影像物件辨識並介面畫線
    * strategyinfo->DIOValue.Switch.D1 = false：不執行任何指令
11. 影像物件辨識：image()
    * 詳細請參照[image()](#image)
12. 介面畫線：Draw()
    * 詳細請參照[Draw()](#Draw)
13. 打印旗標：BasketInfo->PrintFlag
    * 預設值為false
    * 當BasketInfo->PrintFlag = true：不執行任何指令
    * 當BasketInfo->PrintFlag = false：打印BB幸運兔、BB佩佩豬
14. 找球初始動作：FindballInitial()
    * 詳細請參照[FindballInitial()](#FindballInitial)
15. 機器人狀態：BasketInfo->Robot\_State
    * 預設初始狀態為Initialization
    * 初始化(Initialization)
    * 找球(Find\_Ball)
    * 追蹤球(Trace\_Ball)
    * 走向球(Goto\_Ball)
    * 找籃框(Find\_Target)
    * 追蹤籃框(Trace\_Target)
    * 面向籃框(Goto\_Target)
    * 走向籃框(UP\_Basket)
    * 灌籃(SlamDunk\_Basket)
    * 結束(End)
16. 初始化：Initialization
    * 判斷D0小指撥是否開啟
    * 連續步態初始化
    * 執行補償動作
    * IMU值重製
    * 機器人狀態 = Find\_Ball
17. D0小指撥：strategyinfo->DIOValue.Switch.D0
    * strategyinfo->DIOValue.Switch.D0 = true：開啟上籃旗標
    * strategyinfo->DIOValue.Switch.D0 = false：不執行任何指令
18. 上籃旗標：BasketInfo->LayUpFlag
    * 預設值為false
    * 當BasketInfo->LayUpFlag = true：執行上籃策略
    * 當BasketInfo->LayUpFlag = false：執行投籃策略
19. 連續步態初始化：MoveContinuous()
    * 詳細請參照[MoveContinuous()](#MoveContinuous)
20. 補償動作: ros\_com->sendBodySector(BB\_StandFix)
    * BB\_StandFix為補償動作的磁區(1218)
21. 找球：
    * 詳細請參照[FindballHead()](#FindballHead)
22. 追蹤球：
    * 詳細請參照[TraceballHead()](#TraceballHead)
23. 走向球：
    * 詳細請參照[TraceballBody()](#TraceballBody)
24. 找籃框：
    * 詳細請參照[FindbasketHead()](#FindbasketHead)
25. 追蹤籃框：
    * 詳細請參照[TracebasketHead()](#TracebasketHead)
26. 面向籃框：
    * 詳細請參照[TracebasketBody()](#TracebasketBody)
27. 走向籃框：
    * 詳細請參照[UPbasket()](#UPbasket)
28. 灌籃
    * 詳細請參照[SlamDunk()](#SlamDunk)

**Strategy\_main.cpp**

**\_****Draw()**

此函式主要功能為介面畫線

函式宣告如右所示：ros\_com->drawImageFunction(int cnt, DrawMode mode,

int xmin, int xmax, int ymin, int ymax, int r, int g, int b)

* cnt：線條的編號
* mode為繪圖之模式：DrawLine = 1、DrawObject = 2
* xmin、xmax、ymin、ymax：線條的起點與終點

(xmin, ymin) to (xmax, ymax)

* r、g、b：線條的顏色

**Strategy\_main.cpp**

**\_****MoveHead()**

此函式主要功能為讓頭部馬達轉動

MoveHead是將ros\_com->sendHeadMotor(底層的api)包裝而成

1. 頭部馬達位置轉到輸入位置：

ros\_com->sendHeadMotor(HeadMotorID ID, int Position, int Speed)

* ID為頭部馬達的編號：HorizontalID = 1, VerticalID = 2
* Position：要移動到的位置
* Speed：移動的速度

1. 水平頭部馬達位置為輸入位置：

BasketInfo->HorizontalHeadPosition = Position

* 如水平頭部馬達轉動，紀錄馬達位置到水平頭部馬達位置，作為頭部馬達的假回授，垂直同理

**Strategy\_main.cpp**

**\_****MoveContinuous()**

此函式主要功能為設定連續步態參數

函式宣告如右所示：void MoveContinuous(int mode)

