文件编号	YX-YF-M0007
密　级	秘 密

飞控端调试光流方法说明

产 品 名 称：光流模块

产 品 类 别：技术说明

产 品 代 号：UP-FLOW-LC-002

编　制　人：曾小铃

编 制 日 期：2017.03.09

审核人：	审核日期：
批准人：	批准日期：

upixels 优象®

修订记录

序号	修订内容	修订人	修订日期	版本	更改编号
1	初稿		2018.03.09	V1.0	

目　录

一、 光流在飞控端的使用(上接 UP-FLOW 光流模块使用手册-V1.3.1)

飞控端正确读取光流的数据后，接下来就是如何实现对于飞行的控制，即估计飞机真实的水平运动速度。以光流悬停为例，光流是如何抑制飞机悬停时的漂移的呢？一般来说，飞机在无光流时产生随机漂移的原因是姿态角度的误差造成的，使得飞机在自身有倾角时而错误的认为自身是水平而导致的，由于多旋翼是欠驱动系统，有倾角从而产生该方向上的漂移速度，因此光流就是估计这个误差角度或漂移速度，并反馈给控制系统进行控制(一般是串级的 PID 控制)，控制输出的结果一般是飞机的倾斜角度，即期望角度。

在实际使用中，用光流实现悬停时主要有两部分的工作，一是光流数据的处理，可以是光流的自身滤波，或与加速度计进行融合滤波，获取飞机的位移与速度信息；二是用处理后获得的估计位置与速度数据进行 PID 控制，控制飞机位置与速度调整，实现悬停。其基本流程如图 1 所示。

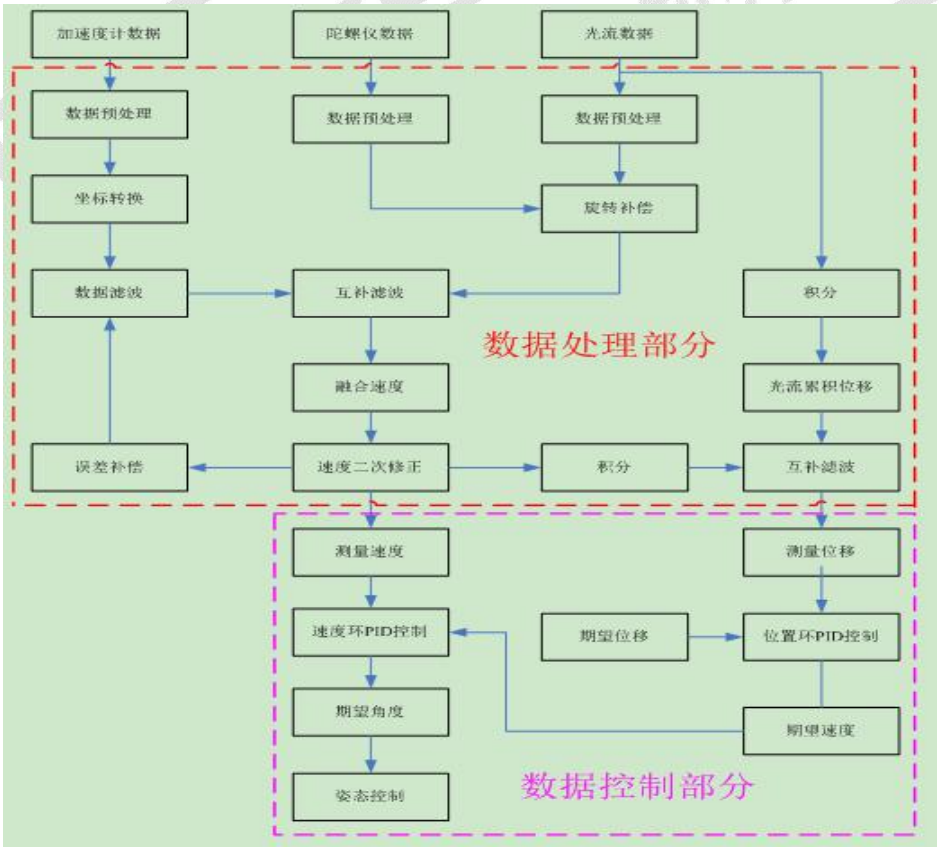

图 1 光流数据处理与控制流程图

二、 光流数据飞控端的处理

光流数据在飞控端的处理可以分为两种情况，一是只使用光流纯数据；二是将光流数据与加速度计进行融合。

只使用纯光流数据进行控制，平滑性与抗环境干扰性会差一些，但好处是处理简单，需要调试的参数少，且在悬停时与进行加速度融合控制效果也差不了多少，建议初次使用时先采用该方式，先跑通整个控制框架；将光流与 IMU 融合，如果 IMU 加速度计数据处理得好，数据各个方面的性能会好很多，如果处理不好或加速度计输出数据比较差反而会适得其反，且处理相对比较复杂，细节较多，调试更困难，因此在单纯用光流效果不理想，且调试一段时间后还没有改善，建议采用本方式。

