PhotonVisionLib

LabVIEW

Reference

Version 2025_0

Table of Contents

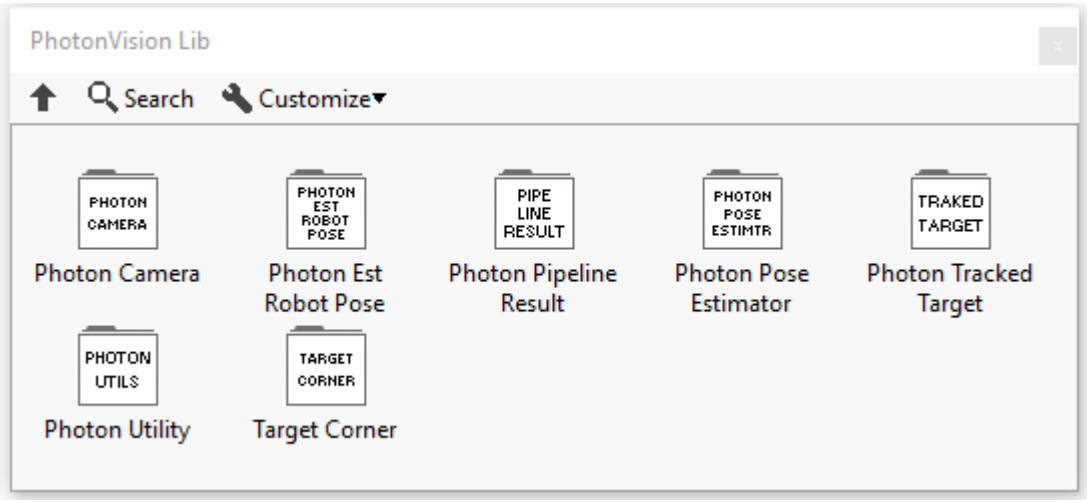
Introduction

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

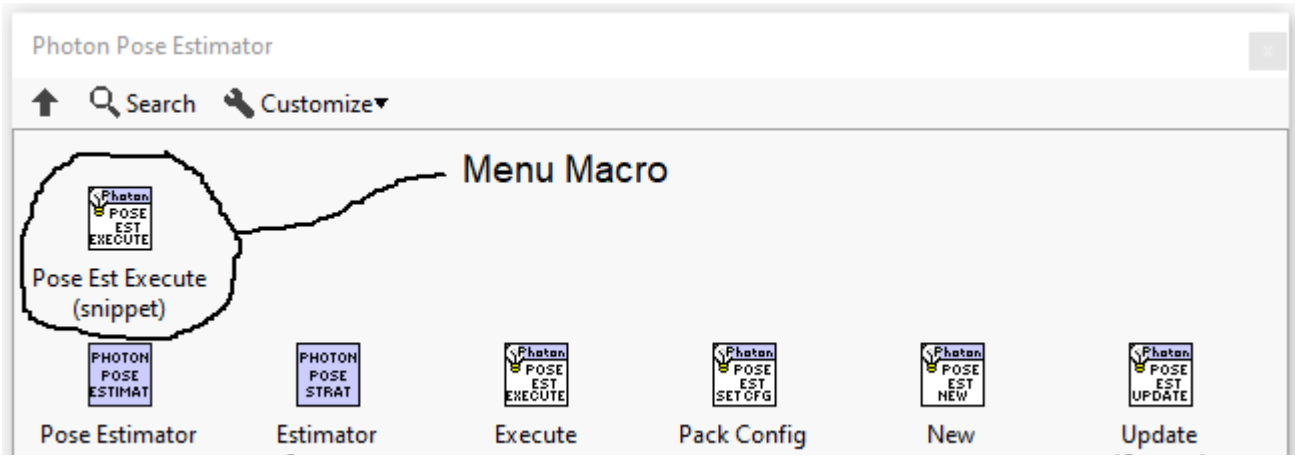
Function Menus

A PhotonVision function palette contains the PhotonVision functions and type definitions. This palette can be accessed from the WPI Robotics Library Third Party palette.



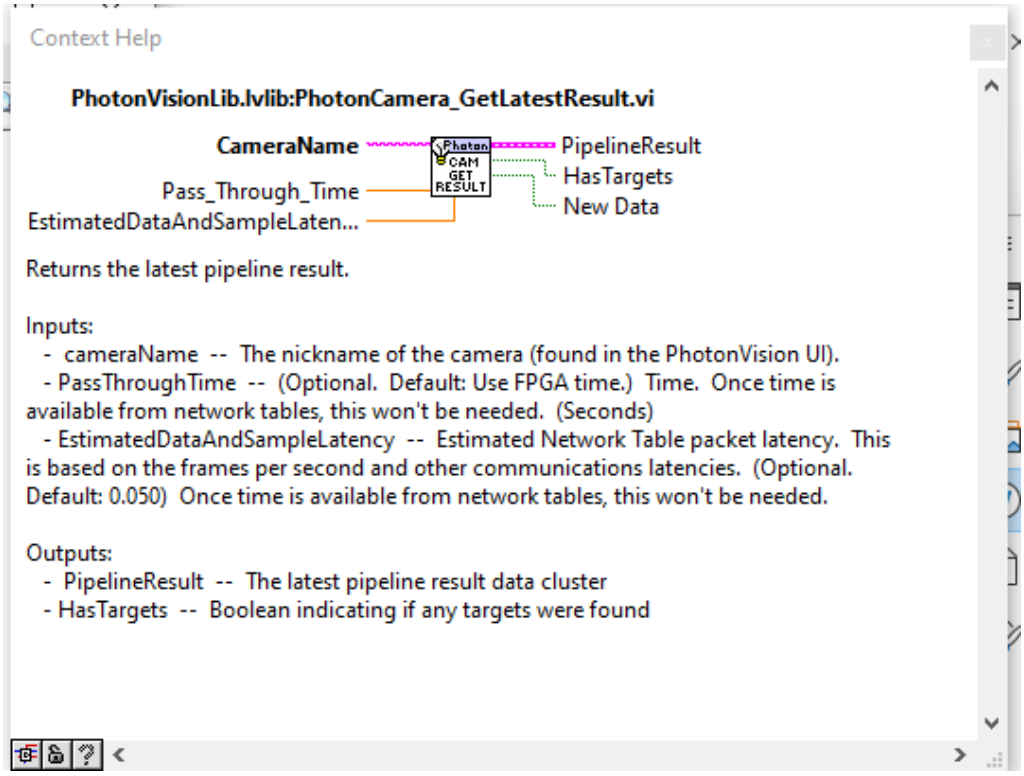
Menu Macros

Some of the menu items place “snippets” of code into a VI. These “macros” can greatly speed up development by placing large sections of mostly completed code in a VI. Usually macros have a different color menu palette icon and may contain “(snippet)” in the description.