* mode為欲改變之參數，有
* ContinuousStand：初始化，只存初始化相關的參數
* ContinuousStay：原地踏步
* ContinuousTurnRight：原地右旋
* ContinuousSmallTurnRight：原地小右旋
* ContinuousTurnLeft：原地左旋
* ContinuousSmallTurnLeft：原地小左旋
* ContinuousForward：前進
* ContinuousSmallForward：小前進
* ContinuousSmallLeft：前進小左旋
* ContinuousBigLeft：前進大左旋
* ContinuousSmallRight：前進小右旋
* ContinuousBigRight：前進大右旋
* ContinuousFastForward：快速前進
* ContinuousFastSmallLeft：快速前進小左旋
* ContinuousFastBigLeft：快速前進大左旋
* ContinuousFastSmallRight：快速前進小右旋
* ContinuousFastBigRight：快速前進大右旋
* ContinuousBackward：後退
* ContinuousBackInit：後退初始化參數，在策略中未使用
* ContinuousMove.ini檔裡面的參數來調整x, y, z, theta, IMU, ChangeMode
* 當BasketInfo->Robot\_State為Initialization時，進行連續步態的初始化：
* walk\_con->setWalkParameterInit(float x\_init, float y\_init, float z\_init,

float theta\_init)：設定連續步態初始值

* walk\_con->setWalkParameterMax(float x\_max, float y\_max, float z\_max, float theta\_max)：設定連續步態上限，最大不會超過此數值
* walk\_con->setWalkParameterMin(float x\_min, float y\_min, float z\_min, float theta\_min)：設定連續步態下限，最小不會超過此數值
* walk\_con->setWalkParameterExp(float x\_exp, float y\_exp, float z\_exp, float theta\_exp)：設定欲達到之數值
* walk\_con->setWalkParameterOneAddValueAndPeriod(float x\_one, float y\_one, float z\_one, float theta\_one, int t)：設定每周期變化量
* 其餘情況則是作為改變連續步態的模式(執行前進、旋轉等等)

**Strategy\_main.cpp**

**\_****InversePerspective()**

此函式主要功能為利用逆透視法計算出目前機器人距離籃框的距離

1. 計算攝影機中心與機器人的夾角加上頭部誤差：

BasketInfo->MiddleAngle = atan(((double)BasketInfo->VisionMiddle / (double)BasketInfo->RobotSearchBasketHeight)) + BasketInfo->FeedBackError \* Deg2Rad

* BasketInfo->VisionMiddle：當籃框底部位於畫面中央時，目標物(籃框)與機器人的距離
* 目前預設的距離為84.5(公分)，可在Throw.ini中更改
* BasketInfo->RobotSearchBasketHeight：籃框底部到機器人視野的高度
* 目前預設的高度為16.95(公分)，可在Throw.ini中更改
* BasketInfo->FeedBackError：反饋誤差
* 目前預設的誤差角度為5.35(度)，可在Throw.ini中更改
* Deg2Rad：角度換成徑度的比例
* 定義的值為0.017453292(pi/180)(弳度/度)，可在basketballinfo.h中更改

1. 計算攝影機底部與機器人的夾角：

BasketInfo->BAngle = atan((double)BasketInfo->ScreenButtom / (double)BasketInfo->RobotSearchBasketHeight)

* BasketInfo->ScreenButtom：當籃框底部位於畫面底部時，目標物(籃框)與機器人的距離
* 目前預設的距離為32(公分)，可在Throw.ini中更改

1. 計算攝影機中心與攝影機底部的夾角：

BasketInfo->AAngle = BasketInfo->MiddleAngle - BasketInfo->BAngle

* 將攝影機中心與機器人的夾角減攝影機底部與機器人的夾角

1. 計算目標物與攝影機中心的夾角：

當籃框的Y像素最大值大於120時：

BasketInfo->dyAngle = atan2((double)((BasketInfo->Basket.YMax - 120) \* tan((double)BasketInfo->AAngle)), 120)

當籃框的Y像素最大值大於120時：

BasketInfo->dyAngle = atan2((double)((120 - BasketInfo->Basket.YMax) \* tan((double)BasketInfo->AAngle)), 120)

* BasketInfo->Basket.YMax：籃框的Y像素最大值，同時代表目標物
* 120為影像Y像素中間值

1. 計算目標物與機器人的夾角：

當籃框的Y像素最大值大於120時：

BasketInfo->BasketAngle = BasketInfo->MiddleAngle - BasketInfo->dyAngle

當籃框的Y像素最大值大於120時：

BasketInfo->BasketAngle = BasketInfo->MiddleAngle + BasketInfo->dyAngle

1. 計算與目標物距離：

BasketInfo->Distancenew = abs((double)BasketInfo->RobotSearchBasketHeight \* tan((double)BasketInfo->BasketAngle)) + BasketInfo->Error

