PhotonVisionLib LabVIEW Reference

Version 2023_3

Table of Contents

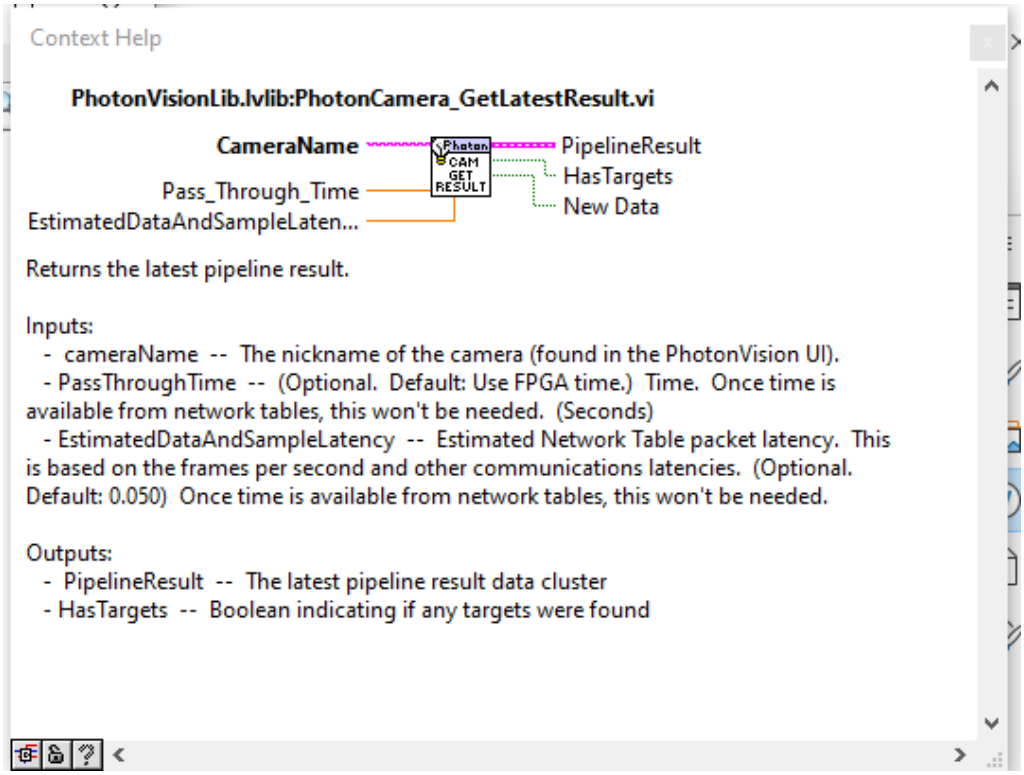
Introduction

The PhotonVision LabVIEW library provides utility functions to read, decode, and interpret data sent from PhotonVision application.

The library source code, package build specifications, and test package can be found here https://github.com/jsimpso81/PhotonVisionLabVIEW

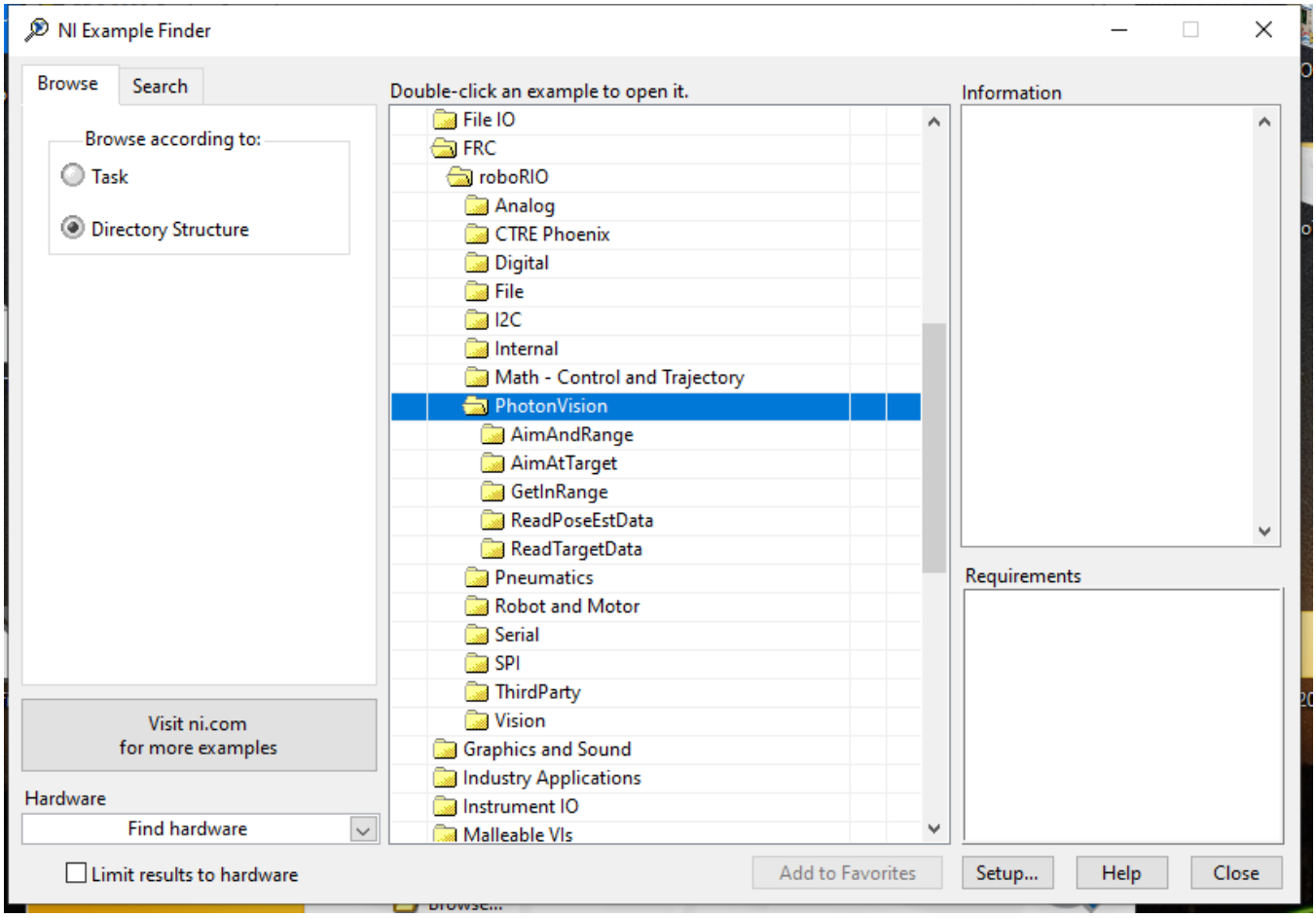
Function Help

Each VI includes help that can be accessed using the standard LabVIEW help toggle (Ctrl H).



Function Examples

Many of the functions have examples that can be found under the LabVIEW "Find examples..." function. (Help -> Find Examples...). The function examples are easiest to find when "Directory Structure" is selected.


