Obstacle Avoidance Tutorial

1. Overview

The obstacle avoidance system detects whether there are obstacles within the flight distance of the aircraft, and will activate the brake mechanism automatically for protection. Currently, the system supports omnidirecional obstacle avoidance: front, rear, top, bottom, left, and right.

2. Enabling/Disabling Obstacle Avoidance

Parameter Configuration

Parameter: **VisionKey.KeyObstacleAvoidance**

```
val key = KeyTools.createKey(VisionKey.KeyObstacleAvoidance)
getKeyManager()?.setValue(key, open, object : CommonCallbacks.CompletionCallback
{
    override fun onSuccess() {

    }

    override fun onFailure(code: IAutelCode, msg: String?) {

    }
})
```

Warning Distance Parameters

There are warning distance parameters for horizontal, upward, and downward obstacle avoidance, respectively.

- Horizontal: **VisionKey.KeyHorizontalWarningDistance**
- Upward: **VisionKey.KeyTopWarningDistance**
- Downward: **VisionKey.KeyBottomWarningDistance**

Horizontal obstacle avoidance is used as an example.

```
val key = KeyTools.createKey(VisionKey.KeyHorizontalWarningDistance)
getKeyManager()?.setValue(key, warningDistance, object :
CommonCallbacks.CompletionCallback {
    override fun onSuccess() {

    }

    override fun onFailure(code: IAutelCode, msg: String?) {

    }
})
```

Brake Distance Parameters

There are brake distance parameters for horizontal, upward, and downward obstacle avoidance, respectively.

Horizontal: **VisionKey.KeyHorizontalBrakeDistance**

Upward: **VisionKey.KeyTopBrakeDistance**

Downward: **VisionKey.KeyBottomBrakeDistance**

Horizontal obstacle avoidance is used as an example.

```
val key = KeyTools.createKey(VisionKey.KeyHorizontalBrakeDistance)
getKeyManager()?.setValue(key, brakeDistance, object :
CommonCallbacks.CompletionCallback {
    override fun onSuccess() {

    }

    override fun onFailure(code: IAutelCode, msg: String?) {

    }
})
```

3. Reporting Radar Chart

The obstacle avoidance distance information during flight will be reported in real time, which contrasts the actual distance between obstacles with the warning and braking distances in the abovementioned directions.

```
val keyWarning = KeyTools.createKey(VisionKey.KeyReportEmergency)
getKeyManager()?.listen(keyWarning, object :
CommonCallbacks.KeyListener<List<VisionRadarInfoBean>> {
    override fun onValueChange(oldValue: List<VisionRadarInfoBean>?,newValue:
List<VisionRadarInfoBean>) {

    }
})
```

Vision And Radar Information

```
data class VisionRadarInfoBean(
    var timeStamp: Long = 0,//Timestamp
    var position: VisionSensorPositionEnum =
VisionSensorPositionEnum.FRONT,//Sensor position
    var distances: List<Float>? = null//Distances between sensors and obstacles
)
```