1. 只使用光流纯数据的处理方法

给出的光流数据经解码与处理后(参考<up_flow.c>)得到的 x、y 方向的光流 flow_dat.x，flow_dat.y。因此光流纯数据的处理也是基于该两个变量。

1.1 速度计算方法：

speed_x = (h*flow_dat.x/ (integration_timespan*0.000001))*100

speed_y = (h*flow_dat.y/(integration_timespan*0.000001))*100

其中 h 为测量高度，单位 m；integration_timespan 为相邻两个光流输出的间隔时间,单位 us，100 是 m->cm 的转换。

1.2 位移计算方法：

sum_flow_x += speed_x*(integration_timespan*0.000001);

sum_flow_y += speed_y*(integration_timespan*0.000001);

该位移默认开始悬停点为零，计算的是后续累积时间内，飞机距离该初始位置点的水平位移，单位为 cm。

1.3 备注：

1.3.1 由于计算的速度帧率不是很高,且帧间的速度信息可能差别比较

大，因此可以根据控制需要，选择对计算的速度进行低通滤波，但不宜引入过大的延时。

1.3.2 理论上在速度计算之前需要进行旋转补偿，但在初始测试时，可以暂时先不考虑，先跑通整个框架。如果悬停效果可以接受也不用进行旋转补偿；若悬停时低频震荡且 PID 参数怎么也调试不好，就可能需要进行补偿，具体的旋转补偿方法在 2.2 节中再做详细叙述。

1.3.3 光流位移量的处理有些细节需要注意，在某些情况下需要清除该累积位移与控制积分量。情况一，打摇杆后需要清零，否则松摇杆后飞机还会往前进的反方向后退一段距离；情况二，起飞前或正在起飞时需要清零，否则飞机会斜着起飞，即起飞上升时飞机往一边严重漂移。情况三，悬停时，当累积距离大于一定范围。

1.3.4 在只使用气压计进行定高而没有超声波时，由于气压计会有很大漂移，因此输出的高度要做限制。首先限制高度要非负，另外在较低高度时可以使用固定高度，我们在测试时一般将高度写死固定为 1m，实际测试发现在实际高度 5m 以下，使用该写死的高度也能有较好的悬停效果；当实际高度比较高时，可以适当放大该固定高度比例。

2. 光流与加速度计融合的处理方法

一般单独使用光流的数据也能够获取较好的静态悬停效果，如果对光流的动态性能也有较高要求，比如实现更好的抗风、位移控制、目标跟随等高级功能，则需要进行光流与加速度计数据的融合以获取更加精确、平滑与实时的数据。具体实现可以参考<up_flow_fusion_demo.c>，下面对该文件进行简单梳理，并给出计算过程。

第一步：确定使用高度。在只使用气压计高度时，高度不是很准(原因参考 2.1.3 备注中第四点)，因此根据气压计的输出值，对使用的高度进行了分段处理，实际调试中可以根据飞机不同高度是否震荡进行调整，使飞机不震荡即可。

第二步：光流角速度的计算。在 2.1.2 中 speed_x 为线速度，若 speed_x 不乘高度即为光流角速度，表示为 fx = flow_dat.x/ (integration_timespan*0.000001))，同理可求 fy，单位为 rad/s。

第三步：光流旋转补偿。该步骤是提升性能核心，否则不能够支持比较大的 PID 控制参数，表现为加大控制参数飞机震荡。基本思想是使用陀螺仪角速度对光流进行旋转补偿，目的是使得飞机在只有旋转没有平移时光流最终输出为零。

补偿公式为：

fx_gyro_fix = ((fx + LIMIT(((gyro_lpf_y)/57.3f),-flow_t1,flow_t1)) *10 *use_height) ; //rotation compensation

fy_gyro_fix = ((fy - LIMIT(((gyro_lpf_x)/57.3f),-flow_t2,flow_t2)) *10 *use_height) ;

以 X 方向的补偿为例，其中，fx_gyro_fix 为旋转补偿后的输出，单位是 mm/s;fx 为光流角速度；(10 *use_height)为使用的高度值，单位为 mm。