NI Example Finder
Browse
Search
Browse according to:
Task
Directory Structure
Visit ni.com
for more examples
Hardware
Find hardware
Limit results to hardware
Double-click an example to open it.
File IO
FRC
roboRIO
Analog
CTRE Phoenix
Digital
File
I2C
Internal
Math - Control and Trajectory
PhotonVision
AimAndRange
AimAtTarget
GetInRange
ReadPoseEstData
ReadTargetData
Pneumatics
Robot and Motor
Serial
SPI
ThirdParty
Vision
Graphics and Sound
Industry Applications
Instrument IO
Malleable VIs
Information
Requirements
Add to Favorites
Setup...
Help
Close

Function Groups

Packet

Packet_decode_ByteI8

Decode (or strip out) an unsigned byte from an array of bytes. No conversion or decoding is actually done, the memory is just copied, and the value is cast to the correct type.

Inputs

- RawData -- Array of bytes containing the data to be extracted.

- StartByte -- The starting byte number (the first byte starts at 0)

Outputs

- Int8_Value -- Byte in the form of an INT 8 value.

- NextByte -- The starting byte index for the next value (regardless of type).

Packet_decode_Flt64

Decode (or strip out) a double precision floating point value FLT 64. No conversion or decoding is actually done, the memory is just copied, and the value is cast to the correct type.

Inputs

- RawData -- Array of bytes containing the data to be extracted.

- StartByte -- The starting byte number (the first byte starts at 0)

Outputs

- FLT64_Value -- Extracted float 64 (double) value.

- NextByte -- The starting byte index for the next value (regardless of type).

Packet_decode_Int32

Decode (or strip out) a 32 bit integer value from the provided byte array. No conversion or decoding is actually done, the memory is just copied, and the value is cast to the correct type.

Inputs

- RawData -- Array of bytes containing the data to be extracted.

- StartByte -- The starting byte number (the first byte starts at 0)

Outputs

- INT32_Value -- Extracted 32 bit integer value.

- NextByte -- The starting byte index for the next value (regardless of type).

Packet_decode_Transform3d

Decode (or strip out) aTransform3d value from the provided byte array. No conversion or decoding is actually done, the memory is just copied, and the value is cast to the correct type.

Inputs

- RawData -- Array of bytes containing the data to be extracted.

- StartByte -- The starting byte number (the first byte starts at 0)

Outputs

- Transform3d -- Extracted Transform3d cluster.

- NextByte -- The starting byte index for the next value (regardless of type).

Packet_encode_ByteI8

Encode (or append) an unsigned byte at the end of an array of bytes. No conversion is actually done, the memory is just copied to the end of the array.

Inputs

- RawData In -- Array of bytes to be appended to..

- Int8_Value -- Byte in the form of an INT 8 value.

Outputs

- RawData Out -- Array of bytes containing the appended value.

Packet_encode_FLT64

Encode (or append) a double floating point value (FLT 64) at the end of an array of bytes. No conversion is actually done, the memory is just copied to the end of the array.

Inputs

- RawData In -- Array of bytes to be appended to..

- FLT64_Value -- Double floating point value.

Outputs

- RawData Out -- Array of bytes containing the appended value.

Packet_encode_Int32

Encode (or append) a 32 bit integer at the end of an array of bytes. No conversion is actually done, the memory is just copied to the end of the array.

Inputs

- RawData In -- Array of bytes to be appended to..

- In32_Value -- 32 bit integer the form of an INT 32 value.

Outputs

- RawData Out -- Array of bytes containing the appended value.

Packet_encode_Transform3d

Encode (or append) a double floating point value (FLT 64) at the end of an array of bytes. No conversion is actually done, the memory is just copied to the end of the array.

Inputs

- RawData In -- Array of bytes to be appended to..

- FLT64_Value -- Double floating point value.

Outputs

- RawData Out -- Array of bytes containing the appended value.

PhotonCamera

PhotonCamera_GetDriverMode

Returns whether the camera is in driver mode.

Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- DriverMode -- Boolean indicating whether the camera is in driver mode.

PhotonCamera_GetLEDMode

Returns the current LED mode.

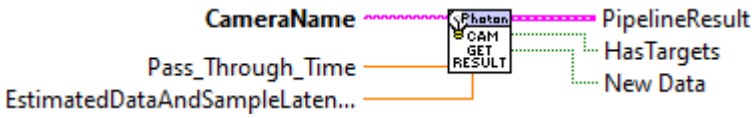
Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- LedMode -- The current LED mode.

PhotonCamera_GetLatestResult



Returns the latest pipeline result.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).

- PassThroughTime -- (Optional. Default: Use FPGA time.) Time. Once time is available from network tables, this won't be needed. (Seconds)

- EstimatedDataAndSampleLatency -- Estimated Network Table packet latency. This is based on the frames per second and other communications latencies. (Optional. Default: 0.050) Once time is available from network tables, this won't be needed.

Outputs:

- PipelineResult -- The latest pipeline result data cluster

- HasTargets -- Boolean indicating if any targets were found

PhotonCamera_GetPipelineIndex

Returns the active pipeline index.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- PipelineIndex -- Active Pipeline Index

PhotonCamera_IsConnected

Determines if communications to camera is valid.

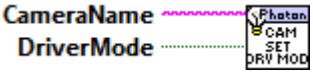
Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- IsConnected -- TRUE if communications have been received within the last 0.5 seconds.

PhotonCamera_SetDriverMode



Toggles driver mode.

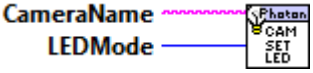
Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- DriverMode -- Whether to set driver mode.

Outputs:

- >none<

PhotonCamera_SetLEDMode



Sets the LED mode.

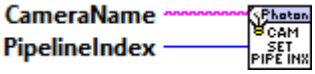
Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).
- LedMod -- The mode to set to.

Outputs:

- >none<

PhotonCamera_SetPipelineIndex



Allows the user to select the active pipeline index.

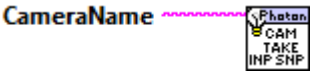
Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- index The active pipeline index.

Outputs:

- >none<

PhotonCamera_TakeInputSnapshot



Request the camera to save a new image file from the input camera stream with overlays. Images take up space in the filesystem of the PhotonCamera. Calling it frequently will fill up disk space and eventually cause the system to stop working. Clear out images in /opt/photonvision/photonvision_config/imgSaves frequently to prevent issues.

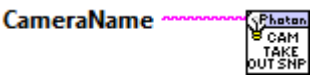
Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- >none<

PhotonCamera_TakeOutputSnapshot



Request the camera to save a new image file from the output stream with overlays. Images take up space in the filesystem of the PhotonCamera. Calling it frequently will fill up disk space and eventually cause the system to stop working. Clear out images in /opt/photonvision/photonvision_config/imgSaves frequently to prevent issues.

Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).

Outputs:

- >none<

PhotonEstimatedRobotPose

PhotonEstimatedRobotPose_GetAll

Extracts individual data items from a Photon Estimated Robot Pose cluster.