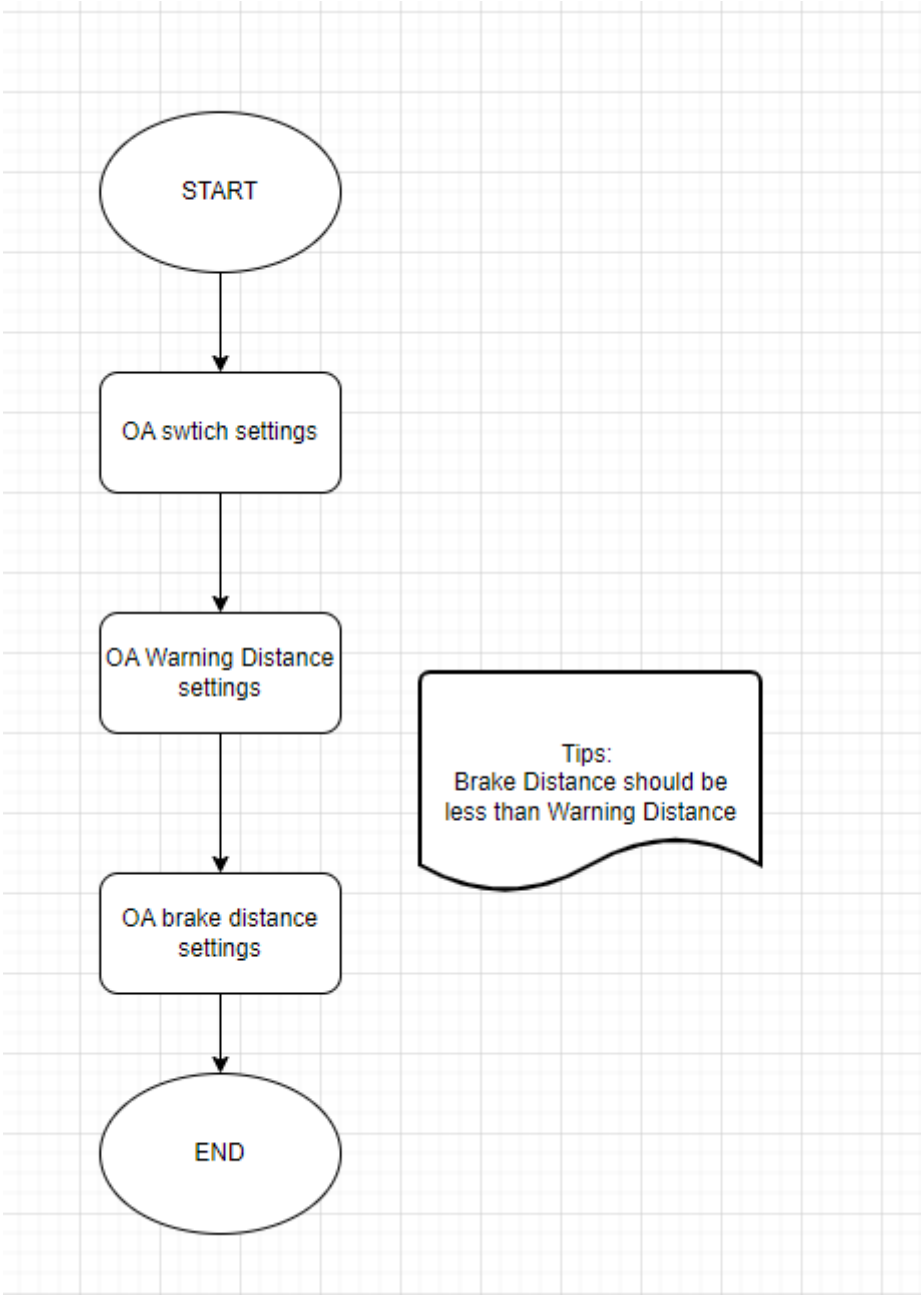
Vision Sensor Position

```
enum class VisionSensorPositionEnum(var value: Int) {
    FRONT(0),           //Front
    REAR(1),            //Rear
    BOTTOM(2),          //Bottom
    RIGHT(3),           //Right
    LEFT(4),            //Left
    TOP(5)              //Top
}
```

Sample

```
val keyWarning = KeyTools.createKey(VisionKey.KeyReportEmergency)
getKeyManager()?.listen(keyWarning, object :
CommonCallbacks.KeyListener<List<VisionRadarInfoBean>> {
    override fun onValueChange(oldValue: List<VisionRadarInfoBean>?, newValue:
List<VisionRadarInfoBean>) {
        for (value in newValue) {
            if (value.position == VisionSensorPositionEnum.TOP) {
                val distanceList = value.distances
                if (distanceList != null) {
                    if (distanceList.min() < getTopWarningDistance()) {
                        ToastUtils.showToast("Upward obstacles. Fly with
caution.")
                    }
                }
            }
        }
    }
})
```

4. API Calling Procedure


START
OA swtich settings
OA Warning Distance settings
OA brake distance settings
END
Tips:
Brake Distance should be less than Warning Distance