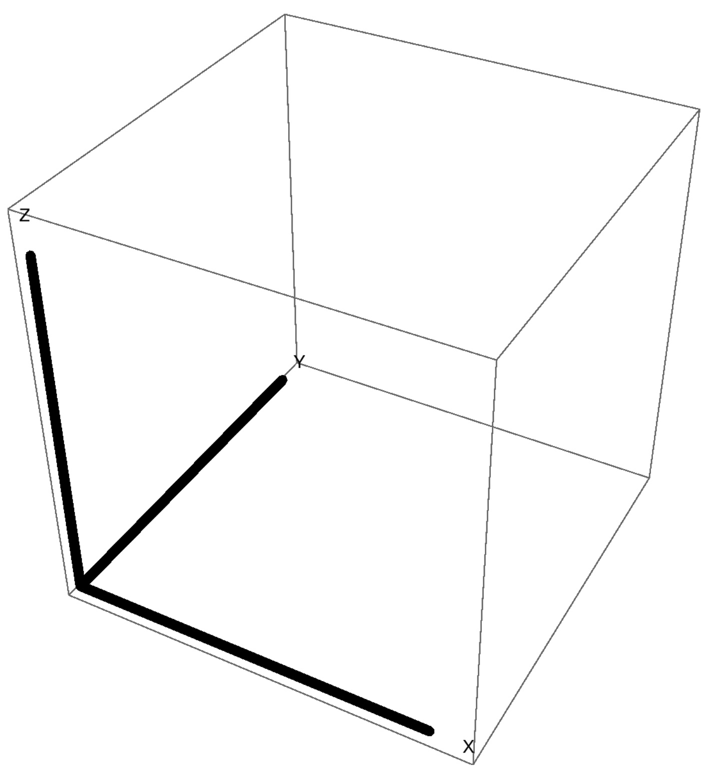
Vikram Voleti (20091845, University of Montreal) 3D Vision Term Project - Dec 2018

Importing stuff

In[1]:=
```
axes[m_, couleur_] := Module[{im, p, q, r}, (
    (* on utilise la transformation
     inverse pour aller du monde transformé vers le monde *)
    im = InverseFunction[m];
    (* p : le centre et les 3 axes x,y, et z *)
    p = {{0, 0, 0}, {5, 0, 0}, {0, 5, 0}, {0, 0, 5}};
    (* q : les points du systèmes d'axe exprimés dans vers le monde *)
    q = im /@ p;
    (* r : points plus éloignés pour afficher le texte *)
    r = im /@ ScalingTransform[{1.1, 1.1, 1.1}] /@ p;
    Graphics3D[{
      Thickness[0.015],
      couleur,
      Line[q[[{1, 2}]]], (* relier q[[1]] a q[[2]] *)
      Line[q[[{1, 3}]]],
      Line[q[[{1, 4}]]],
      Text["X", r[[2]]],
      Text["Y", r[[3]]],
      Text["Z", r[[4]]]
     }])];
axes[m_] := axes[m, Black];
axes[] := axes[Identity];
axes[]
```

Out[4]=



3D Points

In[440]:=
```
points3DX = RandomReal[{-100, 100}, {20, 1}];
points3DY = RandomReal[{-100, 100}, 20];
points3DZ = RandomReal[{100, 200}, 20];
points3D =
  MapThread[Append, {MapThread[Append, {points3DX, points3DY}], points3DZ}];
points3D // MatrixForm
```

Out[444]//MatrixForm=

-50.0761	82.9223	176.697
43.8513	-31.4558	114.289
-33.575	99.5875	170.74
25.6058	61.6854	179.754
-13.1533	-57.4739	175.976
-59.4381	-73.0332	179.02
82.4357	-59.6499	158.151
-25.9143	72.3303	117.533
80.439	-89.08	198.669
-85.0016	30.3223	119.332
-21.1333	-85.4791	197.819
14.8949	-43.1642	186.906
-62.8267	-59.9477	121.775
42.5604	19.978	107.691
72.2117	-41.288	111.06
-8.24741	-21.7607	119.251
39.7534	68.4113	130.405
-69.5375	-6.63656	138.006
-80.7541	73.414	129.324
-90.7416	-21.8206	117.092