Inputs:

- PhotonEstimatedRobotPose -- PhotonEstimatedRobotPose -- Data cluster..

Outputs:

-- EstimatedPose -- Pose3d -- Estimated absolute position of robot on field.

-- TimeStamp -- Double -- Time stamp (seconds) of the packet from photonvision. This uses the FGPA elapsed time.

PhotonEstimatedRobotPose_New

Create a new Photon Estimated Robot Pose cluster.

Inputs:

-- EstimatedPose -- Pose3d -- Estimated absolute position of robot on field.

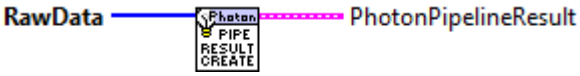
-- TimeStamp -- Double -- Time stamp (seconds) of the packet from photonvision. This uses the FGPA elapsed time.

Outputs:

- PhotonEstimatedRobotPose -- Created data cluster..

PhotonPipelineResult

PhotonPipelineResult_CreateFromPacket



Internal function to parse the data returned by PhotonCamera_GetLatestResult.

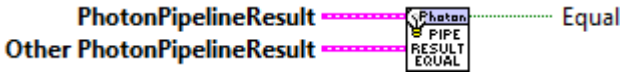
Inputs:

- RawData -- Byte array containing the data to parse.

Outputs:

- PipelineResult -- Parsed PipelineResult cluster.

PhotonPipelineResult_Equals



Determines if two PipelineResults are equal.

Inputs:

- PipelineResult -- PipelineResult cluster.

- Other PipelineResult -- Other PipelineResult cluster.

Outputs:

- Equal -- Returns TRUE, if both pipeline results are equal..

PhotonPipelineResult_GetBestTarget

Returns the best target in this pipeline result. If there are no targets, this method will return an empty target. The best target is determined by the target sort mode in the PhotonVision UI.

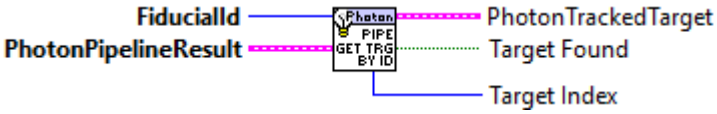
Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- TrackedTarget -- Cluster containing the best target

PhotonPipelineResult_GetBestTargetById



Returns the best target in this pipeline result filtered by fiducial ID (April tag numer). If there are no targets, or no targets with this fiducial ID, this method will return an empty target. IF more than 1 targets have the same fiducial ID, the best target is determined by the sort mode in the PhotonVision UI.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

- Fiducial ID -- The April tag number being searched for.

Outputs:

- TrackedTarget -- Cluster containing the best target

- TargetFound -- TRUE if a target matching the search criteria exits.

- TargetIndex -- Index into the targets array of the best target matching the search criteria.

PhotonPipelineResult_GetLatency_Millis

Returns the latency in the pipeline.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- Latency_Millis -- The latency in the pipeline.
- TimeStampSec -- Time stamp of packet (FPGA elapsed time) seconds

---

PhotonPipelineResult_GetTargets

Returns the targets in this pipeline result. If there are no targets, this method will return an empty target array.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- Targets -- Array of Target clusters.

PhotonPipelineResult_GetTimeStampSecs

Returns pipeline packet time stamp

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- TimeStampSec -- Time stamp of packet (FPGA elapsed time) seconds

---

PhotonPipelineResult_HasTargets

Returns whether the pipeline has targets.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- HasTargets -- Whether the pipeline has targets.

PhotonPipelineResult_New

Create a new PhotonPipelineResult cluster.

Inputs:

- Latency_Millis -- Latency, milliseconds
- TrackedTargets -- Array of TracedTarget clusters

Outputs:

- PipelineResult -- Parsed PipelineResult cluster.

PhotonPipelineResult_PopulatePacket

Convert a PipelineResult to a Packet array of bytes ready for writing to a Network Table variable.

Inputs:

- PipelineResult -- PipelineResult cluster.

Outputs:

- Pacet_RawData -- Byte array containing the encoded PipelineResult.

PhotonPipelineResult_SetTimeStampSecs

Returns pipeline packet time stamp

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.
- TimeStampSec -- Time stamp of packet (FPGA elapsed time) seconds

Outputs:

- Out PipelineResult -- Updated PipelineResult cluster.

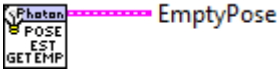
PhotonPoseEstimator

PhotonPoseEstimator_CalcDifference

This is an internal function. Use Update or Update_Pipeline instead of this function.

It calculates an overall difference between two poses. Difference is defined as the vector magnitude between the two poses

PhotonPoseEstimator_GetEmptyPose



This is an internal function.

This returns an empty pose3d.

Inputs:

--

Outputs:

-- EmptyPose -- Pose3d -- Empty Pose

PhotonPoseEstimator_GetFieldTags

Get the AprilTagFIeldLayout cluster from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- AprilTagFieldLayout -- cluster -- Contains the absolute definitions of the April tags on a field.

PhotonPoseEstimator_GetHighestAmbiguity

Get the highest allowed ambiguity from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- HighestAmbiguity -- double -- This is the highest allowed ambiguity for targets to be considered valid. Ambiguity describes the difference between the "best" and "alternate" targets. Lower numbers are better.

PhotonPoseEstimator_GetLastPose

Get the last pose3d calculated by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- LastPose -- Pose3d -- Last robot pose calculated by the PhotonPoseEstimator

---

PhotonPoseEstimator_GetReferencePose

Get the reference pose from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- referencePose -- Pose3d -- This is the reference pose being used by the Photon Pose Estimator. Often this would be set the the current Pose calculated by the robot's odometry.

PhotonPoseEstimator_GetRobotToCamera

Get the robot to camera transform from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- RobotToCamera -- Transform3d -- This describes the location of the camera relative to the robot.

PhotonPoseEstimator_GetStrategy

Get the current strategy being used by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- PhotonPoseStrategy -- enum -- Current strategy being used by the Photon Pose Estimator

---

PhotonPoseEstimator_New

Create a new Photon Pose Estimator data cluster.

The PhotonPoseEstimator functions filters or combines readings from all the AprilTags visible at a given timestamp on the field to produce a single robot in field pose, using the strategy set below. Example usage can be found under the LabVIEW Find Examples..

Inputs:

-- AprilTagFieldLayout -- cluster -- A WPILib LabVIEW Math cluster defining the absolute positions of April Tags on a FIRST field. with respect to the FIRST field

-- Strategy -- Enum -- The strategy it should use to determine the best pose. Current strategies include:

LOWEST_AMBIGUITY -- Choose the Pose with the lowest ambiguity

CLOSEST_TO_CAMERA_HEIGHT -- Choose the Pose which is closest to the camera height.

CLOSEST_TO_REFERENCE_POSE -- Choose the Pose which is closest to a set Reference position.

CLOSEST_TO_LAST_POSE -- Choose the Pose which is closest to the last pose calculated

AVERAGE_BEST_TARGETS -- Weight average of acceptable targets based on ambiguity..

-- CameraName -- String -- Name of the photon camera.