* BasketInfo->Error：距離的誤差值
* 目前預設的誤差距離為0(公分)，可在Throw.ini中更改

**Strategy\_main.cpp**

**\_Triangulation()**

此函式主要功能為利用三角測量法計算出目前機器人距離籃框的距離

1. (籃框的Y像素值減影像的Y像素中間值)的絕對值大於0：

abs(BasketInfo->Basket.Y - 120) > 0

* 120為影像的Y像素中間值

1. 頭與機器人的垂直角度：BasketInfo->HeadVerticalAngle

* 藉由垂直頭部馬達位置、刻度換算成角度的比例、站立反饋和反饋誤差求出角度
* 刻度換算成角度的比例：Scale2Deg
* 定義的值為0.087890625(360/4096)(度/刻度)，可在basketballinfo.h中更改
* 站立反饋：BasketInfo->RobotStandFeedBack
* 目前預設的角度為0(度)，可在basketballinfo.cpp中更改
* 反饋誤差：BasketInfo->FeedBackError
  + 目前預設的角度為5.35(度)，可在Throw.ini中更改

1. 三角測量法的距離：BasketInfo->Distancenew

* 藉由機器人高度、攝影機高度、頭與機器人的垂直角度、角度換成徑度的比例和三角測量距離誤差求出距離
* 機器人高度：BasketInfo->RobotHeight
* 目前預設的高度為0(公分)，可在basketballinfo.cpp中更改，在loadparameter.cpp中計算
* 攝影機高度：CameraHeight
* 定義的高度為3.925(公分)，可在basketballinfo.h中更改
* 角度換成徑度的比例：Deg2Rad
* 定義的值為0.017453292(pi/180)(弳度/度)，可在basketballinfo.h中更改
* 三角測量距離誤差：BasketInfo->DistanceError
* 目前預設的距離為0(公分)，可在basketballinfo.cpp中更改

**Strategy\_main.cpp**

**\_****image()**

此函式主要功能為獲取物件(球、籃框)資訊

物件的資訊有：

* Size：影像大小(pixel)
* X、XMin、XMax：物件的像素值在X軸上的中點、起點、末點
* Y、YMin、YMax：物件的像素值在Y軸上的中點、起點、末點

**Strategy\_main.cpp**

**\_****AreaSizeMeasure()**

此函式主要功能為利用面積法計算出目前機器人距離籃框的距離。

前邊界：front

* 目前機器人所在區間最小值

後邊界：back

* 目前機器人所在區間最大值

面積法計算距離：BasketInfo->Distancenew

* 利用前邊界及後邊界的面積與距離計算出目前機器人與籃框的距離

**Strategy\_main.cpp**

**\_****ComputeSpeed()**

此函式主要功能為計算投球速度

藉由計算出的距離(在此以*D*表示，策略的變數為BasketInfo->Distancenew)去做權重運算，參考以下公式：

* 
* 
* 
* 
* 
* 
* 

固定距離力道*S*可在Speed.ini中做更改，將其力道與各權重去做相乘並再將所有結果相加可得出投籃所需的力道，公式如下：



**Strategy\_main.cpp**

**\_SelectBaseLine()**

此函式主要功能為選出籃框垂直基準線

* 先判斷距離
* 決定籃框垂直基準線
* 再依照機器人的位置
* 決定垂直基準線是否需要做調整

**Strategy\_main.cpp**

**\_****FindballInitial()**

此函式主要功能為預設機器人的頭部馬達在開始找球時的初始位置

* 垂直頭部馬達位置
* 目前預設的位置為1623(刻度)，可在strategy\_main.cpp中更改
* 水平頭部馬達位置
* 目前預設的位置為2651(刻度)，可在strategy\_main.cpp中更改

**Strategy\_main.cpp**

**\_****FindballHead()**

1. 球的面積大於球最遠的面積：BasketInfo->Ball.size > Ballfarsize

* 球最遠的面積所定義的值為500(像素值)，可在basketballinfo.h中更改

1. 垂直頭部狀態：BasketInfo->HeadVerticalState

* 頭部向上轉動(HeadTurnNear)：抬頭動作MoveHead(HeadMotorID::VerticalID, 1575 ,200)
* 頭部向下轉動(HeadTurnClose)：低頭動作MoveHead(HeadMotorID::VerticalID, 1100 ,200)
* 預設初始狀態為HeadTurnNear，預設執行抬頭動作