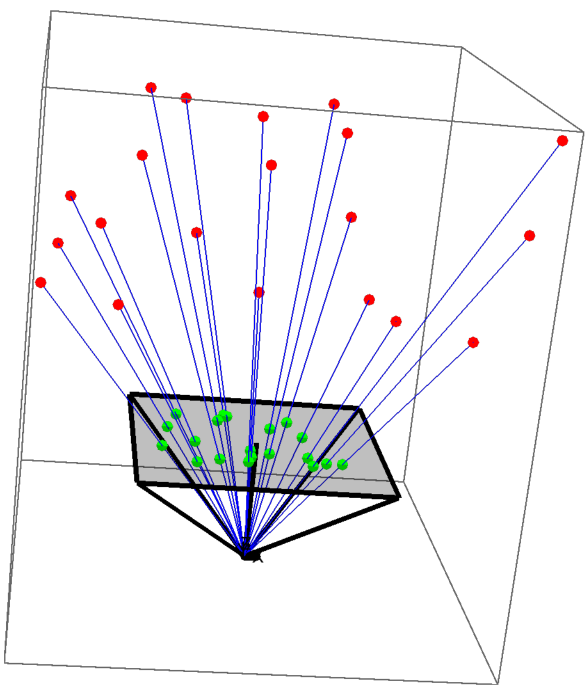
Camera

In[1236]:=
```
camera[imX_, imY_, aovDeg_, camPos_, camRotAngleDeg_, camRotAxis_] := Module[
   {aovDeg2f, fd, proj2Dvers3D, interne, mInt, mExt, m}, (
    aovDeg2f[aovD_] := Module[{}, (
       N[Max[imX, imY] / (2 * Tan[aovD / 2 * π / 180])]
      )];
    fd = aovDeg2f[aovDeg];
    proj2Dvers3D[m_] := AffineTransform[TransformationMatrix[m]];
    interne[fd_, cx_, cy_] := AffineTransform[{({{fd, 0}, {0, fd}}), ({{cx}, {cy}})}];
    mInt = proj2Dvers3D[interne[fd, imX / 2, imY / 2]];
    mExt = Composition[RotationTransform[-camRotAngleDeg * π / 180, camRotAxis],
      TranslationTransform[-camPos]];
    {mInt.mExt, fd, mInt[[1]][[1 ;; 3, 1 ;; 3]], RotationTransform[
      -camRotAngleDeg * π / 180, camRotAxis], TranslationTransform[-camPos]}
   )];
camCapture[points3D_, cam_, f_] := Module[{pointIm, pointImPixel, pointImF}, (
    (* find 3D point in camera world *)
    pointIm = N[cam[points3D]];
```

```
    (* find pixel value *)
    pointImPixel = pointIm / pointIm[[All, 3]];
    (* Get same z as the image plane *)
    pointImF = pointImPixel * f;
    (* Find the point in world coordinates *)
    {pointImPixel[[All, 1 ;; 2]], InverseFunction[cam][pointImF]}
   )];
showCam[cam_, f_] :=
  Module[{im, camLoc, coinsW, coinsIm, csW, csIm, axeOptiqueW, axeOptiqueIm}, (
    im = InverseFunction[cam];
    camLoc = TransformationMatrix[im][[1 ;; 3, 4]];
    (* coins exprimés dans le monde de la caméra, en 3D *)
    coinsW = N[{{0, 0, 1}, {imX, 0, 1}, {imX, imY, 1}, {0, imY, 1}} * -f];
    coinsIm = N[{{0, 0, 1}, {imX, 0, 1}, {imX, imY, 1}, {0, imY, 1}} * f];
    csW = im /@ coinsW;
    csIm = im /@ coinsIm;
    axeOptiqueW =
     N[Normalize[Cross[csW[[2]] - csW[[1]], csW[[3]] - csW[[1]]]] * -f];
    axeOptiqueIm = N[Normalize[Cross[csIm[[2]] - csIm[[1]],
          csIm[[3]] - csIm[[1]]]] * f];
    Show[
     Graphics3D[{
       Black, Thick,
       Line[{camLoc, #}] & /@ csIm,
       Line[{camLoc, camLoc + axeOptiqueIm}],
       Opacity[0.25],
       EdgeForm[Thick],
       Polygon[csIm]
      }]
    ]
   )];
```