-- RobotToCamera -- Transform3d -- Transform from the center of the robot to the camera mount positions

Outputs:

-- PhotonPoseEstimator -- cluster -- created data cluster.

PhotonPoseEstimator_SetFieldTags

Set the AprilTagFIeldLayout cluster into the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- AprilTagFieldLayout -- cluster -- Contains the absolute location definitions of the April tags on a field.

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetHighestAmbiguity

Set the highest allowed ambiguity from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- HighestAmbiguity -- double -- This is the highest allowed ambiguity for targets to be considered valid. Ambiguity describes the difference between the "best" and "alternate" targets. Lower numbers are better.

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetLastPose

Set the last pose3d calculated by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- LastPose -- Pose3d -- Last robot pose calculated by the PhotonPoseEstimator

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetLastPose_Pose2d

Set the last pose3d calculated by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- LastPose -- Pose2d -- Last robot pose calculated by the PhotonPoseEstimator

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetReferencePose

Set the reference pose from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- referencePose -- Pose3d -- This is the reference pose being used by the Photon Pose Estimator. Often this would be set the the current Pose calculated by the robot's odometry.

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetReferencePose_Pose2d

Set the reference pose from the PhotonPoseEstimator cluster.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- referencePose -- Pose2d -- This is the reference pose being used by the Photon Pose Estimator. Often this would be set the the current Pose calculated by the robot's odometry.

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_SetStategy

Set the current strategy being used by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- PhotonPoseStrategy -- enum -- Current strategy being used by the Photon Pose Estimator. Current strategies include:

LOWEST_AMBIGUITY -- Choose the Pose with the lowest ambiguity

CLOSEST_TO_CAMERA_HEIGHT -- Choose the Pose which is closest to the camera height.

CLOSEST_TO_REFERENCE_POSE -- Choose the Pose which is closest to a set Reference position.

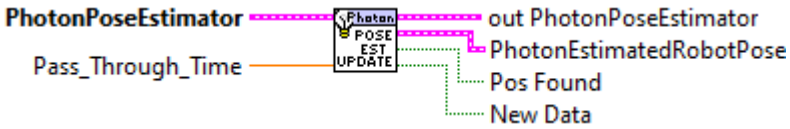
CLOSEST_TO_LAST_POSE -- Choose the Pose which is closest to the last pose calculated

AVERAGE_BEST_TARGETS -- Weight average of acceptable targets based on ambiguity..

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated estimator.

PhotonPoseEstimator_Update



Poll data from the configured cameras and update the estimated position of the robot. Returns empty if there are no cameras set or no targets were found from the cameras.

Inputs:

-- PhotonPoseEstimator -- data cluster -- Data guiding how the position determination is done.

-- PassThroughTime -- double -- FPGA time stamp in seconds. (Optional. Default: Read FPGA time.)

Outputs:

-- PhotonPoseEstimator -- data cluster -- Updated data cluster.

-- PhotonEstimatedRobotPose -- data cluster -- Estimated robot position and timestamp.

-- PosFound -- boolean -- A position has been determined.

-- NewData -- boolean -- The data from photonvision is newer than the last time this routine was called.

PhotonPoseEstimator_Update_PipeResult

Provided data from polling a camera, update the estimated position of the robot. Returns empty if there are no cameras set or no targets were found from the cameras.

Inputs:

-- PhotonPoseEstimator -- data cluster -- Data guiding how the position determination is done.

-- PipelineResults -- data cluster -- Data provided from PhotonCamera_GetLatestResults.

-- HasResults -- boolean - TRUE if targets have been identified in the Pipeline results. This can be provided by PhotonCamera_GetLatestResults.

Outputs:

-- PhotonPoseEstimator -- data cluster -- Updated data cluster.

-- PhotonEstimatedRobotPose -- data cluster -- Estimated robot position and timestamp.

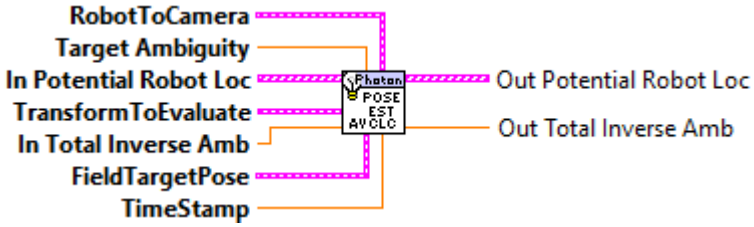
-- PosFound -- boolean -- A position has been determined.

PhotonPoseEstimator_averageBestTargetsStrategy

This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based the weighted average position of valid potential positions.

PhotonPoseEstimator_averageBestTargetsStrategy_calc



This is an internal function. Use Update or Update_Pipeline instead of this function.

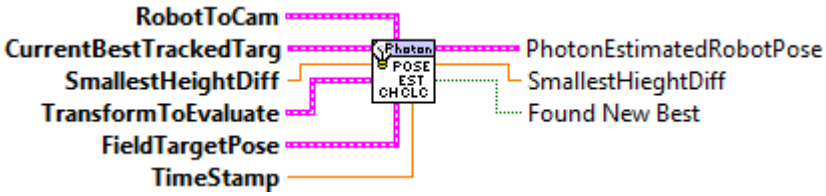
Determine the best target based the weighted average position of valid potential positions.

PhotonPoseEstimator_closestToCameraHeightStrategy

This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based on the target height closed to the camera height.

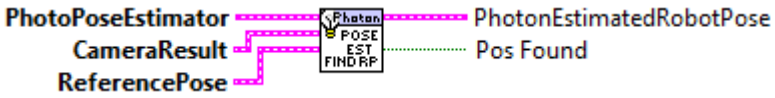
PhotonPoseEstimator_closestToCameraHeightStrategy_calc



This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based on the target height closed to the camera height.

PhotonPoseEstimator_closestToReferencePoseStrategy

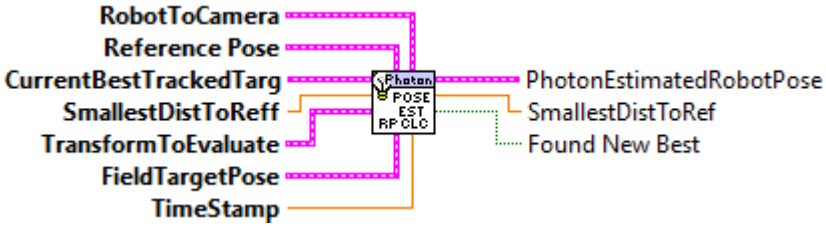


This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based on the closest to a set Reference position

---

PhotonPoseEstimator_closestToReferencePoseStrategy_calc



This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based on the closest to a set Reference position

---

PhotonPoseEstimator_lowestAmbiguityStrategy

This is an internal function. Use Update or Update_Pipeline instead of this function.

Determine the best target based the lowest ambiguity.

PhotonTrackedTarget

PhotonTrackedTarget_CreateFromPacket

Internal function to parse the data for each target returned by PhotonCamera_GetLatestResult.