(gyro_lpf_y)/57.3 为陀螺仪输出角速度，单位 rad/s，flow_t1 为对陀螺仪输出的限幅，值的大小由 fx 的输出最大值决定，因为陀螺仪的输出是远远大于光流的输出范围的，限幅避免过度补偿。gyro_lpf_y 为原始的陀螺仪角速度(度/秒)，经低通滤波后所得：LPF_1_(3.0f,dT,gyro_x,gyro_lpf_x)，目的是使得光流输出的相位与陀螺仪相位一致。

第四步：求水平航向坐标系下加速度值 a(n)(该坐标系与俯仰和滚转角为零时的机体坐标系重合)。由姿态角可以获取姿态旋转矩阵，该姿态旋转矩阵在 yaw=0 时得到的旋转矩阵即为所求 R，再使用 a(n) = R*a(b)求得目标向量值，包括 xyz 三个方向的加速度值，其中 a(b)为机体加速度计测量向量，即传感器原始输出值，单位是 mm/(s*s)。

第五步：加速度积分获取速度，并与光流进行互补滤波,获取初步的融合速度 f1_fx.out。

第六步：融合速度修正，由于加速度计积分有漂移，因此可能造成融合的速度在悬停或运动中会漂移真实值，而光流的计算是没有漂移及累积误差的，因此以光流数据做参考，对 f1_fx.out 做一个二次修正，得到最终融合速度输出 f_out_x。

备注：

本算法中只对融合速度进行了处理，而没有融合高度的处理。因为在测试中发现使用融合速度积分得到位移精度反而不如单纯的光流累积计算精度高，因此建议位移的获取仍采用 2.1.2 中位移的获取方法。

三、基于光流数据的控制方法

由光流获取飞机水平漂移的速度与位移后，即可以根据这些信息进行控制。以悬停为例，则期望的位移为零，反馈的即光流的测量值，一般采用的是串级的 PID 控制，即位置环+速度环控制，输出为期望角度，给姿态控制环。具体可以参考<up_flow.c>中的控制部分。由于该部分比较简单，在这里只对其中一些需要注意的细节进行描述。

3.1. 对于位置环，期望位置为零，反馈值为光流测量位移，输出为期望速度。

3.2. 对于速度环，期望值为位置环的输出，反馈值为光流测量速度，速度环输出为期望角度或期望加速度。理论上，输出应为期望加速度，并通过非线性转换为期望角度，从而传递给姿态环。在 demo 中我们的输出直接是期望角度，这是因为在小角度时，我们认为期望加速度与角度是线性相关的。

3.3 定点控制中比较重要的一点，要确保外环的执行频率要低于内环的执行频率，否则会造成系统的不稳定。比如，我们位置环的控制周期为 40ms,则速度环的控制周期为 20ms 等等。

3.4 控制量在某些情况下需要清零，特别是控制积分量。比如打摇杆、起飞时或一键翻滚时。

四、光流使用中需要注意的细节

4.1 光流模块的检查。拿到光流芯片后，首先需要检查镜头表面是否有塑料薄膜、水珠、划痕等，确保镜头表面干净，器件或连线无松动等影响镜头成像效果的问题。

4.2 光流模块的安装。在飞机上安装光流模块时，尽量保证镜头的水平，不要有倾斜角度，光流模块与机体坐标系不要有偏航方向的夹角且在机架的旋转中心，即光流模块的轴与机体坐标系的轴应尽量保持重合。特别注意一旦调试通过，进入批量生产时，在其他飞机上都需要以该方向固定安装，不可随意改变安装方向。

4.3 初始调试时，接上光流后，使用上位机观察是否有输出，验证输出值是否与图 2.所述的光流坐标系定义一致，比如往左水平运动光流 Y 是否为正；水平向右运动光流 Y 是否为负，且输出值是否在合理范围等。

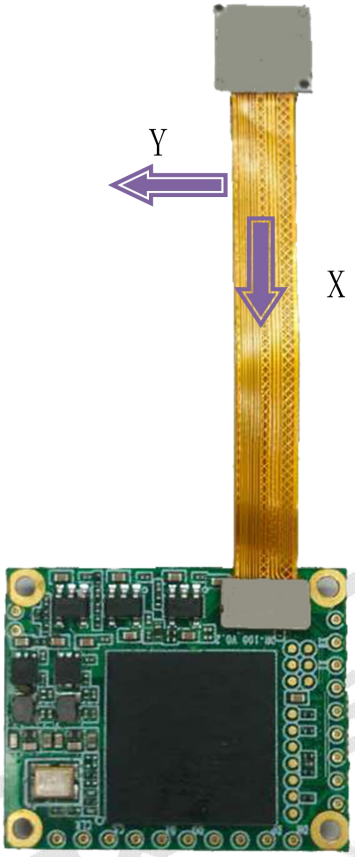

图 2 光流坐标系

4.4.确保光流的输出与机体控制坐标系的一致性，即实飞测试前需要验证光流的控制输出值与期望给飞机的控制指令是否一致(手拿着飞机水平移动，看输出是否符合预期)，避免造成控制指令的不匹配导致的飞机乱飞现象。