1. 水平轉頭狀態：BasketInfo->HeadHorizontalState

* 預設初始狀態為HeadTurnRight，預設執行向右方轉頭動作
* 頭部向右轉動(HeadTurnRight)：向右方轉頭動作
* 頭部向左轉動(HeadTurnLeft)：向左方轉頭動作

1. (水平頭部馬達位置減轉動刻度)大於最小水平位置：

(BasketInfo->HorizontalHeadPosition - BasketInfo->HeadTurnSpeed) > BasketInfo->HorizontalMinAngle

* 是：向右轉頭

MoveHead(HeadMotorID::HorizontalID,

BasketInfo->HorizontalHeadPosition - BasketInfo->HeadTurnSpeed, 200)

* + 水平頭部馬達位置-轉動刻度=向右旋轉的目標位置
  + 執行動作：向右旋轉一次轉動刻度BasketInfo->HeadTurnSpeed
* 否：水平轉頭狀態= HeadTurnLeft

1. (水平頭部馬達位置加轉動刻度)小於最大水平位置：

(BasketInfo->HorizontalHeadPosition + BasketInfo->HeadTurnSpeed) < BasketInfo->HorizontalMaxAngle

* 是：向左轉頭

MoveHead(HeadMotorID::HorizontalID,

BasketInfo->HorizontalHeadPosition + BasketInfo->HeadTurnSpeed, 200)

* + 水平頭部馬達位置+轉動刻度=向左旋轉的目標位置
  + 執行動作：向左旋轉一次轉動刻度BasketInfo->HeadTurnSpeed
* 否：水平頭部狀態= HeadTurnRight

**Strategy\_main.cpp**

**\_TraceballHead()**

1. 計算球的X像素值與球垂直基準線距離：

BasketInfo->Ball.X - BasketInfo->BallVerticalBaseLine

* + 計算球與球垂直基準線的X像素值誤差

1. 計算球的Y像素值與球水平基準線距離：

BasketInfo->Ball.Y - BasketInfo->BallHortionalBaseLine

* + 計算球與球水平基準線的Y像素值誤差

1. 計算水平誤差角度：BasketInfo->ErrorHorizontalAngle
   * 根據球與球垂直基準線(預設為190的X像素值)的水平誤差，計算出水平頭部馬達需轉動的角度
2. 計算垂直誤差角度：BasketInfo->ErrorVerticalAngle
   * 根據球與球水平基準線(預設為120的Y像素值)的垂直誤差，計算出垂直頭部馬達需轉動的角度
3. 根據水平誤差角度還有位置與角度的換算來讓頭水平旋轉：

MoveHead(HeadMotorID::HorizontalID, BasketInfo->HorizontalHeadPosition – (BasketInfo->ErrorHorizontalAngle \* TraceDegreePercent \* 1 \* Deg2Scale), 200)

* + 每次轉多少的角度百分比：TraceDegreePercent
* 依據執行策略結果決定要分幾次轉到目標位置
  + 角度換算成刻度的比例：Deg2Scale
* 定義的值為11.377777777 (4096/360)(刻度/度)，可在basketballinfo.h中更改
  + 垂直同理