Inputs:

- RawData -- Byte array containing the data to parse.

Outputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

PhotonTrackedTarget_Equals

Determines if two tracked targets are equal

Inputs:

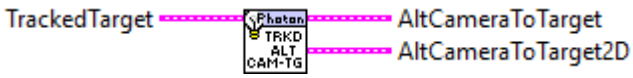
- TrackedTarget -- TrackedTarget cluster.

- OtherTrackedTarget -- Second TrackedTarget cluster.

Outputs:

- Area -- Boolean, equals TRUE if both TrackedTargets are the same.

PhotonTrackedTarget_GetAltCameraToTarget



Get CameraToTarget Transform2d from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- CameraToTarget -- Camera to Target Transform2d

PhotonTrackedTarget_GetArea

Get Area from Tracked Target

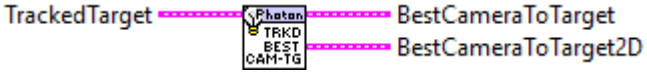
Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Area -- Area value

PhotonTrackedTarget_GetBestCameraToTarget



Get CameraToTarget Transform2d from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- CameraToTarget -- Camera to Target Transform2d

PhotonTrackedTarget_GetDetectedCorners

Get Detected Corners from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Detected Corners -- Array of TargetCorner cluster containing detected corners

---

PhotonTrackedTarget_GetFiducialId

Get Area from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Area -- Area value

PhotonTrackedTarget_GetMinAreaRectCorners

Get Minimum Area Rectagle Corners from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- MinAreaRectCorners -- Corners array of TargetCorner cluster

---

PhotonTrackedTarget_GetPitch

Get Pitch from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Pitch -- Pitch value

PhotonTrackedTarget_GetPoseAmbiguity

Get Area from Tracked Target

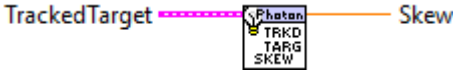
Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Area -- Area value

PhotonTrackedTarget_GetSkew



Get Skew from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Skew -- Skew value

PhotonTrackedTarget_GetYaw

Get Yaw from Tracked Target

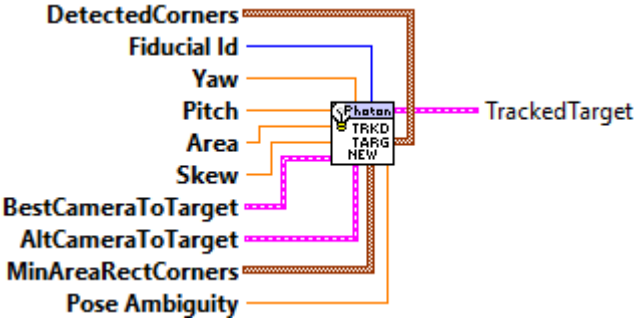
Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Yaw -- Yaw value

PhotonTrackedTarget_New



Create a new TracedTarget data cluster

Inputs:

- Yaw --
- Pitch --
- Area --
- Skew --
- CameraToTarget -- Transform2d
- MinAreaRectCorners -- Array of 4 corners.
- Pose Ambiguity
- Fiducial ID

- Detected Corners

Outputs:

- TrackedTarget -- Created TrackedTarget cluster.

PhotonTrackedTarget_PopulatePacket

Encode a TrackedTarget cluster into an array of bytes.

Inputs:

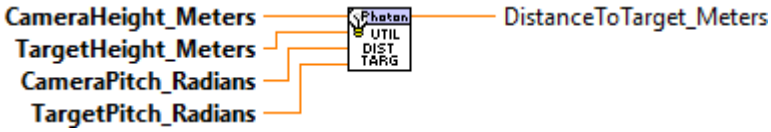
- TrackedTarget -- TrackedTarget cluster.

Outputs:

- RawDataOut -- Byte array containing the encoded data.

PhotonUtils

PhotonUtils_CalculateDistanceToTarget



Algorithm from https://docs.limelightvision.io/en/latest/cs_estimating_distance.html Estimates range to a target using the target's elevation. This method can produce more stable results than SolvePNP when well tuned, if the full 6d robot pose is not required. Note that this method requires the camera to have 0 roll (not be skewed clockwise or CCW relative to the floor), and for there to exist a height differential between goal and camera. The larger this differential, the more accurate the distance estimate will be.

Units can be converted using the {@link edu.wpi.first.math.util.Units} class.

Inputs:

- CameraHeight_Meters -- The physical height of the camera off the floor in meters.
- TargetHeightMeters -- The physical height of the target off the floor in meters. This should be the height of whatever is being targeted (i.e. if the targeting region is set to top, this should be the height of the top of the target).
- CameraPitch_Radians -- The pitch of the camera from the horizontal plane in radians. Positive values up.
- TargetPitchRadian -- The pitch of the target in the camera's lens in radians. Positive values up.

Outputs

- DistanceToTarget_Meters -- The estimated distance to the target in meters.

PhotonUtils_EstimateCameraToTarget

Estimates a Transform2d that maps the camera position to the target position, using the robot's gyro. Note that the gyro angle provided *must* line up with the field coordinate system -- that is, it should read zero degrees when pointed towards the opposing alliance station, and increase as the robot rotates CCW.

Inputs:

- CameraToTargetTranslation -- A Translation2d that encodes the x/y position of the target relative to the camera.
- FieldToTarget -- A Pose2d representing the target position in the field coordinate system.
- GyroAngle -- The current robot gyro angle, likely from odometry.

Outputs:

- EstimateCameraToTarget -- A Transform2d that takes us from the camera to the target.

PhotonUtils_EstimateCameraToTargetTrans

Estimate the Translation2d of the target relative to the camera.

Inputs:

- TargetDistance_Meters -- The distance to the target in meters.
- Yaw -- The observed yaw of the target.

Outputs:

- CameraToTarget -- The target's camera-relative translation.

PhotonUtils_EstimateFieldToCamera

Estimates the pose of the camera in the field coordinate system, given the position of the target relative to the camera, and the target relative to the field. This *only* tracks the position of the camera, not the position of the robot itself.

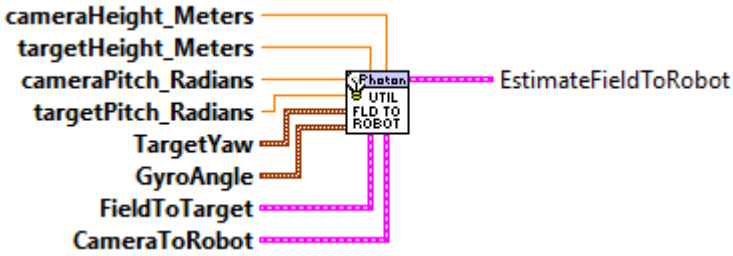
Inputs:

- CameraToTarget -- The position of the target relative to the camera.
- FieldToTarget -- The position of the target in the field.

Outputs:

- EstimateFieldToCamera -- The position of the camera in the field.

PhotonUtils_EstimateFieldToRobot



Estimate the position of the robot in the field.