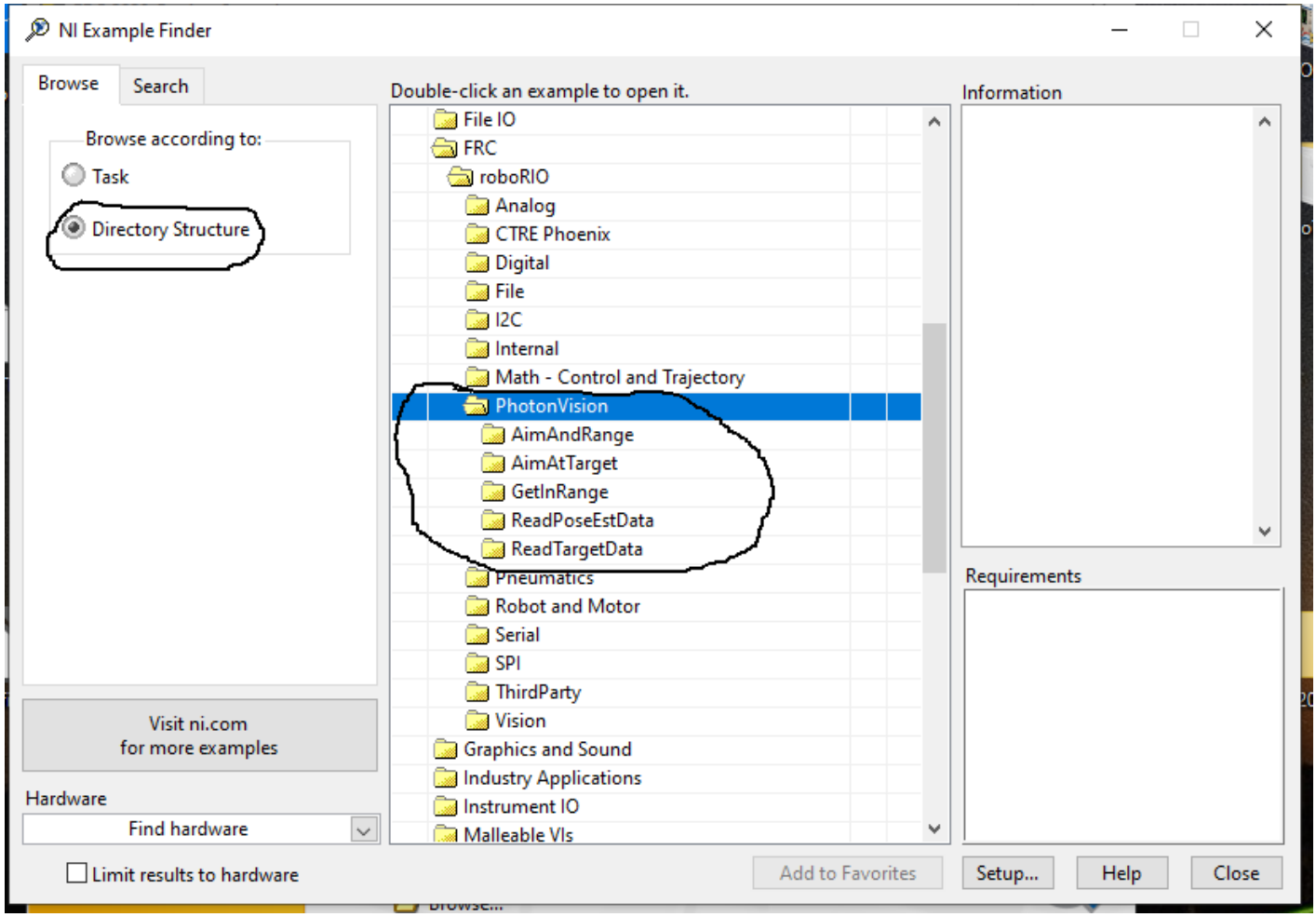
Function Help

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.



Function Groups

Packet

Packet_decode_Boolean

Decode (or strip out) a boolean 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. A TRUE boolean is stroed as an integer value = 1.

Inputs

- RawData -- Array of bytes containing the data to be extracted.
- StartByte -- The starting byte number (the first byte starts at 0)

Outputs

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

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_Flt32

Decode (or strip out) a single precision floating point value FLT 32. 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

- FLT32_Value -- Extracted float 32 (single float) 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_Int16

Decode (or strip out) a 16 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

- INT16_Value -- Extracted 16 bit integer 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_Int64

Decode (or strip out) a 64 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

- INT64_Value -- Extracted 64 bit integer value.
- NextByte -- The starting byte index for the next value (regardless of type).

Packet_decode_TargetCornerArray

Decode (or strip out) aTarget Corner Array 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

- Target Corner Array -- Extracted Array of Target Corner cluster.

- 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_Boolean

Encode (or append) a boolean to the end of an array of bytes. The boolean is stored as an unsigned integer byte, where TRUE = 1, FALSE = 0.

Inputs

- RawData In -- Array of bytes to be appended to..
- Boolean_Value -- Boolean value.

Outputs

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

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_FLT32

Encode (or append) a single floating point value (FLT 32) 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..
- FLT32_Value -- Single floating point 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_Int16

Encode (or append) a 16 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..
- Int16_Value -- 16 bit integer the form of an INT 16 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_Int64

Encode (or append) a 64 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..
- In64_Value -- 64 bit integer the form of an INT 64 value.

Outputs

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

Packet_encode_TargetCornerArray

Encode (or append) a Target Corner Array 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..
- Target Corner Array -- Array of Target Corner clusters to be encoded.

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_ChkTimeSync

Check Time Sync. If server has not been running, then signal this to the user. Write a message to the driver station error message.

Inputs:

- PhotonPipelineResult -- Cluster containing photon pipeline result.

- Pass_Through_Time -- Optional. Current roboRIO FPGA time. If not provided, this VI will read the time.

Outputs:

- TimeNotSynced -- TRUE if time sync server has NOT been functioning correctly.

PhotonCamera_GetCameraMatrix

Returns the camera matrix

Inputs:

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

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- CameraMatrix - matrix - 3 x 3 matrix of camera intrinsics. If empty, nothing was returned or nothing exists.

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetDistCoeffs

Returns the distortion coeeficitions

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- DistCoeffs - matrix - 5 x 1 matrix of distortion coefficients. If empty, nothing was returned or nothing exists.
- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetDriverMode

Returns whether the camera is in driver mode.

Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

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

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetLEDMode

Returns the current LED mode.

Inputs:

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- LedMode -- The current LED mode.
- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetLatestResult

Returns the latest pipeline result.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)
- 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

- New Data -- Boolean indicating if response is new.

- Error Out -- Contains error information if a Network table ot other error occured.

- Parsing Error -- Boolean indicating that number of bytes processes <> number of packet bytes

---

PhotonCamera_GetLatestResult_OLD

Returns the latest pipeline result.

Inputs:

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

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

- 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

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetPipelineIndex

Returns the active pipeline index.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- PipelineIndex -- Active Pipeline Index
- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_GetVersion

Returned photonvision version string

Inputs:

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- Version - string - Photonvision version string
- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_IsConnected

Determines if communications to camera is valid.

Inputs:

- CameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- IsConnected -- TRUE if communications have been received within the last 0.5 seconds.
- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_SetDriverMode

Toggles driver mode.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)
- DriverMode -- Whether to set driver mode.

Outputs:

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_SetLEDMode

Sets the LED mode.

Inputs:

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)
- LedMod -- The mode to set to.

Outputs:

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_SetPipelineIndex

Allows the user to select the active pipeline index.

Inputs:

- cameraName -- The nickname of the camera (found in the PhotonVision UI).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)
- index The active pipeline index.

Outputs:

- Error Out -- Contains error information if a Network table ot other error occured.

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).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- Error Out -- Contains error information if a Network table ot other error occured.

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).
- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- Error Out -- Contains error information if a Network table ot other error occured.

PhotonCamera_VerifyVersion

Verify photonvision version string

Currently this does not write a driver station message if the version is wrong.

Inputs:

- BaseNTName - string - Base Nework Table name. (Optional. Default: /photonvision)

Outputs:

- VersionOk - string - Version matches co-processor.
- Error Out -- Contains error information if a Network table ot other error occured.

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.

-- TargetsUsed -- array -- Array of targets used for pose estimation

-- StrategyUsed -- enum -- Strategy used for pose estimation.

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.

-- TargetsUsed -- array -- A list of the targets used to compute this pose

-- Strrategy -- enum -- The strategy actually used to produce this pose

Outputs:

- PhotonEstimatedRobotPose -- Created data cluster..

PhotonMultiTargetPNPResult

PhotonMultiTargetPNPResult_Equals

Determines if two PhotonMultiTargetPNPResult clusters are equal.

Inputs:

- PhotonMultiTargetPNPResult -- PhotonMultiTargetPNPResult cluster.
- OtherPhotonMultiTargetPNPResult -- OtherPhotonMultiTargetPNPResult cluster.

Outputs:

- Equal -- Returns TRUE, if both PhotonMultiTargetPNPResult clusters are equal..

PhotonMultiTargetPNPResult_MAX_IDS

Seeing 32 apriltags at once seems like a sane limit. (Current value is 32.)

PhotonMultiTargetPNPResult_New

Creaate an empty Multi Target PNP result

Inputs:

- PhotonPNPResult - cluster - Result to store in this mutilTarget results
- FiducialIDsUsed - Array - List of Fidicual IDS (april tags) used to locate this target.

Outputs:

- MultiTargetPNPResult - cluster - created result cluster.

PhotonMultiTargetPNPResult_New_Empty

Creaate an empty Multi Target PNP result

Inputs:

Outputs:

- MultiTargetPNPResult - cluster - created empty result cluster.

PhotonMultiTargetPNPResult_pack

Encode a MultiTargetPNPResult cluster into an array of bytes.

Inputs:

- MultiTagetPNPResult -- cluster containing the data to pack.

Outputs:

- RawDataOut -- Byte array containing the encoded data.

PhotonMultiTargetPNPResult_unpack

Internal function to parse the data for MultiTargetPNPResult

Inputs:

- RawData -- Byte array containing the data to parse.
- StartingByte - Index into RawData to start parsing. Beginning of data is index 0.