4.5.初始调试时应尽量选择光照明亮，地面纹理丰富且不反光，无风的环境进行，避免引入环境的干扰因素。

五、悬停问题速查

5.1 完全没有悬停效果，现象与没接光流是一样的。首先应检查板子上电是

否正常，硬件线是否连接好，镜头是否被遮挡，如果都没发现问题，再用上位机检查是否有数据输出。

5.2 飞机往某个方向乱飞，漂移更快，像失控的感觉，而没有加光流时反而漂移慢一些。可能是：

5.2.1 镜头安装方向与程序里写的方向不一致；

5.2.2 环境光线明暗变化比较大，比如有闪光灯；

5.2.3 光流镜头下地面上有大片的运动物体，比如在有风时的水面上；

5.2.4 程序中光流控制部分有些数据没清零，比如起飞时与打摇杆后积分量没清零；

5.3.飞机缓慢飘，方向随机，但又比没光流时要好，表现为无操作时，往一个方向飘后，还会往回飘。可能是地面环境纹理太差，比如纯净的木地板、水泥地面、反光大理石地面等，或者是环境光线太暗。

5.4 飞机开始时能够正常悬停，悬停一段时间后开始震荡，可能是：

5.4.1 气压计数据漂移很大，高度严重失真，导致光流数据与该高度相乘后输出值变大，控制超调；

5.4.2 光流与加速度计有融合，可能是温度变化比较大或飞机震动大，导致加速度计漂移严重；

5.5 飞机开始时能够正常悬停，悬停一段时间后开始一直往一个方向漂移，可能是：

5.5.1 程序算法处理上光流的权重是动态的，某些原因导致悬停时间长后，光流权重减弱，无法抑制漂移；

5.5.2 姿态解算的欧拉角由于震动或温度等出现较大误差，时间久后飞机本身姿态倾斜，光流最大输出后也补偿不了该误差；

5.6 飞机一开始就开始震荡，可能是：

5.6.1 PID 控制参数太大了；

5.6.2 滤波比较厉害，使得输出数据有延时，造成低频震荡；

5.6.3 输入数据不平滑，PID 控制中又有 D 项，使得飞机高频震荡；

5.6.4 飞机的性能变差，比如电机、桨叶磨损，震动大，使得输入数据噪声变大；

5.6.5 光流旋转补偿没有做好，尤其是数据同步与限幅；

附录 1：<up_flow.h> 与 <up_flow.c>

```
// up_flow.h


#ifndef _up_flow_H
#define _up_flow_H

#include "stm32f4xx.h"
#include "ms5611.h"




void Flow_Init(void);
void Flow_Duty(float dT);
void Flow_Get(u8);


extern struct flow_float flow_dat;


extern float exp_rol_flow,exp_pit_flow;

struct flow_integral_frame {
    unsigned short frame_count_since_last_readout;
    signed short pixel_flow_x_integral;
    signed short pixel_flow_y_integral;
    signed short gyro_x_rate_integral;
    signed short gyro_y_rate_integral;
    signed short gyro_z_rate_integral;
    unsigned int integration_timespan;
    unsigned int sonar_timestamp;
    unsigned short ground_distance;
    signed short gyro_temperature;
    unsigned char qual;
} ;

struct flow_float{
    float x;
    float y;
    unsigned short dt;
    unsigned char qual;
```

```
    unsigned char  update;
};#endif


// up_flow.c


#define LIMIT( x,min,max ) ( ((x) < (min)) ? (min) : ( ((x) > (max))? (max) : (x) ) )
#define LPF_1_(hz,t,in,out) ((out) += ( 1 / ( 1 + 1 / ( (hz) *3.14f *(t) ) ) ) *( (in) - (out) )) //一阶低通滤波
#define safe_div(numerator,denominator,safe_value) ( (denominator == 0)? (safe_value) : ((numerator)/(denominator)) )

void filter_1(float base_hz,float gain_hz,float dT,float in,_filter_1_st *f1)
//动态调整滤波截止频率的一阶滤波
{
    LPF_1_(gain_hz,dT,(in - f1->out),f1->a); //低通后的变化量

    f1->b = my_pow(in - f1->out);  //求一个数平方函数

    f1->e_nr = LIMIT(safe_div(my_pow(f1->a),((f1->b) + my_pow(f1->a)),0),0,1);
//变化量的有效率，LIMIT 将该数限制在 0-1 之间，safe_div 为安全除法

    LPF_1_(base_hz *f1->e_nr,dT,in,f1->out); //低通跟踪
}


_xyz_f_st heading_coordinate_acc;

_xyz_f_st heading_coordinate_speed_fus;
_xyz_f_st heading_coordinate_speed_err_i;

float fx_gyro_fix,fy_gyro_fix;
float fx_o,fy_o;
float f_out_x,f_out_y;