Inputs:

- CameraHeightMeters The physical height of the camera off the floor in meters.
- TargetHeightMeters The physical height of the target off the floor in meters.

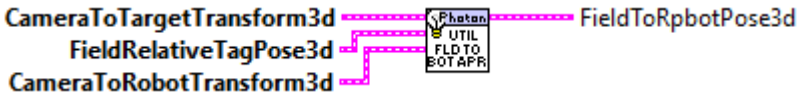
  This should be the height of whatever is being targeted (i.e. if the targeting region is set to top, this should be the height of the top of the target).
- CameraPitchRadians The pitch of the camera from the horizontal plane in radians. Positive values up.
- TargetPitchRadians The pitch of the target in the camera's lens in radians. Positive values up.

- TargetYaw The observed yaw of the target. Note that this *must* be CCW-positive, and Photon returns CW-positive.
- GyroAngle The current robot gyro angle, likely from odometry.
- FieldToTarget A Pose2d representing the target position in the field coordinate system.
- CameraToRobot The position of the robot relative to the camera. If the camera was mounted 3 inches behind the "origin" (usually physical center) of the robot, this would be Transform2d (3 inches, 0 inches, 0 degrees).

Outputs

- EstimateFieldToRobot -- The position of the robot in the field.

PhotonUtils_EstimateFieldToRobotAprilTag



Estimates the pose3d of the robot in the field coordinate system, given the pose3d of the fiducial tag, the robot relative to the camera, and the target relative to the camera.

nputs:

- CameraToTarget -- Transform3D of the target relative to the camera, returned by PhotonVision
- FieldRelativeTagPose3d -- The field relative pose3d of the target
- CameraToRobot -- Transform3D of the robot relative to the camera. Origin of the robot is defined as the center.

Outputs

- FieldToRobot -- Transform3d Robot position relative to the field

---

PhotonUtils_EstimateFieldToRobot_Alt

Estimates the pose of the robot in the field coordinate system, given the position of the target relative to the camera, the target relative to the field, and the robot relative to the camera.

Inputs:

- CameraToTarget -- The position of the target relative to the camera.
- FieldToTarget -- The position of the target in the field.
- CameraToRobot -- The position of the robot relative to the camera. If the camera was mounted 3 inches behind the "origin" (usually physical center) of the robot, this would be Transform2d(3 inches, 0 inches, 0 degrees).

Outputs:

- EstimateFieldToRobot -- The position of the robot in the field.

PhotonUtils_GetDistanceToPose

Returns the distance between two poses

Inputs:

- RobotPose -- Pose2d of the robot.
- TargetPose -- Pose2d of the target

Outputs

- DistanceToPose -- The calculated distance between the poses.

PhotonUtils_GetYawToPose

Returns the yaw between your robot and a target.

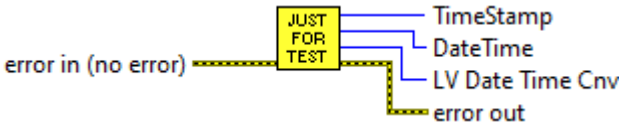
Inputs:

- RobotPose -- Pose2d of the robot.
- TargetPose -- Pose2d of the target

Outputs

- YawToPose -- Yaw to the target

PhotonUtils_TestGetDateTime



TargetCorner

TargetCorner_Equals

Determines if two target corners are equal

Inputs:

- TargetCorner -- TargetCorner cluster

- OtherTargetCorner -- TargetConrer cluster to compare

Outputs:

- Equals -- TRUE if both TargetCorners are equal

TargetCorner_GetAll

Get the individual components of a TargetCorner

Inputs:

- TargetCorner -- TargetCorner cluster

Outputs:

- X --

- Y --

TargetCorner_New

Create a new TargetCorner data cluster

Inputs:

- X --
- Y --

Outputs:

- TargetCorner -- Created TargetCorner cluster.