1. 球是否在畫面基準點(中心)附近：
   * 球減掉球垂直基準線的X誤差小於等於6像素值
   * 球的Y減掉球水平基準線的誤差小於等於14像素值
2. 直接夾球旗標：BasketInfo->StraightCatchFlag
   * 預設為true
   * BasketInfo->StraightCatchFlag = true：判斷水平頭部馬達位置是否大致在中心且是否關閉步態，來決定能否直接夾球
   * BasketInfo->StraightCatchFlag = false：不執行任何指令
3. 預先旋轉旗標：BasketInfo->PreRotateFlag
   * 預設為false
   * BasketInfo->PreRotateFlag = true：判斷機器人是否需要旋轉調整，再利用垂直頭部馬達位置判斷連續步態減速線是否需要增加
   * BasketInfo->PreRotateFlag = false：不執行任何指令
4. 連續步態旗標：BasketInfo->ContinuousFlag
   * 詳細請參照[TraceballBody()](#TraceballBody)
5. 彎腰旗標：BasketInfo->StoopFlag
   * 詳細請參照[TraceballBody()](#TraceballBody)
6. 連續步態減速線：BasketInfo->ContinuousSlowLine
   * 詳細請參照[TraceballBody()](#TraceballBody)
7. 重新找球旗標：BasketInfo->ReStartFindBallFlag
   * 預設值為true
   * BasketInfo->ReStartFindBallFlag = true：利用垂直頭部馬達位置判斷是否要重新找球
   * BasketInfo->ReStartFindBallFlag = false：機器人狀態= Goto\_Ball
8. 夾球線：BasketInfo->CatchBallLine
   * 詳細請參照[TraceballBody()](#TraceballBody)
9. 水平最小位置：BasketInfo-> HorizontalMinAngle
   * 目前預設的位置為1302(刻度)，可在Throw.ini中更改
10. 水平最大位置：BasketInfo-> HorizontalMaxAngle
    * 目前預設的位置為2794(刻度)，可在Throw.ini中更改
11. 垂直最小位置：BasketInfo->VerticalHeadMinAngle
    * 目前預設的位置為1050(刻度)，可在Throw.ini中更改
12. 垂直最大位置：BasketInfo->VerticalHeadMaxAngle
    * 目前預設的位置為2000(刻度)，可在Throw.ini中更改

**Strategy\_main.cpp**

**\_****TraceballBody()**

1. 連續步態旗標：BasketInfo->ContinuousFlag
   * 預設值為true
   * BasketInfo-> ContinuousFlag = true：判斷球的面積是否小於等於球最遠的面積
   * BasketInfo-> ContinuousFlag = false：判斷彎腰旗標
2. 連續步態減速線：BasketInfo->ContinuousSlowLine
   * 作為切換連續步態前進速度的界線
   * 目前預設的位置為1400(刻度)，可在Throw.ini中更改
3. 夾球線：BasketInfo->CatchBallLine
   * 作為是否要停止找球並執行彎腰動作的界線
   * 目前預設的位置為1130(刻度)，可在Throw.ini中更改
4. 彎腰旗標：BasketInfo->StoopFlag
   * 預設值為false
   * BasketInfo->StoopFlag = true：判斷球的面積是否小於等於球最遠的面積
   * BasketInfo->StoopFlag = false：判斷移手旗標
5. 球是否在畫面基準點附近：

abs(BasketInfo->Ball.X - BasketInfo->BallVerticalBaseLine)

<= BasketInfo->BallVerticalError &&

abs(BasketInfo->Ball.Y - BasketInfo->BallHorizontalBaseLine)

<= BasketInfo->BallHorizontalError

* + 球與球垂直基準線的X像素值誤差小於等於球的垂直誤差值
  + BasketInfo-> BallVerticalError：球的垂直誤差值
    - 目前預設的誤差值為6(像素值)，可在Throw.ini中更改
  + 球與球水平基準線的Y像素值誤差小於等於球的水平誤差值
  + BasketInfo->BallHorizontalError：球的水平誤差值
    - 目前預設的誤差值為14(像素值)，可在Throw.ini中更改