In[3760]:=
```
imX = 100;
imY = 100;
aovDeg1 = 90;
camPos1 = {0, 0, 0};
camRotAngleDeg1 = 0;
camRotAxis1 = {1, 0 , 0};
camFull1 = camera[imX, imY, aovDeg1, camPos1, camRotAngleDeg1, camRotAxis1];
cam1 = camFull1[[1]];
f1 = camFull1[[2]];
K1 = camFull1[[3]];
R1 = camFull1[[4]];
T1 = camFull1[[5]];
camCapturePoints1 = camCapture[points3D, cam1, f1];
camImgPoints1 = camCapturePoints1[[1]];
camPointsDeproj1 = camCapturePoints1[[2]];
Show[axes[],
  Graphics3D[{Red, PointSize[0.02], Point[#]}] & /@ points3D,
  Graphics3D[{Green, PointSize[0.02], Point[#]}] & /@ camPointsDeproj1,
 Graphics3D[{Blue, Line[{camPos1, #}] & /@ points3D}],
 showCam[cam1, f1]]
```

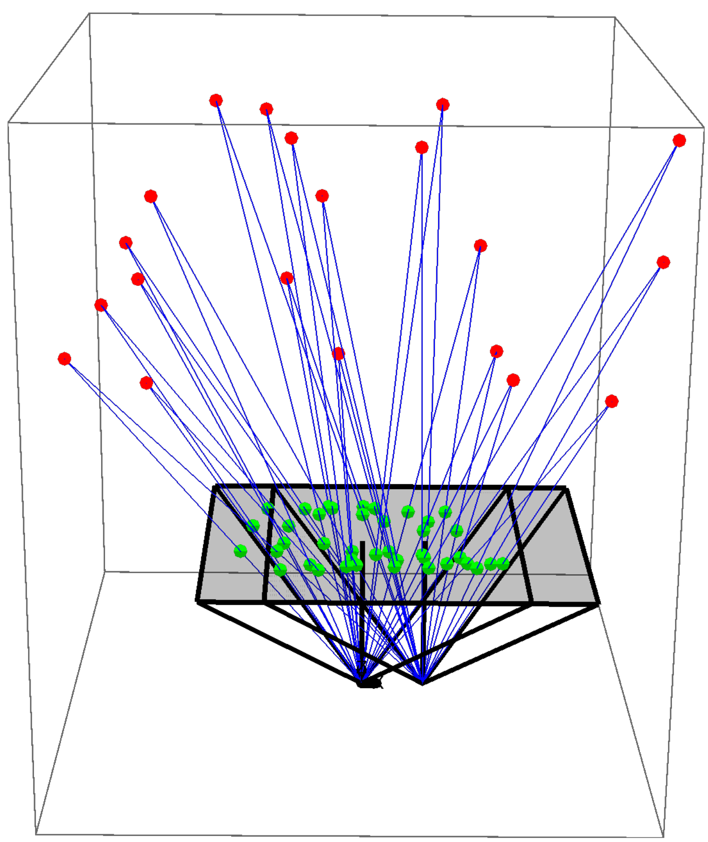
Out[3775]=



Example1: Camera1 + translation in X

In[3917]:=
```
aovDeg2 = aovDeg1;
camTransl2 = {20, 0, 0};
camPos2  = camPos1 + camTransl2;
camRotAngleDeg2  = 0;
camRotAxis2  = {1, 0 , 0};
camFull2 = camera[imX, imY, aovDeg2, camPos2, camRotAngleDeg2, camRotAxis2];
cam2 = camFull2[[1]];
f2 = camFull2[[2]];
K2 = camFull2[[3]];
R2 = camFull2[[4]];
T2 = camFull2[[5]];
camCapturePoints2 = camCapture[points3D, cam2, f2];
camImgPoints2 = camCapturePoints2[[1]];
camPointsDeproj2 = camCapturePoints2[[2]];
Show[axes[],
  Graphics3D[{Red, PointSize[0.02], Point[#]}] & /@ points3D,
  Graphics3D[{Green, PointSize[0.02], Point[#]}] & /@ camPointsDeproj2,
 Graphics3D[{Blue, Line[{camPos2, #}] & /@ points3D}],
 showCam[cam2, f2],
 Graphics3D[{Green, PointSize[0.02], Point[#]}] & /@ camPointsDeproj1,
 Graphics3D[{Blue, Line[{camPos1, #}] & /@ points3D}],
 showCam[cam1, f1]]
```

Out[3931]=



SLAM

- Find Essential Matrix

1. Get K^(-1).p for each point (in pixels) in image1 and K^(-1).q for each point in image2
2. Solve for Essential Matrix E using (KinvP)^T.E.(KinvP)=0

- Estimate pose of second camera from Essential Matrix using SVD: E = [t]_x.R

3. Find the SVD of E = U.W.V^T, but use E = U.diag(1,1,0).V^T
4. Then, pose of second camera is [U.W.V^T | u3], where W={{0,-1,0},{1,0,0},{0,0,1}} and u3 is last column of U

