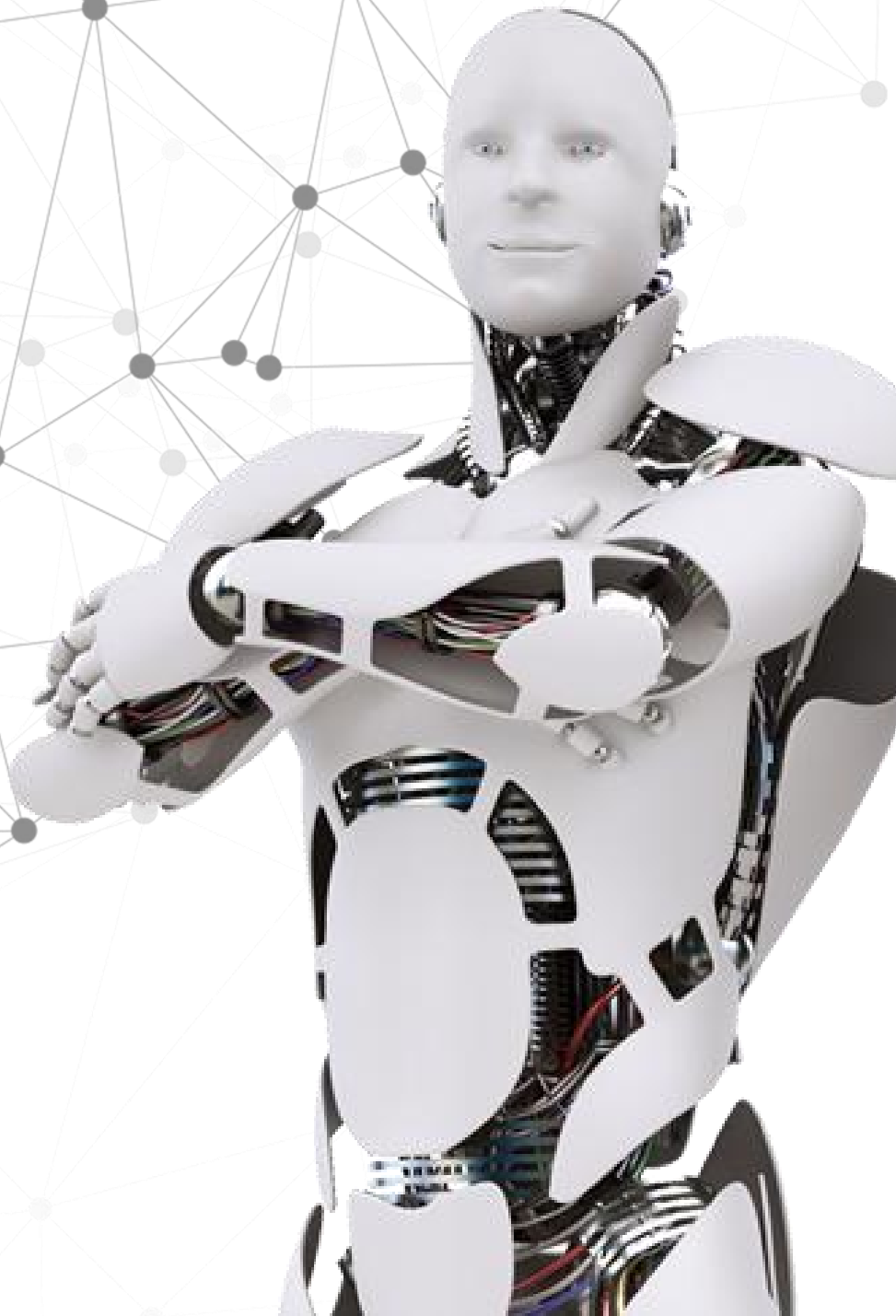
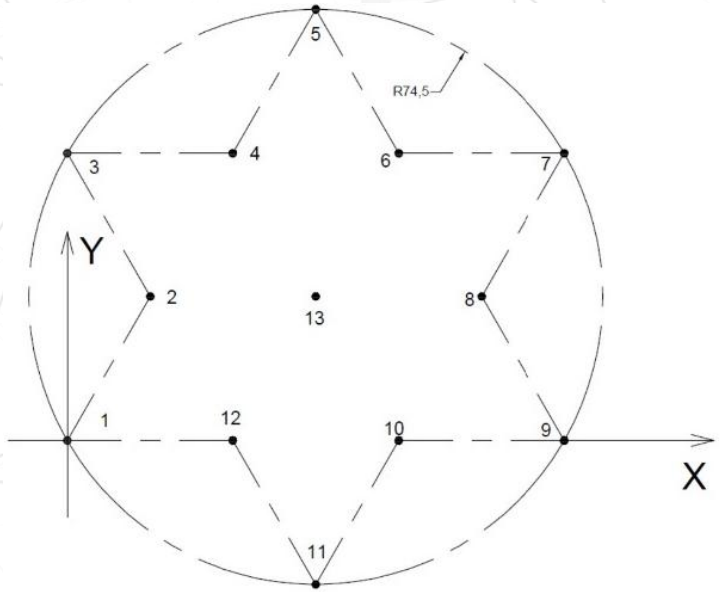
XY二轴插补程序编程