1. 夾球垂直頭部馬達位置：BasketInfo->CatchBallVerticalHeadPosition
   * 當機器人為彎腰時，視線方向需要與地面垂直
   * 目前預設的位置為1432(刻度)，可在Throw.ini中更改
2. 彎腰動作：ros\_com->sendBodySector(BB\_WaistDown)
   * BB\_WaistDown為彎腰動作的磁區(118)
3. 移手旗標：BasketInfo->MoveFlag
   * 預設值為false
   * BasketInfo->MoveFlag = true：判斷移手方向與刻度並移手
   * BasketInfo->MoveFlag = false：判斷拿球旗標
4. 計算相差的Y像素值：BasketInfo->count
   * 球與夾球基準線的Y像素值誤差
   * BasketInfo->CatchBallYLine：夾球基準線
     + 目前預設的位置為195(刻度)，可在Throw.ini中做更改
5. 計算手需移動的刻度：BasketInfo->HandMove
   * 相差的Y像素值乘上像素值換算成刻度的比例
   * 像素值換算成刻度的比例：
     + 根據Y像素值誤差與實際要移動多少刻度決定像素值換算成刻度的比例
     + 目前預設像素值換算成刻度的比例為2
6. 向外回正旗標：BasketInfo->OutReturnFlag
   * 初始值為false
   * BasketInfo->OutReturnFlag = true：雙手向外回正
   * BasketInfo->OutReturnFlag = false：判斷上修旗標
7. 向內回正旗標：BasketInfo->InReturnFlag
   * 初始值為false
   * BasketInfo->InReturnFlag = true：雙手向內回正
   * BasketInfo->InReturnFlag = false：不執行任何指令
8. 拿球旗標：BasketInfo->GetBallFlag
   * 預設值為false
   * BasketInfo->GetBallFlag = true：執行夾球動作
   * BasketInfo->GetBallFlag = false：判斷轉向旗標
9. 夾球動作：ros\_com->sendBodySector(BB\_WaistCatch)
   * BB\_WaistCatch為夾球動作的磁區(1110)
10. 挺腰動作：ros\_com->sendBodySector(BB\_WaistUp)
    * BB\_WaistUp為挺腰動作的磁區(1111)
11. 挺腰回復動作：ros\_com->sendBodySector(BB\_WaistUpFeedBack)
    * BB\_WaistUpFeedBack為挺腰回復動作的磁區(1112)
12. 轉向旗標：BasketInfo->TurnFlag
    * 預設值為false
    * BasketInfo->TurnFlag = true：機器人面向籃框
    * BasketInfo->TurnFlag = false：不執行任何指令
13. IMU的Yaw軸值：strategy\_info->getIMUValue().Yaw
    * 正前方數值為0，左旋數值遞增，右旋數值遞減
14. 面框旗標：BasketInfo->FaceBasketFlag
    * 預設值為false
    * BasketInfo->FaceBasketFlag = true：改變機器人位置的旗標，來改變旋轉方向
    * BasketInfo->FaceBasketFlag = false：不執行任何指令

**Strategy\_main.cpp**

**\_****FindbasketHead()**

跟[FindballHead()](#FindballHead)相同概念

1. 籃框面積大於籃框最遠的面積：BasketInfo->Basket.size > Basketfarsize
   * 籃框最遠的面積所定義的值為1200(像素值)，可在basketballinfo.h中更改
   * 是：機器人狀態= Trace\_Target
   * 否：找籃框

**Strategy\_main.cpp**

**\_****TracebasketHead()**

1. 計算籃框的X像素值與影像的X像素中間值的差：

BasketInfo->BasketMoveX = BasketInfo->Basket.X - 160

* + 160為影像的X像素中間值

1. 計算籃框的Y像素值與影像的Y像素中間值的差：

BasketInfo->BasketMoveY = BasketInfo->Basket.Y - 120

* + 120為影像的Y像素中間值

1. 計算籃框的X像素值與籃框垂直基準線距離：

BasketInfo->BasketMoveX = (BasketInfo->Basket.X -

BasketInfo->BasketVerticalBaseLine)

* + 算籃框與垂直基準線的X像素值誤差
  + BasketInfo->BasketVerticalBaseLine：投籃時籃框垂直基準線，目前預設的位置為200(刻度)，可在Throw.ini中更改

1. 計算籃框的Y像素值與籃框水平基準線距離：

BasketInfo->BasketMoveY = (BasketInfo->Basket.Y -

BasketInfo->BasketHorizontalBaseLine)

* + 算籃框與水平基準線的Y像素值誤差
  + BasketInfo-> BasketHorizontalBaseLine：投籃時籃框水平基準線，目前預設的位置為120(刻度)，可在Throw.ini中更改

1. 籃框的面積：BasketInfo->SizeOfDist
   * 代表機器人在各個距離下，所接收到的像素值大小，可在Throw.ini中做輸入數值：
     + BasketInfo->SizeOfDist[0]：距離50公分時的籃框的像素值大小
     + BasketInfo->SizeOfDist[1]：距離60公分時的籃框的像素值大小
     + BasketInfo->SizeOfDist[2]：距離70公分時的籃框的像素值大小
     + BasketInfo->SizeOfDist[3]：距離80公分時的籃框的像素值大小