Outputs:

- MultiTargetPNPResult -- Parsed result cluster.
- NextByte -- Index to byte following ths data in RawData.

PhotonPipelineMetadata

PhotonPipelineMetadata_Equals

Determines if two PipelineMetadata clusters are equal.

Inputs:

- PipelineMetadata -- PipelineMetadata cluster.

- Other PipelineMetadata -- Other Pipelinemetadata cluster.

Outputs:

- Equal -- Returns TRUE, if both pipeline metadata clusters are equal..

PhotonPipelineMetadata_GetAll

Returns individual elements of PipelineMetadata cluster

Inputs:

- PipelineMetadata -- Parsed PipelineMetadata cluster.

Outputs:

- CaptureTimestampMicros -- Image capture timestamp, in microseconds

- PublishTimestampMicros -- NT publish timestamp, in microseconds

- SequenceID -- Mirror of the heartbeat entry -- monotonically increasing

- TimeSinceLastPong -- Time from last Time Sync Pong received and the construction of this metadata, in uS

PhotonPipelineMetadata_GetLatency

Returns the time between image capture and publish to NT

Inputs:

- PipelineMetadata -- Parsed PipelineMetadata cluster.

Outputs:

- Latency_ms -- Latency between cature and publish (milliseconds)

PhotonPipelineMetadata_New

Create new Pipeline MetaData cluster

Inputs:

- CaptureTimestampMicros -- Image capture timestamp, in microseconds
- PublishTimestampMicros -- NT publish timestamp, in microseconds
- SequenceID -- Mirror of the heartbeat entry -- monotonically increasing
- TimeSinceLastPong -- Time from last Time Sync Pong received and the construction of this metadata, in uS

Outputs:

- PipelineMetaData -- Parsed PipelineMetadata cluster.

PhotonPipelineMetadata_Pack

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

Inputs:

- PipelineMetadata -- PipelineMetadata cluster.

Outputs:

- Pacet_RawData -- Byte array containing the encoded PipelineResult.

PhotonPipelineMetadata_Unpack

Internal function to parse the data returned by PhotonCamera_GetLatestResult.

Inputs:

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

Outputs:

- PipelineMetadata -- Parsed PipelineMetadata cluster.

PhotonPipelineResult

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_GetMetadata

Returns the metadata in this pipeline result.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- Metadata -- Metadata cluster.

PhotonPipelineResult_GetMultiTagResult

Returns the MultiTargetPNPREsultin this pipeline result. Be sure to check IsPresent before using the pose estimate!

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- MultiTargetPNPResult-- Cluster containing the multitarget PNP result
- IsPresent - boolean - TRUE if a MultiTarget PNP result contains a pose.

PhotonPipelineResult_GetTargetCount

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_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 the estimated time the frame was taken, in the Time Sync Server's time base (nt::Now). This is calculated using the estiamted offset between Time Sync Server time and local time. The robot shall run a server, so the offset shall be 0.

Inputs:

- PipelineResult -- Parsed PipelineResult cluster.

Outputs:

- TimeStampSec -- The timestamp in 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:

- Metadata -- Cluster containing pipeline meta-data
- PhotonTrackedTargets -- Array of tracked targets. (could be empty)
- PhotonMultiTargetPNPResult - Multi Target Result.

Outputs:

- PipelineResult -- Parsed PipelineResult cluster.

PhotonPipelineResult_NewIndividual

Create a new PhotonPipelineResult cluster.

Inputs:

- CaptureTimestampMicros -- Image capture timestamp, in microseconds
- PublishTimestampMicros -- NT publish timestamp, in microseconds
- SequenceID -- Mirror of the heartbeat entry -- monotonically increasing
- TimeSinceLastPong -- Time from last Time Sync Pong received and the construction of this metadata, in uS
- PhotonTrackedTargets -- Array of tracked targets. (could be empty)
- PhotonMultiTargetPNPResult - Multi Target Result.

Outputs:

- PipelineResult -- Parsed PipelineResult cluster.

PhotonPipelineResult_Pack

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.

PhotonPipelineResult_Unpack

Internal function to parse the data returned by PhotonCamera_GetLatestResult.

Inputs:

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

Outputs:

- PipelineResult -- Parsed PipelineResult cluster.
- Parsing Error -- Boolean indicating that number of bytes processes <> number of packet bytes

PhotonPNPResult

PhotonPNPResult_Equals

Determines if two PhotonPNPResult clusters are equal.

Inputs:

- PhotonPNPResult -- PhotonPNPResult cluster.
- OtherPhotonPNPResult -- OtherPhotonPNPResult cluster.

Outputs:

- Equal -- Returns TRUE, if both PhotonMultiTargetPNPResult clusters are equal..

PhotonPNPResult_New

Creates a PNP result with both "best" and "alt" result.

Inputs:

- best - transform3d - Best result
- alt - transform3d - Alternate result
- ambiguity - double - result ambiguity
- bestReprojErr - double - best error (pixels)
- AltReprojErr - double - alt error (pixels)

Outputs:

- PNP_Result - cluster - created cluster.

PhotonPNPResult_New_Empty

Returns an empty (invalid) result.

Inputs:

Outputs:

- PNP_Result - cluster - created cluster.

PhotonPNPResult_New_OnlyBest

Returns only the BEST result.

Inputs:

- best - transform3d - Best result
- bestReprojErr - double - best error (pixels)

Outputs:

- PNP_Result - cluster - created cluster.

PhotonPNPResult_pack

Encode a PNP Result cluster into an array of bytes.

Inputs:

- PNP_Result -- PNP_Result cluster.

Outputs:

- RawDataOut -- Byte array containing the encoded data.

PhotonPNPResult_pack_OLD

Encode a PNP Result cluster into an array of bytes.

Inputs:

- PNP_Result -- PNP_Result cluster.

Outputs:

- RawDataOut -- Byte array containing the encoded data.

PhotonPNPResult_unpack

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

Inputs:

- RawData -- Byte array containing the data to parse.
- StartingByte -- Index into raw data to start parsing. First byte is index 0.

Outputs:

- PNP_Result -- Parsed PNP_Result cluster.

PhotonPNPResult_unpack_OLD

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

Inputs:

- RawData -- Byte array containing the data to parse.
- StartingByte -- Index into raw data to start parsing. First byte is index 0.

Outputs:

- PNP_Result -- Parsed PNP_Result 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_Execute

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.

-- AprilTagFieldLayout -- data cluster -- Field layout defining the location of April Tags

Outputs:

- EstimatedPose3d - Pose3d - Estimated 3d position. Only value if NewPoseFound = TRUE
- EstimatedPose2d - Pose2d - Estimated 2d position. Only value if NewPoseFound = TRUE
- NewPoseFound -- boolean -- A new position has been determined.
- TimeStampe - double - Target timestamp (seconds)

- StrategyUsed -- enum -- The strategy used to determine the pose.

- PhotonEstimatedRobotPose -- data cluster -- Complete estimated robot position and timestamp data cluster

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

PhotonPoseEstimator_Execute_OLD

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:

- EstimatedPose3d - Pose3d - Estimated 3d position. Only value if NewPoseFound = TRUE

- EstimatedPose2d - Pose2d - Estimated 2d position. Only value if NewPoseFound = TRUE

- NewPoseFound -- boolean -- A new position has been determined.

- TimeStampe - double - Target timestamp (seconds)

- StrategyUsed -- enum -- The strategy used to determine the pose.

- PhotonEstimatedRobotPose -- data cluster -- Complete estimated robot position and timestamp data cluster

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

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_GetRobotToCameraTransform

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_GetTagModel

Get the current tag model used by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- TagModel -- cluster -- Current tag model.

PhotonPoseEstimator_InvalidatePoseCache

Invalidate Pose Cache.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

Outputs:

-- PhotonVisionPoseEstimator -- cluster -- The updated 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_PackConfig

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:

- BaseTNName -- string - Base network tables name. Default: /photonvision

- Camera -- string - Camera name

- Primary Strategy -- enum - Primary estimation strategy
- HighestAmbiguity -- double -- Highest permitted ambiguity value
- RobotToCamera -- Transform3d -- Position of camera on robot
- InitialPose -- Pose3d - Initial location of robot on the field.
- Fallback Strategy -- enum - Strategy to use when primary strategy fails.

Outputs:

-- PhotonPoseEstimator _Configuration -- data cluster -- Pose Estimator Configuration data

---

PhotonPoseEstimator_SetCamera

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_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_SetMultiTagFallbackStrategy

Set the current fallback strategy being used by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- Fallback Strategy -- 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_SetPrimaryStategy

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_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_SetRobotToCameraTransform

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_SetTagModel

Set the current tag model used by the PhotonPoseEstimator.