任务提出

已知任务中的插补轨迹上的各点坐标见下表，单位为mm，其中13为圆心位置。



序号	X	Y	序号	X	Y
1	0	0	11	64.5	-37.25
2	21.5	37.25	12	43	0
3	0	74.5	13	64.5	37.25
4	43	74.5			
5	64.5	111.75			
6	86	74.5			
7	129	74.5			
8	107.5	37.25			
9	129	0			

一、XY平台回零位

1．创建VS应用程序；

2．调用运动控制器函数库；

3．编写X轴和Y轴的回零运动程序；

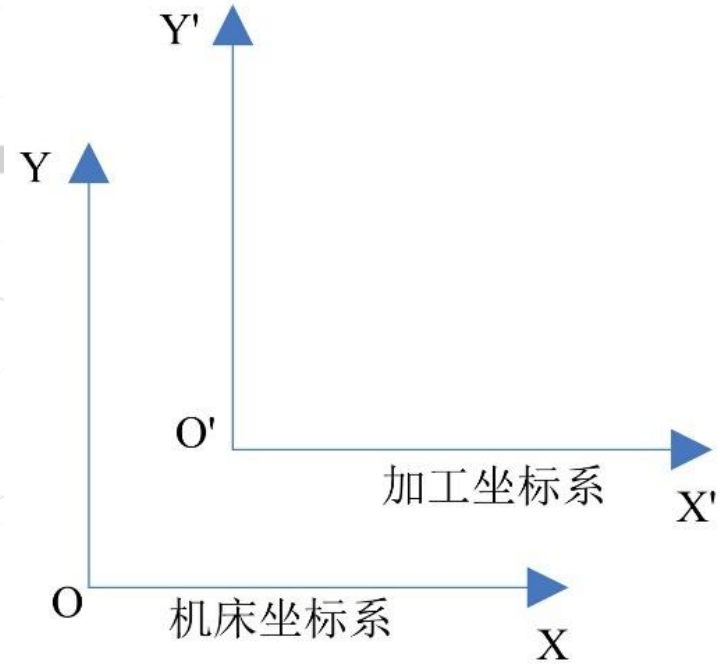
4．完成XY平台回零运动，注意回零运动结束，要对XY轴进行位置清零操作。

二、建立坐标系

1．坐标系建立

首先建立二维坐标系，将物理上的XY两个电机轴与软件计算中的XY两轴对应起来，获得一个以机械固定位置为原点的坐标系。坐标系建立后，末端工具的运动位置均可以由此坐标系表示。

在实际使用情况中，往往被加工的工件本身有一个独立的工件坐标系，工件坐标系的原点通常是加工的起始点，此时需要把机械坐标系与工件坐标系进行统一，此时可以通过坐标偏移的方式解决。



二、建立坐标系

```
void CCoordinateDlg::OnButtonInitialCoordinate()
{
        // TODO: Add your control notification handler code here
        short sRtn;
        TCrdPrm crdPrm;

        memset(&crdPrm,0,sizeof(crdPrm));
        crdPrm.dimension = 2;                   // 建立二维的坐标系
        crdPrm.synVelMax = 50;                  // 坐标系的最大合成速度是: 50 pulse/ms
        crdPrm.synAccMax = 1;                   // 坐标系的最大合成加速度是: 1pulse/ms^2
        crdPrm.evenTime = 50;                   // 坐标系的最小匀速时间为50ms
        crdPrm.profile[0] = 1;                  // 规划器1对应到X轴
        crdPrm.profile[1] = 2;                  // 规划器2对应到Y轴
        crdPrm.setOriginFlag = 1;               // 需要设置加工坐标系原点位置

        // 加工坐标系的原点坐标相对于回零位的坐标（-181879, -114730）
        crdPrm.originPos[0] =-181879;       // 加工坐标系的X轴相对于回零点的偏移量
        crdPrm.originPos[1] =-114730;       // 加工坐标系的Y轴相对于回零点的偏移量
        //根据设定的坐标系参数，建立1号坐标系
        sRtn = GT_SetCrdPrm(1,&crdPrm);
}
```