**Strategy\_main.cpp**

**\_****TracebasketBody()**

1. 選擇基準線：SelectBaseLine()
   * 根據不同的測距，會有不同的BasketInfo->BasketVerticalBaseLine
   * 詳細請參照[SelectBaseLine()](#SelectBaseLine)
2. 旋轉旗標：BasketInfo->RoateFlag
   * 預設值為true
   * 當BasketInfo->RoateFlag = ture：判斷水平頭部馬達位置是否大致在中心
   * 當BasketInfo->RoateFlag = false：判斷轉腰旗標
3. 轉腰旗標：BasketInfo-> WaistFlag
   * 預設值為false
   * 當BasketInfo->WaistFlag = ture：執行轉腰
   * 當BasketInfo->WaistFlag = false：判斷計算旗標
4. 三角測量：Triangulation()
   * 詳細請參照[Triangulation()](#Triangulation)
5. (籃框的X像素值減籃框垂直基準線)的絕對值小於等於轉腰誤差：

abs(BasketInfo->Basket.X - BasketInfo->BasketVerticalBaseLine) <=

BasketInfo->WaistError

* + 籃框的X像素值：BasketInfo->Basket.X
    - 籃框的中心點X座標
  + 籃框垂直基準線：BasketInfo->BasketVerticalBaseLine
    - 投球時籃框垂直基準線
  + 轉腰誤差：BasketInfo->WaistError
    - 籃框的中心點與基準線之誤差，目前預設的誤差值為2(刻度)，可在Throw.ini中更改

1. 計算旗標：BasketInfo->ComputeFlag
   * 預設值為false
   * 當BasketInfo->ComputeFlag = ture：計算投球速度、判斷重新對準旗標並將投球速度送到投球動作的磁區
   * 當BasketInfo->ComputeFlag = false：判斷舉手旗標
2. 計算投球速度：ComputeSpeed()
   * 詳細請參照[ComputeSpeed()](#ComputeSpeed)
3. 重新對準旗標：BasketInfo->ReAimFlag
   * 預設值為false
   * BasketInfo->ReAimFlag = ture：

重新對準BasketInfo->BasketVerticalBaseLine

* + 當BasketInfo->ReAimFlag = false：不執行任何指令

1. 將投球速度送到投球動作的磁區：

ros\_com->sendHandSpeed(BB\_ShootingBall, BasketInfo->disspeed)

* + BB\_ShootingBall為投球動作的磁區(1116)
  + BasketInfo->disspeed為ComputeSpeed()算出的投球速度

1. 舉手旗標：BasketInfo->RaiseFlag
   * 預設值為false
   * 當RaiseFlag = ture：執行投籃舉手動作
   * 當RaiseFlag = false：判斷投球旗標
2. 投籃舉手動作：ros\_com->sendBodySector(BB\_RaiseHand)
   * BB\_RaiseHand為投籃投球動作的磁區(1115)
3. 投球旗標：BasketInfo->ThrowBallFlag
   * 預設值為false
   * 當ThrowBallFlag = ture：執行投球動作
   * 當ThrowBallFlag = false：不執行任何指令

**Strategy\_main.cpp**

**\_****UPbasket()**

1. 上籃停止線：BasketInfo->UpBasketStopLine
   * 目前預設的位置為1750(刻度)，可在Throw.ini內更改
2. 舉左手旗標：BasketInfo->LeftHandUpFlag
   * 預設值為false
   * 當BasketInfo->LeftHandUpFlag = true：執行舉左手
   * 當BasketInfo->LeftHandUpFlag = false：不執行任何指令

**Strategy\_main.cpp**

**\_****SlamDunk()**

1. 上籃舉手旗標：BasketInfo->HandUpFlag
   * 預設值為true
   * 當BasketInfo->HandUpFlag = true：執行上籃舉手動作、執行轉腰修正
   * 當BasketInfo->HandUpPFlag = false：判斷灌籃旗標
2. 轉動頭部馬達對準籃框垂直中心線：
   * 籃框垂直中心線
     + 目前預設的像素值為160(刻度)，代表影像的X像素中間值
   * 計算水平誤差角度：BasketInfo->ErrorHorizontalAngle
     + 根據籃框的X像素值跟影像的X像素中間值的像素差，計算出頭部水平馬達需轉動的角度
   * 根據水平誤差角度還有位置與角度的換算來讓頭水平旋轉：

MoveHead(HeadMotorID::HorizontalID,

BasketInfo->HorizontalHeadPosition – (BasketInfo->ErrorHorizontalAngle \* Deg2Scale), 200)