Inputs:

-- PhotonVisionPoseEstimator -- cluster -- The estimator.

-- TagModel -- cluster -- Current April Tag model used by the Photon Pose Estimator.

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:

- New data has not been received since the last call to {@code update()}.
- No targets were found from the camera
- There is no camera set

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 with an estimated pose, timestamp, and targets used to create the estimate.

-- 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_All

Provided data from polling a camera, updates the estimated position of the robot. Returns empty if:

- New data has not been received since the last call to update.
- No targets were found from the camera
- There is no camera set

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.

-- cameraMatrix -- matrix -- Camera calibration data that can be used in the case of no assigned PhotonCamera.

-- distCoeffs -- matrix -- Camera calibration data that can be used in the case of no assigned PhotonCamera

Outputs:

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

-- PhotonEstimatedRobotPose -- data cluster -- Estimated robot position, timestamp, and targets used to create the estimte.

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

PhotonPoseEstimator_Update_PipeResult

Provided data from polling a camera, updates the estimated position of the robot. Returns empty if:

- The timestamp of the provided pipeline result is the same as in the previous call to update
- No targets were found in the pipeline results.
- No camera is set.

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, timestamp, , and targets used to create the estimate.

-- 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.

PhotonPoseEstimator_multiTagOnCoprocStrategy

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

Determine the best target based the multi-target routine on the coprocessor

PhotonTargetModel

PhotonTargetModel_New

Creates a rectangular, planar target model given the verticies and type information..

Inputs:

- veerticies -- Translation3d array -
- IsPlanar - boolean
- IsSpherical - boolean

Outputs:

- TargetModel - cluster - created target model

PhotonTargetModel_New_AprilTag16h5

Create target model for 16h5 April Tag

PhotonTargetModel_New_AprilTag36H11

Create target model for 36h11 April Tag

PhotonTargetModel_New_RectPlanar

Creates a rectangular, planar target model given the width and height. The model has four vertices:

- Point 0: [0, -width/2, -height/2]
- Point 1: [0, width/2, -height/2]
- Point 2: [0, width/2, height/2]
- Point 3: [0, -width/2, height/2]

Inputs:

- widthMeters - double - Width (meters)
- HeightMeters - double - Height (meters)

Outputs:

- TargetModel - cluster - created target model

PhotonTrackedTarget

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

For fiducials, the order is known and is always counter-clock wise around the tag, like so:

```
  -> +X     3 ----- 2
  |         |       |
 V          |       |
```

+Y 0 ----- 1

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

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

---

PhotonTrackedTarget_GetDetectedObjectClassID

Get the object detection class ID number, or -1 if not set.

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- DetectedObjectClassID

PhotonTrackedTarget_GetDetectedObjectConfidence

Get the object detection confidence, or -1 if not set. This will be between 0 and 1, with 1 indicating most confidence, and 0 least.

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- DetectedObjectConfidence

PhotonTrackedTarget_GetFiducialId

Get Fidicial ID (April Tag value) from Tracked Target

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- Fiducial ID

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 the ratio of best:alternate pose reprojection errors, called ambiguity. This is betweeen 0 and 1 (0 being no ambiguity, and 1 meaning both have the same reprojection error). Numbers above 0.2 are likely to be ambiguous. -1 if invalid.

Inputs:

- TrackedTarget -- Parsed TrackedTarget cluster.

Outputs:

- PoseAmbiguity - double - Get the ratio of pose reprojection errors, called ambiguity. Numbers above 0.2 are likely to be ambiguous. -1 if invalid.

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_Pack

Encode a TrackedTarget cluster into an array of bytes.

Inputs:

- TrackedTarget -- TrackedTarget cluster.

Outputs:

- RawDataOut -- Byte array containing the encoded data.

PhotonTrackedTarget_Unpack

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.

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 to the pose.

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

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:

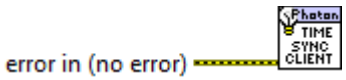
- X --
- Y --

Outputs:

- TargetCorner -- Created TargetCorner cluster.

TimeSyncClient

TimeSyncClient_Exec



Execute a PhotonVision Time Sync Client. Once a second it sends a UDP broadcast ping and listens for a response. If a good response is received, an FPGA time offset is calculated.

This VI contains a loop. Do not place this in another loop.

This routine is for diagnostic purposes. Robot code does NOT need this routine.

Inputs:

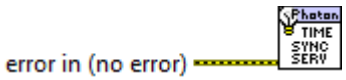
-- Error In -- Optional error input cluster.

Outputs:

--

TimeSyncServ

TimeSyncServ_Exec



Execute a PhotonVision Time Sync Server. It listens for UDP requests from PhotonVision devices and responds with the current FPGA time.

This VI contains a loop. Do not place this in another loop.

Place this all by itself in Periodic Tasks.

Inputs:

-- Error In -- Optional error input cluster.

Outputs:

--

TimeSyncServ_GetAll

Get information on photonvision time sync server.

Inputs:

-

Outputs:

-

Type Definitions

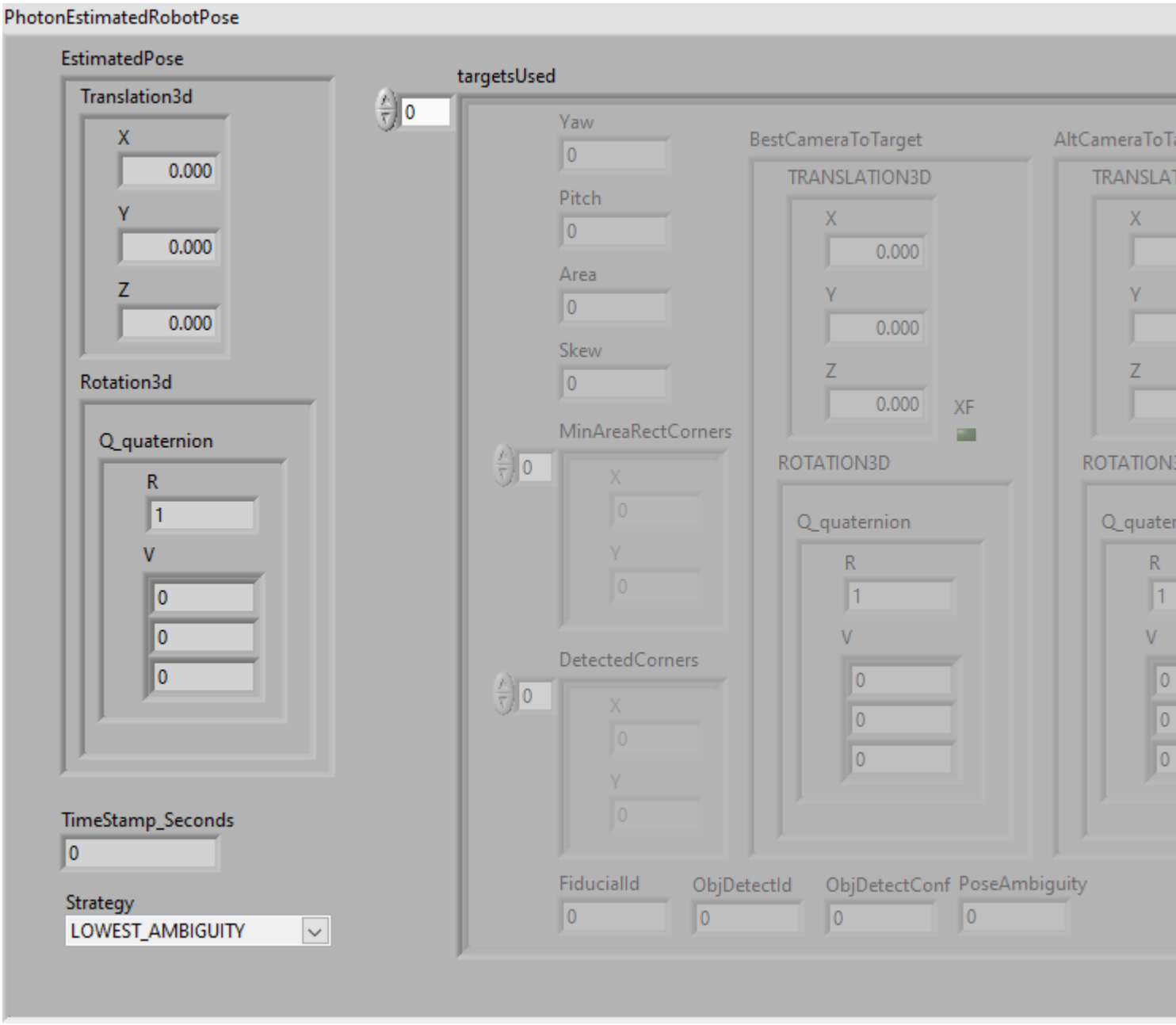
TypeDef

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-PhotonMultiTargetPNPResultType

The best estimated transformation from solvePnP, and possibly an alternate transformation depending on the solvePNP method. If an alternate solution is present, the ambiguity value represents the ratio of reprojection error in the best solution to the alternate (best / alternate).