Type Definitions

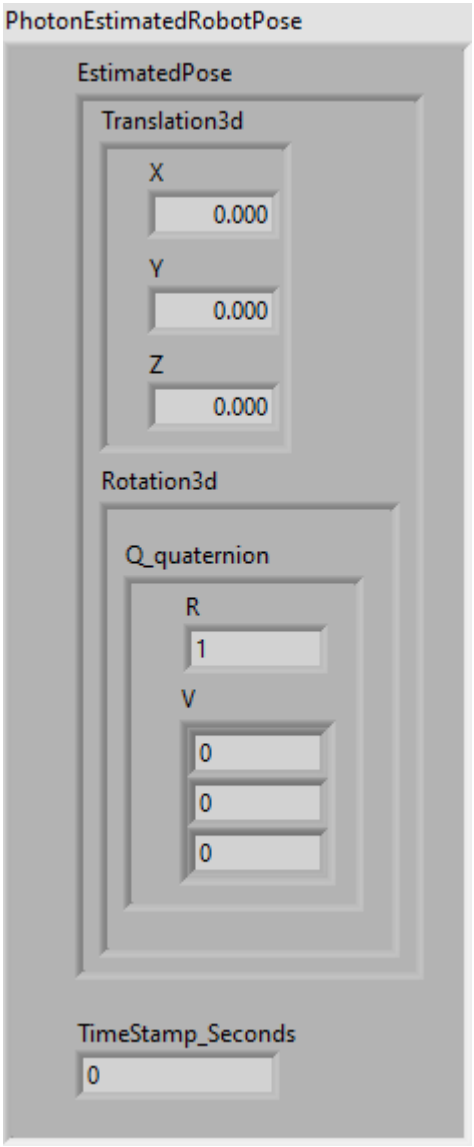
TypeDef

TypeDef-PhotonEstimatedRobotPoseType

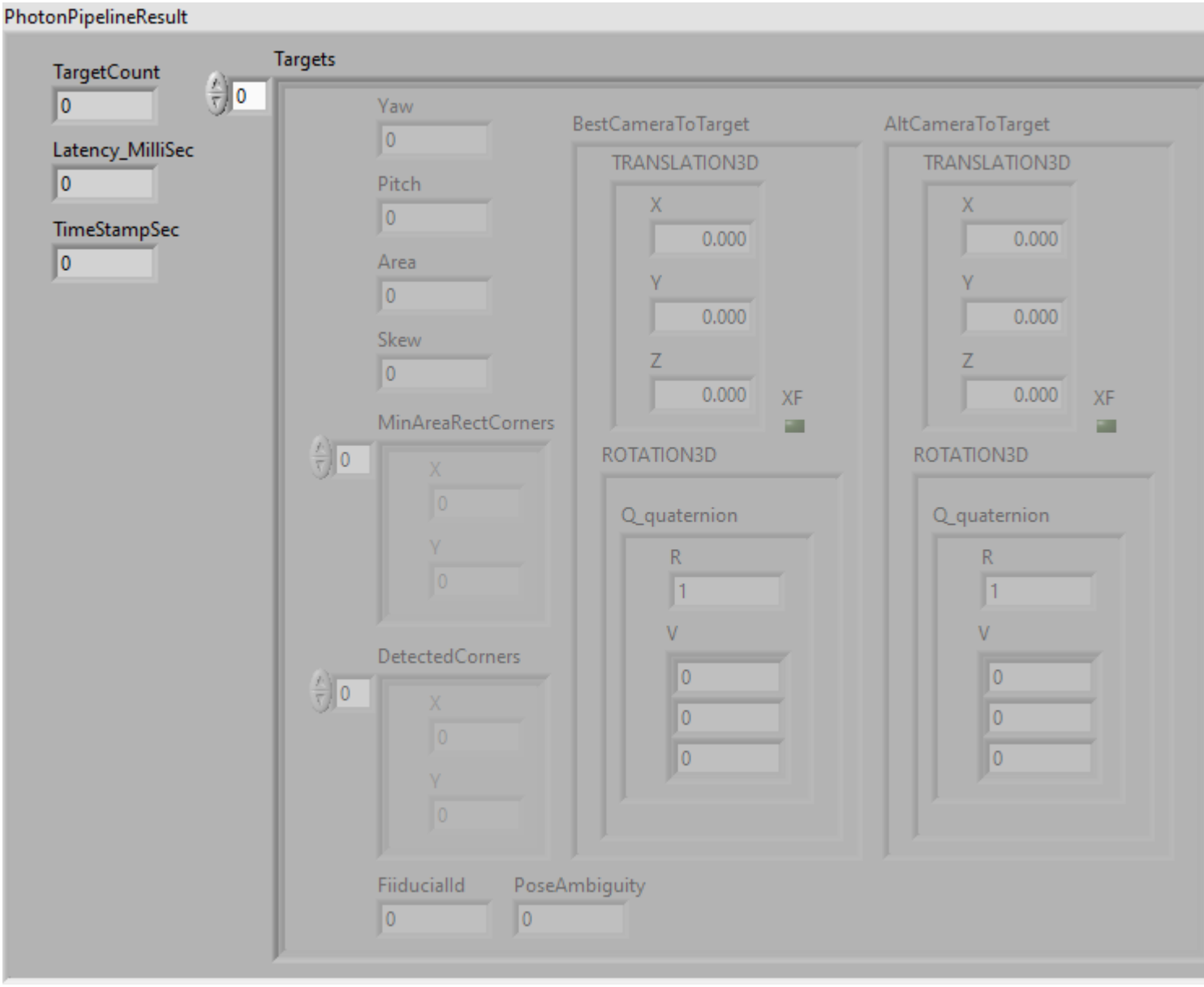
An estimated pose based on pipeline result. This cluster contains:

-- EstimatedPose -- Pose3d -- Estimated absolute robot position based on vision location of April Tags

-- TimeStamp_Seconds -- Double -- Timestamp of data from photonvision. This can be set by the user to add latency or other time offsets.