float gyro_lpf_x,gyro_lpf_y;

_filter_1_st f1_fx;
_filter_1_st f1_fy;

float f1_b,f1_g;

//flow_dat.qual：x，y 方向光流都有效：255，x 有效：2，y 有效：1，都无效：0;
```

```
float gyro_x,gyro_y;

void flow_fusion(float dT,float fx,float fy,s32 flow_height,u8 pos_hold)
{
    float flow_t1 = 1.0,flow_t2 = 1.0;

    fx_o = fx *10 *flow_height;        //fx, fy (rad/s) -->flow speed (mm/s)
    fy_o = fy *10 *flow_height;
    u32 use_height=100;

    if(flow_height<200)    //input height (cm/s)
    {
        use_height = 100;
    }
    else if(flow_height<300)
    {
        use_height = 150;
    }
    else if(flow_height<400)
    {
        use_height = 200;
    }
    else if(flow_height<500)
    {
        use_height = 250;
    }
    else if(flow_height<600)
    {
        use_height = 300;
    }
    else if(flow_height<1000)
    {
        use_height = 350;
    }
    else
    {
        use_height = 400;
    }


    if( pos_hold == 1)   //in pose hold mode
    {
```

```
        gyro_y = (((s16)sensor.Gyro_deg.y/2 )*2 );    //degree per second
        gyro_x = (((s16)sensor.Gyro_deg.x/2 )*2 );

            LPF_1_(3.0f,dT,gyro_y,gyro_lpf_y);      //gyro low pass filter
(delay) for fitting flow data()
            LPF_1_(3.0f,dT,gyro_x,gyro_lpf_x);

        fx_gyro_fix = ((fx   + LIMIT(((gyro_lpf_y)/57.3f),-flow_t1,flow_t1))
*10 *use_height ) ;  //rotation compensation
        fy_gyro_fix = ((fy   - LIMIT(((gyro_lpf_x)/57.3f),-flow_t2,flow_t2))
*10 *use_height ) ;

        vec_3d_transition(&imu_data.z_vec, &imu_data.a_acc,
&heading_coordinate_acc);   //机体坐标系下加速度到水平机体航向坐标系(将机体
水平放置，没有偏航角，

//保持加速度与光流没有偏航方向的影响)

        f1_fx.out += ((s32)heading_coordinate_acc.x/10 *10 ) *dT;
//integrated  acceleration for speed
        f1_fy.out += ((s32)heading_coordinate_acc.y/10 *10 ) *dT;

        f1_g = 2.5f;
        if(flow_dat.qual <3)  //光流效果不好时
        {
            f1_b = 0.5f;
        }
        else
        {
            if(f1_b<1.2f)
            {
                f1_b += 0.02f;
            }
        }

        filter_1(f1_b,f1_g,dT,fx_gyro_fix,&f1_fx);   //flow_data with acc
integrated data complementary filtering
        filter_1(f1_b,f1_g,dT,fy_gyro_fix,&f1_fy);

        heading_coordinate_speed_fus.x  = f1_fx.out;   //融合速度
        heading_coordinate_speed_fus.y  = f1_fy.out;
```

```
        f_out_x = heading_coordinate_speed_fus.x + 0.1f
*heading_coordinate_speed_err_i.x;  //融合速度二次修正，最终输出结果
        f_out_y = heading_coordinate_speed_fus.y + 0.1f
*heading_coordinate_speed_err_i.y;

        heading_coordinate_speed_err_i.x += (fx_gyro_fix - f_out_x) *dT;
        heading_coordinate_speed_err_i.y += (fy_gyro_fix - f_out_y) *dT;

}

}
```