Note that the coordinate frame of these transforms depends on the implementing solvePnP method.

Contains:

- isPresent - boolean - If this result is valid. A false value indicates there was an error in estimation, and this result should not be used.

- best - Transform3d - The best-fit transform. The coordinate frame of this transform depends on the method which gave this result.

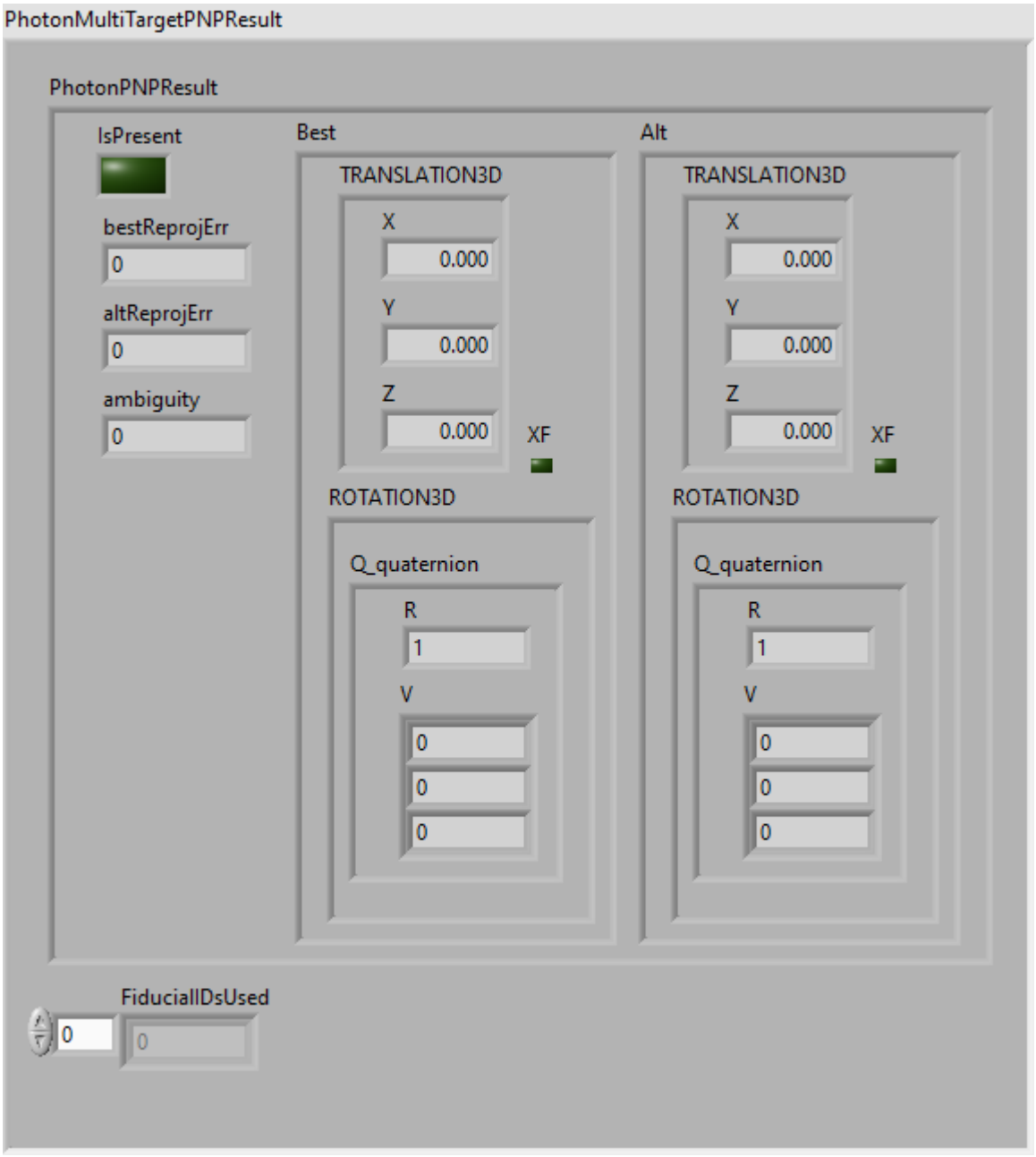
- bestReprojErr - double - Reprojection error of the best solution, in pixels

- alt - Transform3d - Alternate, ambiguous solution from solvepnp. If no alternate solution is found, this is equal to the best solution.

- altReprojErr - double - If no alternate solution is found, this is bestReprojErr

- ambiguity - double - If no alternate solution is found, this is 0

- FidudicalIDsUsed - array - List of fiducial IDs used to determine multi target result.



TypeDef-PhotonPNPResultType

The best estimated transformation from solvePnP, and possibly an alternate transformation depending on the solvePNP method. If an alternate solution is present, the ambiguity value represents the ratio of reprojection error in the best solution to the alternate (best / alternate).

Note that the coordinate frame of these transforms depends on the implementing solvePnP method.

Contains:

- isPresent - boolean - If this result is valid. A false value indicates there was an error in estimation, and this result should not be used.

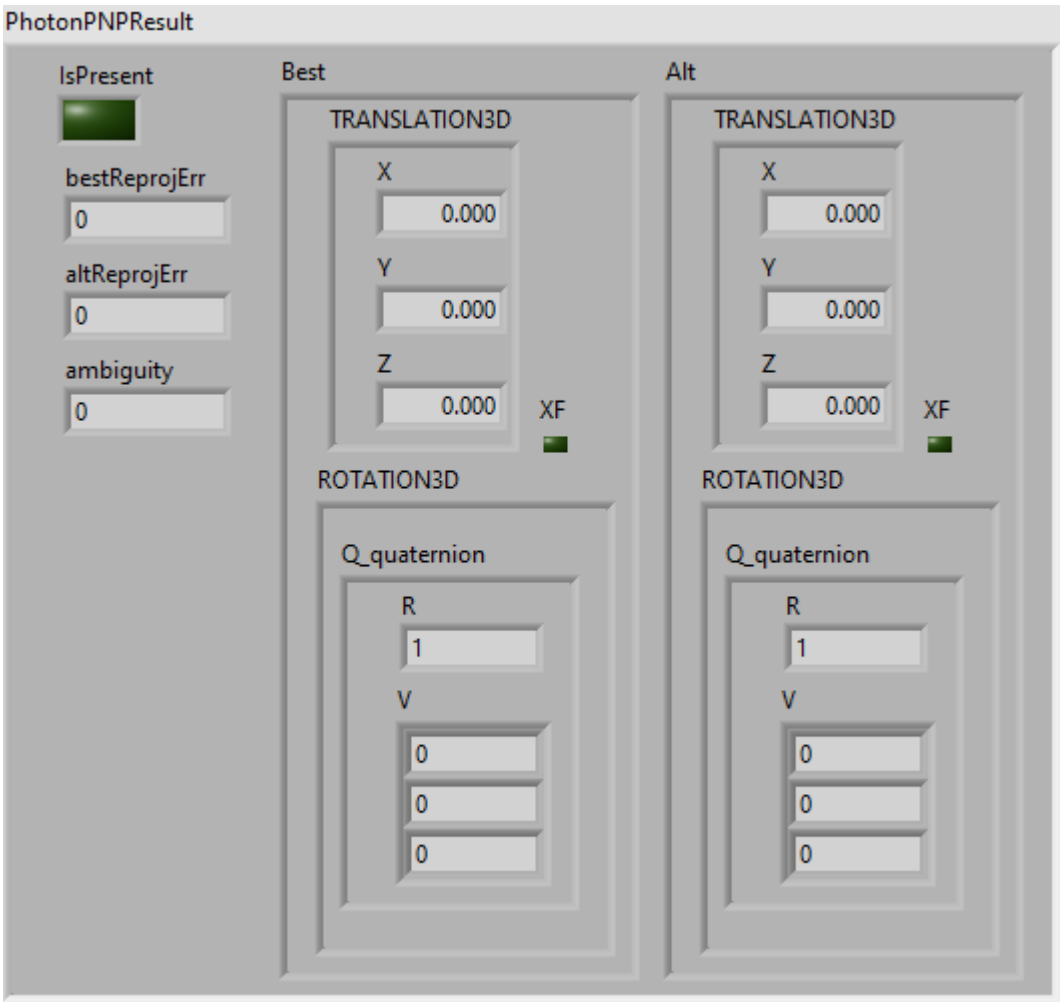
- best - Transform3d - The best-fit transform. The coordinate frame of this transform depends on the method which gave this result.