三、直线插补和圆弧插补程序

建立坐标系后，可直接使用插补指令控制末端工具的移动，如果需要末端工具进行一个直线移动，可直接使用运动控制卡的直线插补指令。通过直线插补指令，可以使末端工具在当前位置直线运动到工作空间中的另一指定坐标位置。

```
void CCoordinateDlg::OnButtonLinearMotion()
{
	short sRtn; // 指令返回值变量
	short run; // 坐标系运动状态查询变量
	long segment; // 坐标系运动完成段查询变量

	sRtn = GT_CrdClear(1, 0);  //首先清除坐标系1的FIFO0缓存区中的数据
	sRtn = GT_LnXY(          //直线插补，从回零位走到坐标（0，0）点
			1,            // 该插补段的坐标系是坐标系1
			0, 0,         // 该插补段的终点坐标(0, 0)
			50,           // 该插补段的目标速度：50pulse/ms
			0.1,          // 插补段的加速度：0.1pulse/ms^2
			0,            // 终点速度为0
			0);           // 向坐标系1的FIFO0缓存区传递该直线插补数据

	//开启激光
	sRtn = GT_BufIO(
			1,            // 坐标系是坐标系1
			MC_GPO,// 数字量输出类型：通用输出
			0x1000,    // bit13输出有操作
			0x0,          // bit13输出低电平
			0);           // 向坐标系1的FIFO0缓存区传递该数字量输出
 }
```

三、直线插补和圆弧插补程序

```
// 向缓存区写入插补数据，从1点到2点
sRtn = GT_LnXY(1, 21500, 32750, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从2点到3点
sRtn = GT_LnXY(1, 0, 74500, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从3点到4点
sRtn = GT_LnXY(1, 43000, 74500, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从4点到5点
sRtn = GT_LnXY(1, 64500, 111750, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从5点到6点
sRtn = GT_LnXY(1, 86000, 74500, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从6点到7点
sRtn = GT_LnXY(1, 129000,74500 , 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从7点到8点
sRtn = GT_LnXY(1, 107500,37250, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从8点到9点
sRtn = GT_LnXY(1, 129000, 0, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从9点到10点
sRtn = GT_LnXY(1, 86000, 0, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从10点到11点
sRtn = GT_LnXY(1, 64500, -37250, 50, 0.1, 0, 0);
// 向缓存区写入插补数据，从11点到12点
sRtn = GT_LnXY(1, 43000, 0, 50, 0.1, 0, 0);
```

```
// 向缓存区写入插补数据，从12点再回到坐标（0，0）点
sRtn = GT_LnXY(1, 0, 0, 50, 0.1, 0, 0);
//画整圆，圆的半径r=74500pulse
sRtn = GT_ArcXYC(
            1, // 坐标系是坐标系1
            0, 0, // 因为是画整圆，所以起点跟终点坐标一样，终点坐
标(0, 0),
            64500,37250, // 圆弧插补的圆心相对于起点位置的偏移量
            0, // 该圆弧是顺时针圆弧
            50, // 该插补段的目标速度：50pulse/ms
            0.1, // 该插补段的加速度：0.1pulse/ms^2
            0, // 终点速度为0
            0); // 向坐标系1的FIFO0缓存区传递该直线插补数据
//关闭激光
sRtn = GT_BufIO(
            1,          // 坐标系是坐标系1
            MC_GPO,// 数字量输出类型：通用输出
            0x1000,   // bit13输出有操作
            0x1000,   // bit13输出高电平
            0);         // 向坐标系1的FIFO0缓存区传递该数字量输出
```

三、直线插补和圆弧插补程序

```
        // 启动坐标系1的FIFO0的插补运动
        sRtn = GT_CrdStart(1, 0);
        //查询坐标系1的 FIFO0 插补运动坐标系状态
        sRtn = GT_CrdStatus(1, &run, &segment, 0);
        do   // 等待运动完成
        {
        // 查询坐标系1的FIFO的插补运动状态
        sRtn = GT_CrdStatus(
                1, // 坐标系是坐标系1
                &run, // 读取插补运动状态
                &segment, // 读取当前已经完成的插补段数
                0); // 查询坐标系1的FIFO0缓存区
         }while(run == 1); // 坐标系在运动, 查询到的run的值为1
}
```

谢 谢 观 看