fovpro

你提出的疑问是非常重要的。让我们更详细地分析一下。**裁切后的 FOV 应该是根据裁切区域相对于原始图像的比例来缩小，而不是放大**，所以我的早期回答仍然有问题。让我重新解释并修正计算。

正确的 FOV 计算方式

当你对图片进行等比例裁切时，实际上相当于“缩小”了相机的视场，FOV 会变小。你裁切的图像区域越小，视角越窄，因此 FOV 应该随着裁切比例**变小**。

正确的公式：

1. **计算裁切比例**：
    - 比例 = 原始图像宽度 / 裁切后的图像宽度
2. **计算裁切后的 FOV**：
    - 裁切后的 FOV = 2 * atan((裁切后的宽度 / 原始图像宽度) * tan(原始 FOV / 2))

这是因为 FOV 是根据焦距和传感器尺寸计算出来的，当你裁切图片时，相当于缩小了传感器的有效使用区域。使用三角函数来计算新的视场角。

解释：

- 原始 FOV 表示原始图像的视场角。
- 如果你裁切了图像，你保留的宽度越小，FOV 就越小。

举例：

假设原始图像的分辨率是 4000x3000 像素，裁切后的图像是 2000x1500 像素，原始水平 FOV 是 70 度。

1. **计算比例**：
    - 裁切比例 = 原始宽度 / 裁切后的宽度 = 4000 / 2000 = 2
2. **使用公式计算裁切后的 FOV**：
    - 裁切后的水平 FOV = 2 * atan(tan(70 / 2) / 2) ≈ 38.02 度

裁切后的 FOV 是 38.02 度，视场角缩小了。

更正后的代码：

```
import kotlin.math.atan
import kotlin.math.tan
import kotlin.math.toDegrees
import kotlin.math.toRadians

fun calculateCroppedFOV(originalFOV: Double, originalWidth: Int,
croppedWidth: Int): Double {
    // 计算宽度比例
    val ratio = originalWidth.toDouble() / croppedWidth
    // 将 FOV 从度转换为弧度进行计算
    val originalFOVRadians = toRadians(originalFOV / 2)
    // 计算裁切后的 FOV
    val croppedFOVRadians = 2 * atan(tan(originalFOVRadians) / ratio)
    // 将结果转换为度
    return toDegrees(croppedFOVRadians)
}
```

使用示例：

```
val originalFOV = 70.0 // 原始 FOV
val originalWidth = 4000
val croppedWidth = 2000

val croppedFOV = calculateCroppedFOV(originalFOV, originalWidth,
croppedWidth)
println("Cropped FOV: $croppedFOV degrees")
```

结论：

- **FOV 变小**：当你对图像进行裁切时，裁切区域越小，视场角会缩小。裁切后保留的区域越小，FOV 越窄。
- 公式修正为使用三角函数来计算缩小的 FOV，确保考虑到裁切对实际视角的影响。

感谢你的耐心，这次的解释应该是准确的。