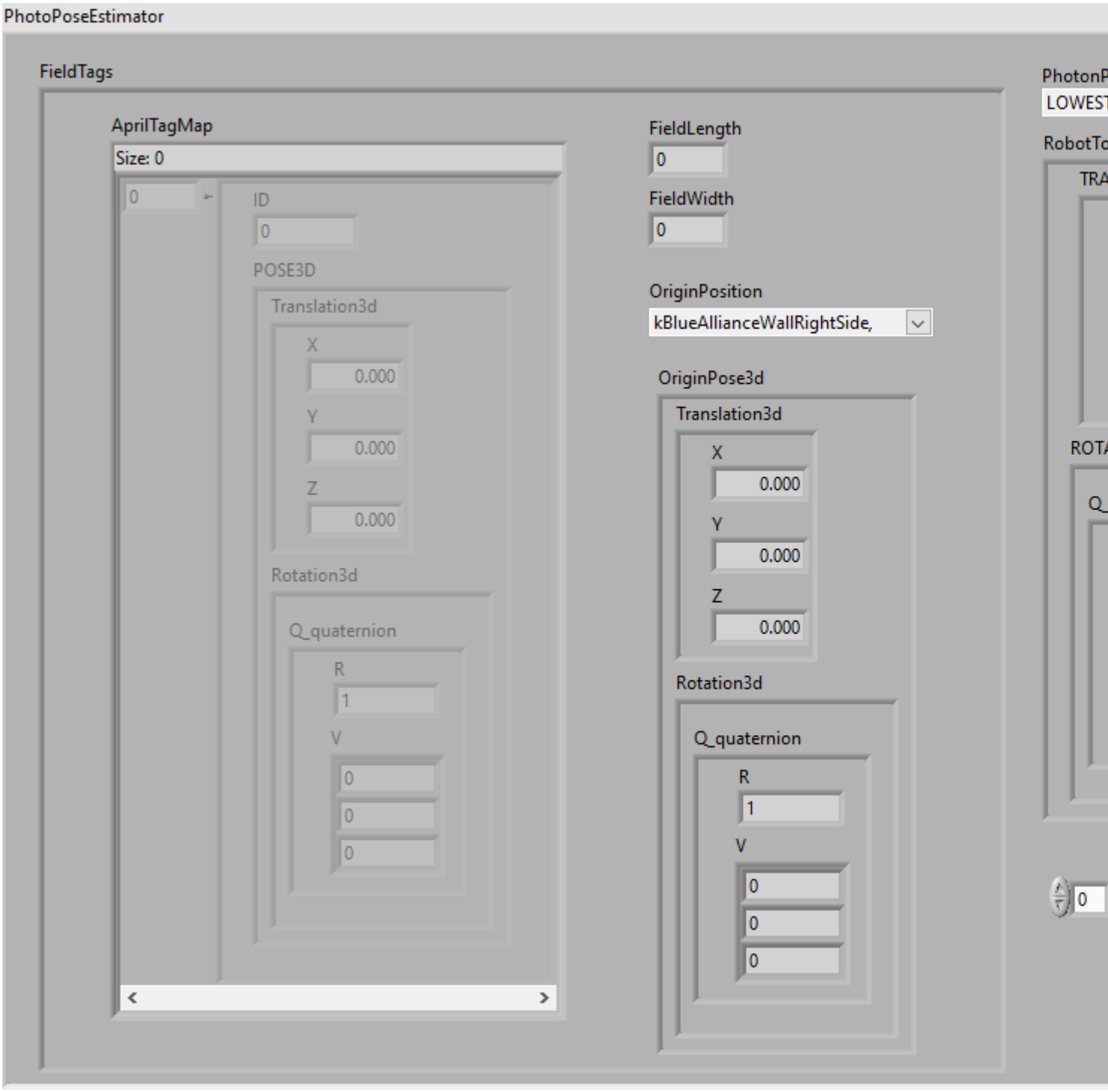


TypeDef-PhotonPipelineResultType



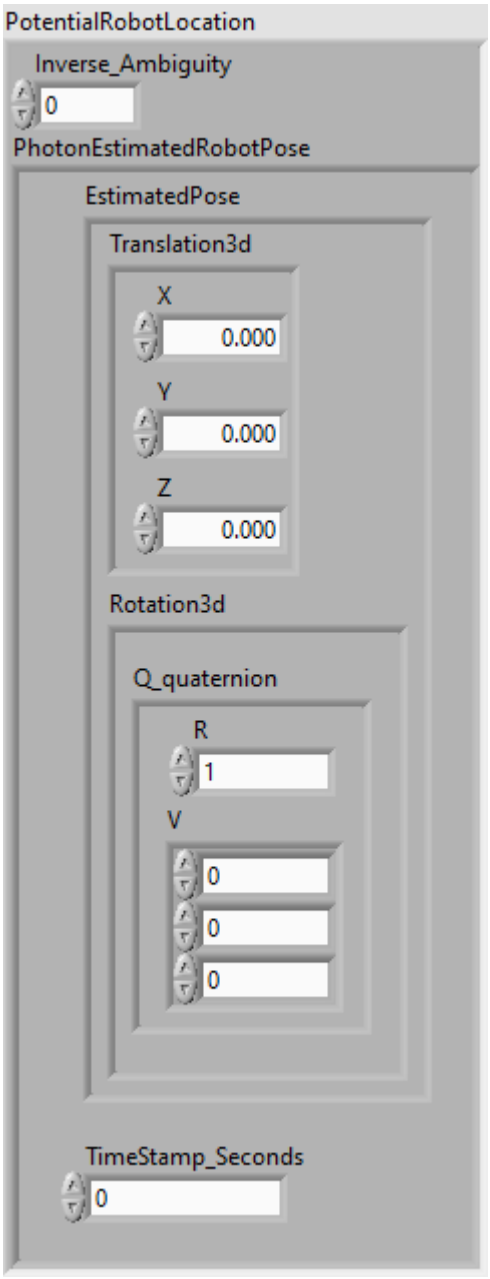
TypeDef-PhotonPoseEstimatorType

The PhotonPoseEstimator cluster contains data that filters or combines readings from all the fiducials visible at a given timestamp on the field to produce a single robot in field pose, using the strategy set below. Example usage can be found under LabVIEW Find Examples..

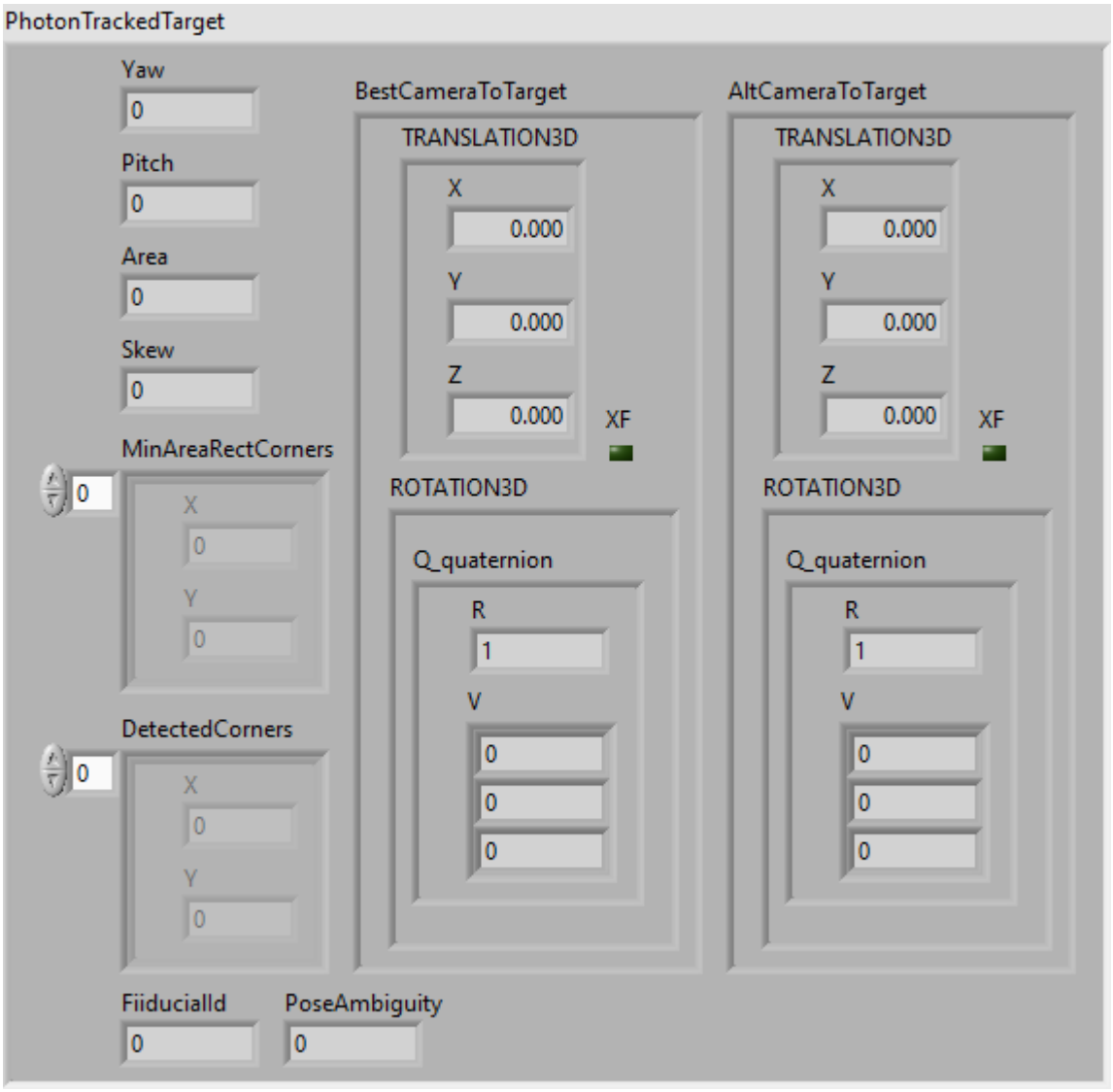


TypeDef-PhotonPoseEstimator_PotentialRobotLocation

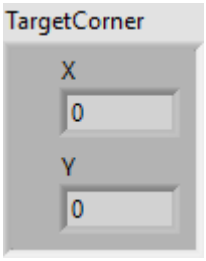
This data cluster is used internally by the PhotonPoseEstimator routines to compare individual potential positions.



TypeDef-PhotonTrackedTargetType



TypeDef-TargetCornerType



Enumerated Type Definitions

Enum

Enum-PhotonPoseStrategy_ENUM

Position estimation strategies that can be used by the PhotonPoseEstimator VIs.

Values are:

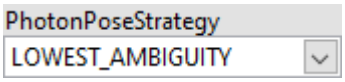
LOWEST_AMBIGUITY -- Choose the Pose with the lowest ambiguity

CLOSEST_TO_CAMERA_HEIGHT -- Choose the Pose which is closest to the camera height

CLOSEST_TO_REFERENCE_POSE -- Choose the Pose which is closest to a set Reference position

CLOSEST_TO_LAST_POSE -- Choose the Pose which is closest to the last pose calculated

AVERAGE_BEST_TARGETS -- Choose the Pose with the lowest ambiguity



Enum-VisionLEDModeType