- bestReprojErr - double - Reprojection error of the best solution, in pixels

- alt - Transform3d - Alternate, ambiguous solution from solvepnp. If no alternate solution is found, this is equal to the best solution.

- altReprojErr - double - If no alternate solution is found, this is bestReprojErr

- ambiguity - double - If no alternate solution is found, this is 0

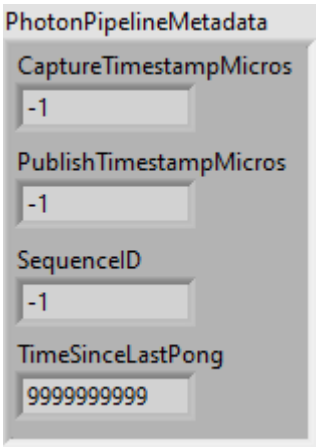


TypeDef-PhotonPipelineMetadataType

Represents a pipeline metadata

Contains:

- CaptureTimestampMicros -- Image capture timestamp, in microseconds
- PublishTimestampMicros -- NT publish timestamp, in microseconds
- SequenceID -- Mirror of the heartbeat entry -- monotonically increasing
- TimeSinceLastPong -- Time from last Time Sync Pong received and the construction of this metadata, in uS

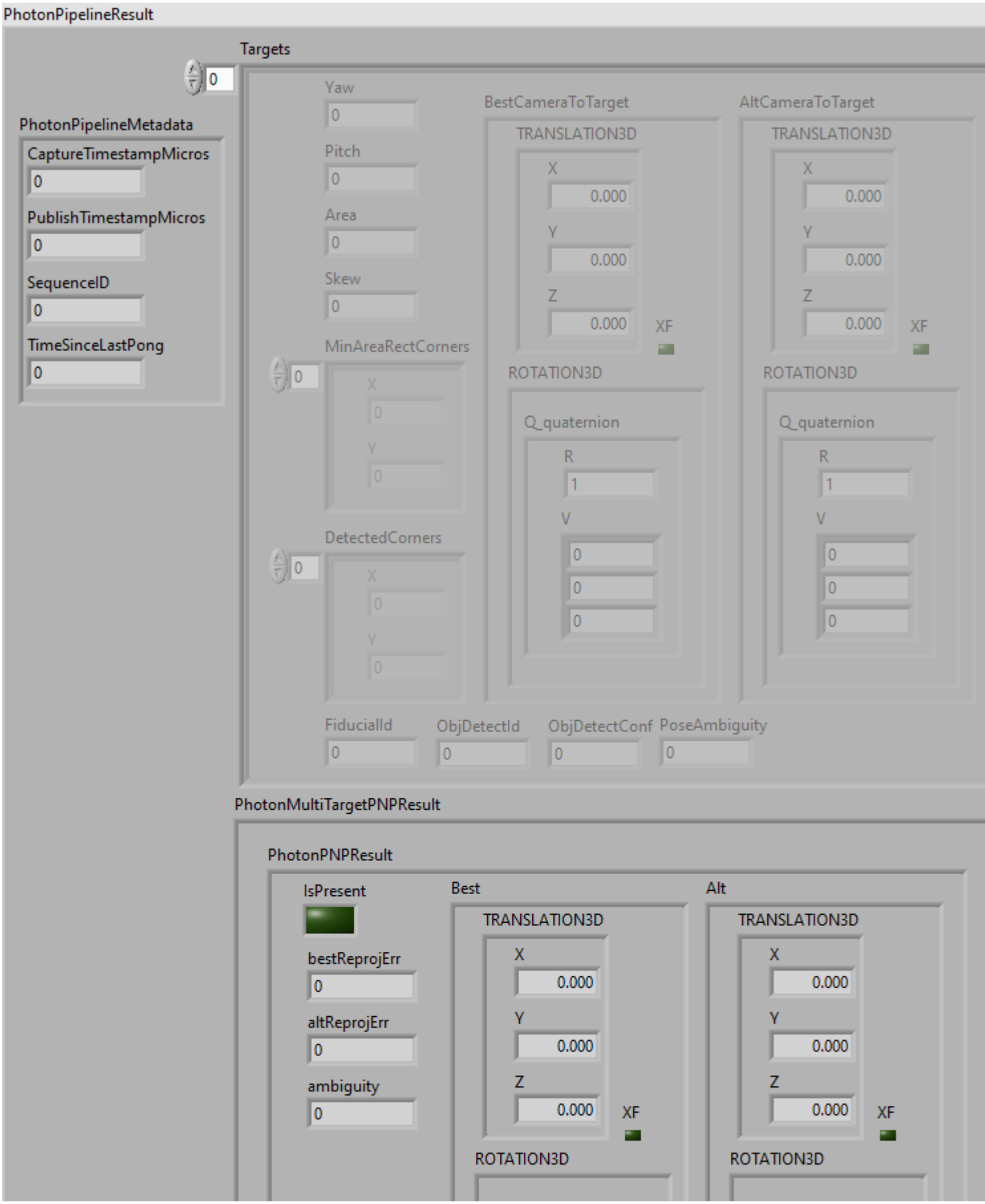


TypeDef-PhotonPipelineResultType

Represents a pipeline result from a PhotonCamera.

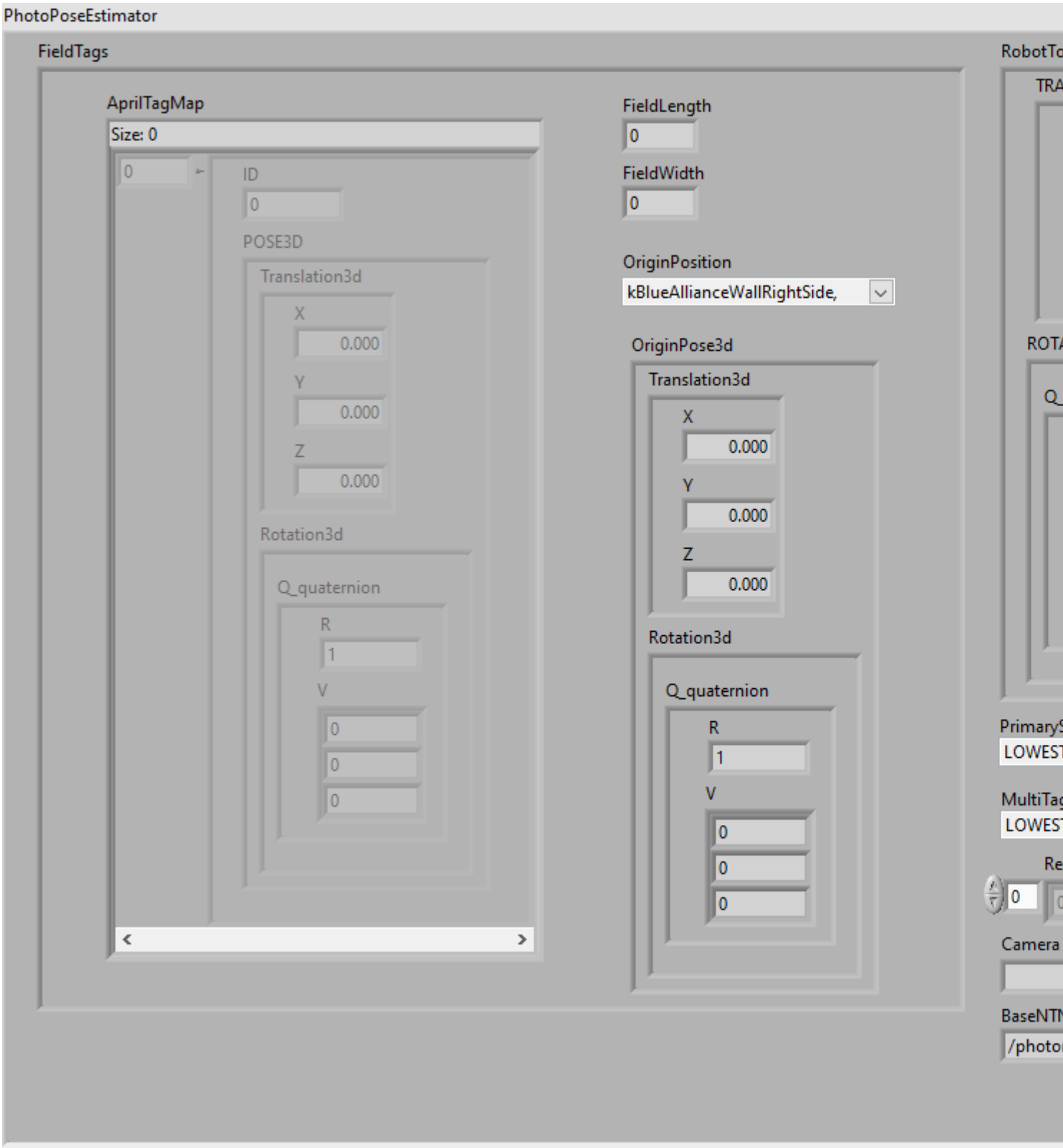
Contains:

- targets - array - Targets to store.
- latencyMillis - double - Latency in milliseconds.
- timestampSeconds - double - Timestamp in milliseconds.
- MultiTargetPNPResult - cluster - Multi-tag result


PhotonPipelineResult
Targets
0
PhotonPipelineMetadata
CaptureTimestampMicros
0
PublishTimestampMicros
0
SequenceID
0
TimeSinceLastPong
0
Yaw
0
Pitch
0
Area
0
Skew
0
MinAreaRectCorners
0
X
0
Y
0
DetectedCorners
0
X
0
Y
0
BestCameraToTarget
TRANSLATION3D
X
0.000
Y
0.000
Z
0.000
XF
ROTATION3D
Q_quaternion
R
1
V
0
0
0
AltCameraToTarget
TRANSLATION3D
X
0.000
Y
0.000
Z
0.000
XF
ROTATION3D
Q_quaternion
R
1
V
0
0
0
FiducialId
0
ObjDetectId
0
ObjDetectConf
0
PoseAmbiguity
0
PhotonMultiTargetPNPResult
PhotonPNPResult
IsPresent
bestReprojErr
0
altReprojErr
0
ambiguity
0
Best
TRANSLATION3D
X
0.000
Y
0.000
Z
0.000
XF
ROTATION3D
Alt
TRANSLATION3D
X
0.000
Y
0.000
Z
0.000
XF
ROTATION3D

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..

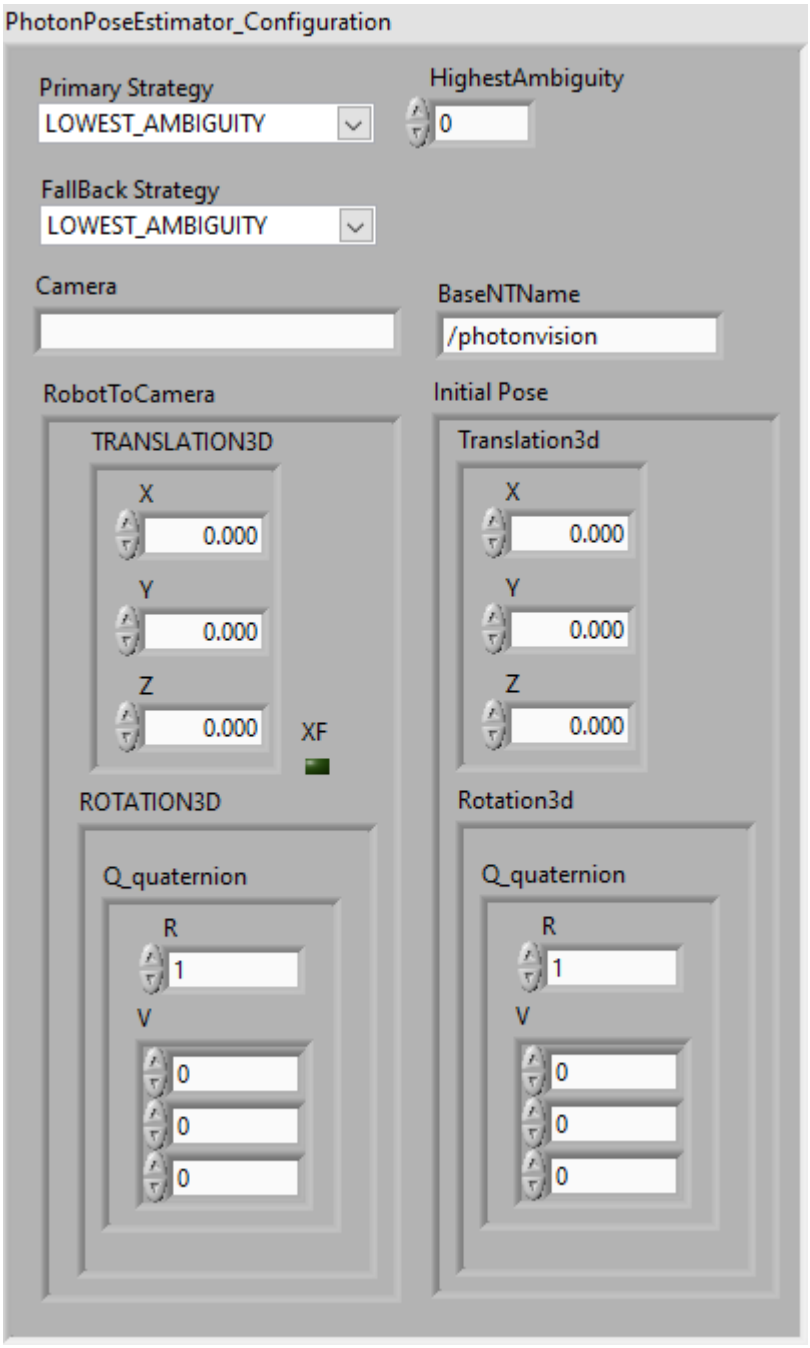

PhotoPoseEstimator
FieldTags
AprilTagMap
Size: 0
0
ID
0
POSE3D
Translation3d
X
0.000
Y
0.000
Z
0.000
Rotation3d
Q_quaternion
R
1
V
0
0
0
FieldLength
0
FieldWidth
0
OriginPosition
kBlueAllianceWallRightSide,
OriginPose3d
Translation3d
X
0.000
Y
0.000
Z
0.000
Rotation3d
Q_quaternion
R
1
V
0
0
0

TypeDef-PhotonPoseEstimator_Configuration

Stores the configuration information for PhotonPoseEstimator_Execute.

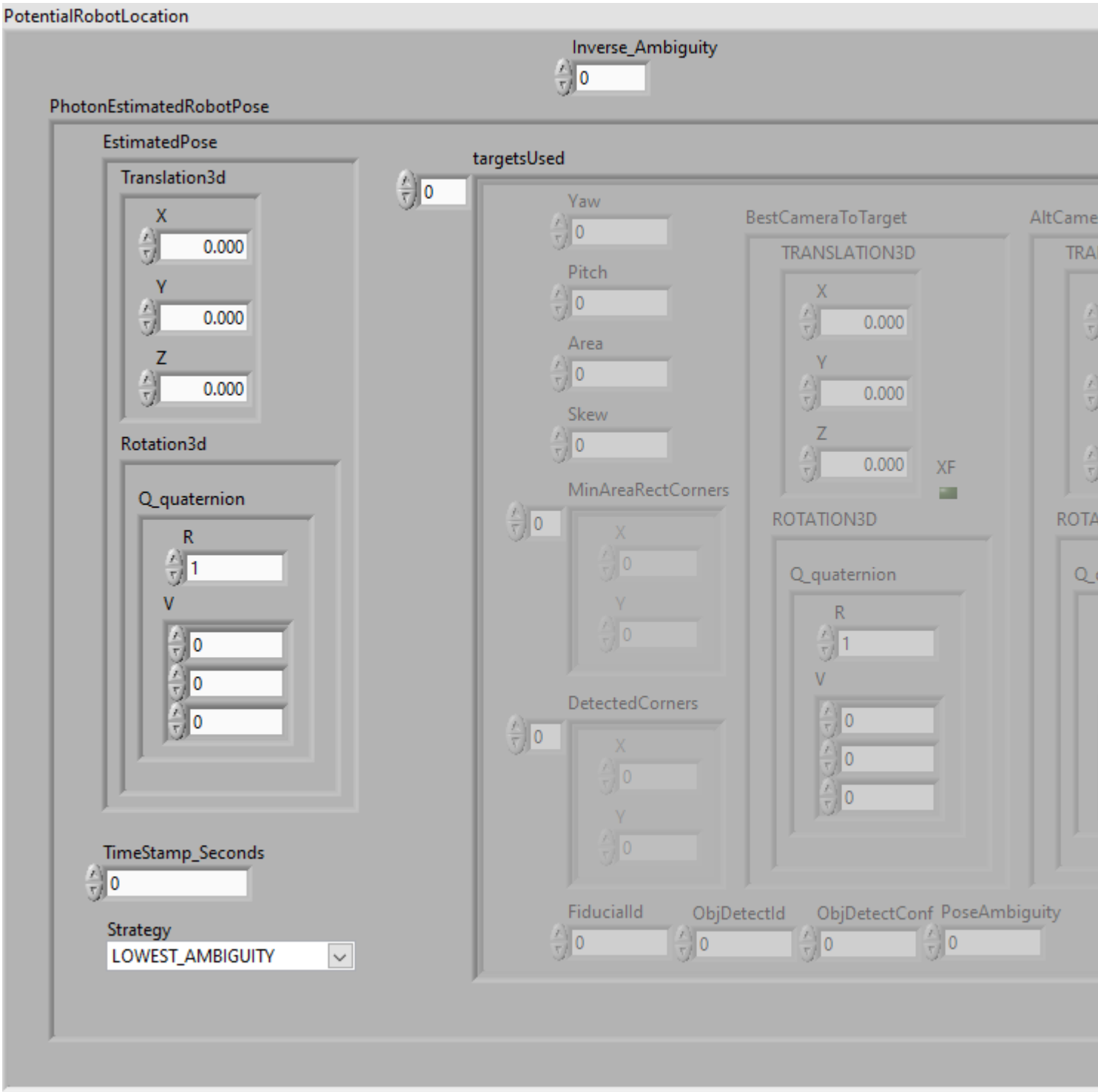
Contains:

- AprilTagField -- enum - Field selector
- UseCustomField -- boolean -- use a custom apriltag field instead of one of the predefined ones.
- Primary Strategy -- enum - Primary estimation strategy
- Fallback Strategy -- enum - Strategy to use when primary strategy fails.
- Highest Ambiguity -- double -- Highest allowed ambiguity.
- Camera -- string - Camera name
- BaseTNName -- string - Base network tables name. Default: /photonvision
- RobotToCamera -- Transform3d -- Position of camera on robot
- InitialPose -- Pose3d - Initial location of robot on the field.
- CustomAprilTagField -- cluster -- If a custom field is used, this defines the field.



TypeDef-PhotonPoseEstimator_PotentialRobotLocation

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



TypeDef-PhotonTargetModel

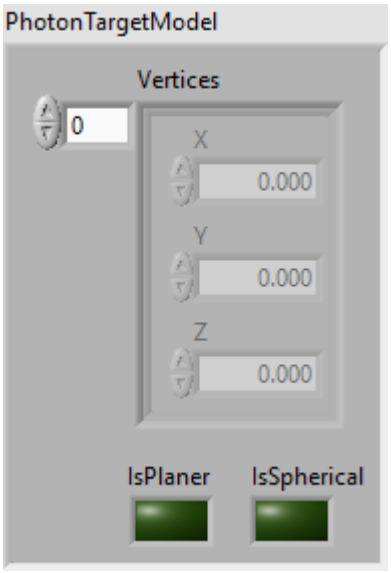
Describes the 3d model of a target.

Contains:

- vertices - Translation3d array - Translations of this target's vertices relative to its pose. Rectangular and spherical targets will have four vertices. See their respective constructors for more info.

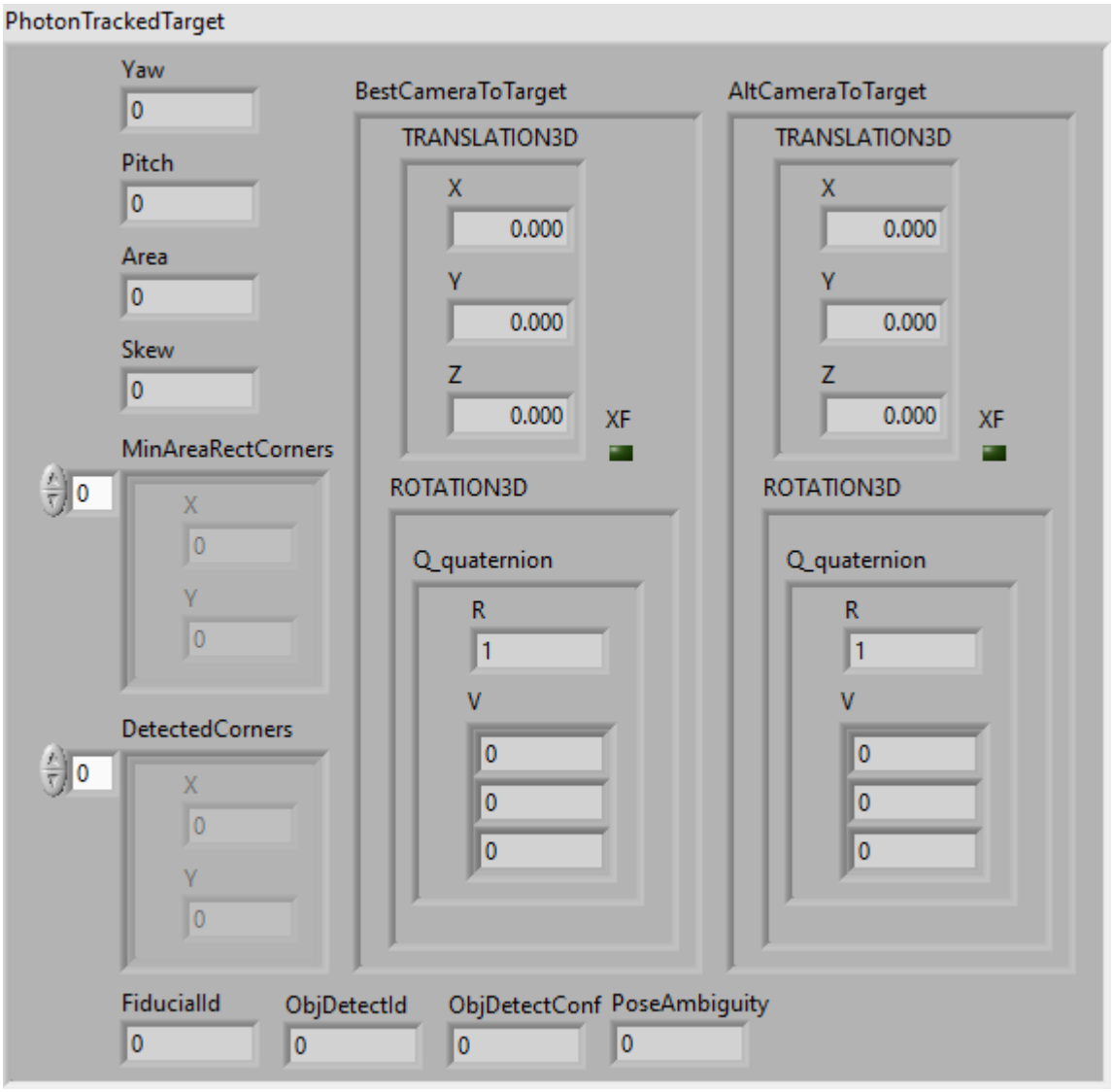
- isPlanar - boolean

- isSpherical - boolean



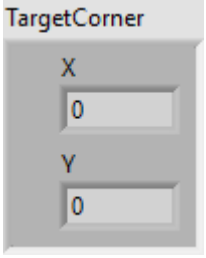
TypeDef-PhotonTrackedTargetType

Stores information about identified tracked targets.



TypeDef-TargetCornerType

Represents a point in an image at the corner of the minimum-area bounding rectangle, in pixels. Origin at the top left, plus-x to the right, plus-y down.



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

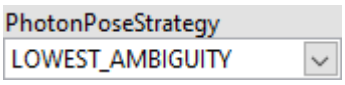
CLOSEST_TO_REFERENCE_POSE -- Choose the Pose which is closest to a set Reference position

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

MULTI_TAG_PNP_ON_COPROCESSOR -- Use all visible tags to compute a single pose estimate on coprocessor. This option needs to be enabled on the PhotonVision web UI as well.

MULTI_TAG_PNP_ON_RIO -- Use all visible tags to compute a single pose estimate. This runs on the RoboRIO, and can take a lot of time.



Enum-VisionLEDModeType

Enumerated variable type for LED mode

Selections are:

- Default
- Off
- On
- Blink