1. 上籃舉手動作：ros\_com->sendBodySector(BB\_UpHand)
   * BB\_UpHand為上籃舉手動作的磁區(1140)
2. 灌籃旗標：BasketInfo->SlamDunkFlag
   * 預設值為false
   * BasketInfo->SlamDunkFlag = true：執行灌籃動作
   * BasketInfo->SlamDunkFlag = false：不執行任何指令
3. 根據上籃誤差值執行腰部修正：

BasketInfo->HorizontalHeadPosition - BasketInfo->SlamDunkHorizontalAngle

* + 水平頭部馬達位置減水平灌籃角度
  + 水平灌籃角度：BasketInfo->SlamDunkHorizontalAngle
    - 目前預設的位置為1810(刻度)，可在Throw.ini中更改

1. 灌籃動作：ros\_com->sendBodySector(BB\_SlamDunk)
   * BB\_SlamDunk為灌籃動作的磁區(1150)

**Loadparameter.cpp**

**\_****initparameterpath()**

此函式主要功能為讀取參數資料夾的路徑

1. 讀取參數資料夾的絕對路徑：

parameter\_path = tool->getPackagePath("strategy")

* + 讀取策略資料夾的絕對路徑加上參數資料夾的相對路徑

**Loadparameter.cpp**

**\_****LoadParameters()**

此函式主要功能為讀取參數

29.ini存有機器人站立姿勢

* 讀取參數後將刻度轉換成角度，計算出機器人的站立身高

Throw.ini內較常用到的參數：

* RobotSearchBasketHeight、VisionMiddle、ScreenButtom、Error：為逆透視法參數
* FeedBackError、DistanceErrorCount：為三角測量的參數
* BasketVerticalBaseLine等等：為瞄準籃框之基準線，根據不同距離調整
* SizeOfDist：不同距離之籃框的面積
* ContinuousSlowLine：連續步態減速的位置(移動到球所在的位置)
* CatchBallLine：夾球的位置
* CatchBallVerticalHeadPosition、CatchBallYLine：夾球修正的參數
* WaistError：投籃誤差容忍值
* UpBasketStopLine：上籃的位置
* SlamDunkHorizontalAngle：上籃轉腰修正的參數

Speed.ini存有投球力道的參數

* 基本上只會動到disXXspeed，XX表示距離(公分)

ContinuousMove.ini存有連續步態的參數

* ContinuousStep矩陣之宣告在basketballinfo.h中
* ContinuousStep為一union，內有ContinuousStand與ContinuousMove
* AddPeriod：數值變化的週期
* ContinuousStand存放連續步態之初始值
* InitX, InitY, InitTheta：前進、平移、旋轉量
* Mode：連續步態的模式
* IMUSet：IMU的設定
* ContinuousMove存放連續步態移動之參數(前進、左右平移等等)
* ExpX, ExpY, ExpZ, ExpTheta：要達到的數值
* AddX, AddY, AddZ, AddTheta：每周期變化量

**Walkparameter**

**\_****setWalkParameterInit()**

1. 連續旗標：continuous\_flag
   * 用以確認是否正在執行連續步態
2. 連續步態初始值旗標：setWalkParameterInit\_flag
   * 用以確認是否已執行初始化

**WalkContinuouse**

**\_****setWalkParameterMax()**

1. 連續步態最大值旗標：setWalkParameterMax\_flag
   * 用以確認是否已執行設定最大值

**WalkContinuouse**

**\_****setWalkParameterMin()**

1. 連續步態最小值旗標：setWalkParameterMin\_flag
   * 用以確認是否已執行設定最小值

**WalkContinuouse**

**\_****setWalkParameterExp()**

1. 連續步態期望值旗標：setWalkParameterExp\_flag
   * 用以確認是否已執行設定期望值
2. 確認資料值函式：checkDataValue()
   * 詳細請參照[checkDataValue()](#checkDataValue)

**WalkContinuouse**

**\_****startContinuous()**

1. 確認所有旗標函式：checkAllFlag()
   * 用來檢查是否有設定初始值、最大值、最小值、期望值、累加值，為保護機制
2. 計時函式開始：start ()
   * 啟動ROS內建的計時器，若已啟動，則不做任何事

**WalkContinuouse**

**\_****stopContinuous()**

1. 計時函式停止：stop ()
   * 暫停ROS內建的計時器，若已暫停，則不做任何事

**WalkContinuouse**

**\_****checkDataValue()**

1. test
   * 當有任何不合法的數值出現，如最大值小於最小值等，test為true，此函式最後會還傳false，表示有數值有誤