```
invert(frame->CameraViewTransform, &camToRef) //assume no error

float4x4 camToOrigin = camToRef * frame->FrameToOrigin;
Vector3f camPinhole(camToOrigin.m41, camToOrigin.m42, camToOrigin.m43);

Matrix3f camToOriginR;

camToOriginR(0, 0) = camToOrigin.m11;
camToOriginR(0, 1) = camToOrigin.m12;
camToOriginR(0, 2) = camToOrigin.m13;
camToOriginR(1, 0) = camToOrigin.m21;
camToOriginR(1, 1) = camToOrigin.m22;
camToOriginR(1, 2) = camToOrigin.m23;
camToOriginR(2, 0) = camToOrigin.m31;
camToOriginR(2, 1) = camToOrigin.m32;
camToOriginR(2, 2) = camToOrigin.m33;

for (size_t i = 0; i < arucoMarkerIds.size(); ++i)
{
markerCorners = arucoMarkers[i];

if (markerCorners.size() != 4)
{

continue;
}

for (size_t j = 0; j < markerCorners.size(); ++j)
{
DetectedMarker detectedMarker;

detectedMarker.markerId =
arucoMarkerIds[i] * 4 + j;

detectedMarker.x = static_cast<int>(markerCorners[j].x);
detectedMarker.y = static_cast<int>(markerCorners[j].y);
detectedMarker.point = camPinhole;

Point uv;

uv.X = static_cast<float>(markerCorners[j].x);
uv.Y = static_cast<float>(markerCorners[j].y);

Windows::Foundation::Point xy;

frame->SensorStreamingCameraIntrinsics
       ->MapImagePointToCameraUnitPlane(uv, &xy))

Vector3f dirCam;

dirCam[0] = xy.X;
dirCam[1] = xy.Y;
dirCam[2] = 1.0f;

detectedMarker.dir =
camToOriginR.transpose() * dirCam;
```

```
auto a = leftDetection.point;
auto b = leftDetection.dir;
auto c = rightDetection.point;
auto d = rightDetection.dir;
```

```
Matrix<float, 3, 2> A;

A.col(0) = b;
A.col(1) = -d;

Vector3f y = (c - a);

Vector2f x = (A.transpose() * A).inverse() * (A.transpose() * y);

TrackedMarker triangulatedMarkerCorner;

triangulatedMarkerCorner.markerId = leftDetection.markerId;
triangulatedMarkerCorner.point = (a + b * x[0] + c + d * x[1]) * 0.5f;
```