We get 4 options.

- Triangulation

5. Triangulate an image point using the 4 possible poses, and see for which pose it lies to the front of both cameras
6. Using this pose, triangulate all pixels points

- Find Essential Matrix

In[4447]:=
```
(* 1. K^(-1).p and K^(-1).q *)
KinvP1 =
  Transpose[Inverse[K1].Transpose[Append[#, 1.] & /@ camImgPoints1]][[All, 1 ;; 2]];
KinvP2 = Transpose[Inverse[K2].Transpose[Append[#, 1.] & /@ camImgPoints2]][[All,
    1 ;; 2]];
KinvP1[[1 ;; 3]] // MatrixForm
```

Out[4449]//MatrixForm=

$$\begin{pmatrix} -0.283401 & 0.469291 \\ 0.383689 & -0.275232 \\ -0.196645 & 0.583271 \end{pmatrix}$$

In[2104]:=
```
KinvPFunc[K_, p_] := Transpose[Inverse[K].Transpose[Append[#, 1.] & /@ p]];
```

In[4450]:=
```
(* 2. Solve for E using KinvP and KinvQ *)
(* Construct the line equation considering KinvP.E.KinvQ=0 *)
line[{x1_, y1_}, {x2_, y2_}] =
  Coefficient[{x1, y1, 1}.{{e1, e2, e3}, {e4, e5, e6}, {e7, e8, e9}}.{x2, y2, 1},
   {e1, e2, e3, e4, e5, e6, e7, e8, e9}];
(* Make line equations from the pairs of points *)
ma = Table[line[KinvP1[[i]], KinvP2[[i]]], {i, 1, Length[KinvP1]}];
(* Solve for the variables (Essential matrix)
 by taking the last eigenvector of this matrix *)
{u, w, v} = SingularValueDecomposition[ma];
Ess = Partition[v[[All, -1]], 3] // Chop;
Ess // MatrixForm
```

Out[4454]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.707107 \\ 0 & -0.707107 & 0 \end{pmatrix}$$

- Estimate pose of second camera

In[4465]:=
```
(* 3. Find the SVD of E *)
{U, W, V} = SingularValueDecomposition[Ess];
```

In[4508]:=
```
(* 4. Estimate pose of second camera *)
W = {{0, -1, 0}, {1, 0, 0}, {0, 0, 1}};
pose2a = MapThread[Append,
   {Transpose[U.W.Transpose[V]], Transpose[U.W.Transpose[V]].-camTransl2}];
pose2b = MapThread[Append, {Transpose[U.W.Transpose[V]],
    -U.W.Transpose[V].-camTransl2}];
pose2c = MapThread[Append, {Transpose[U.Transpose[W].Transpose[V]],
    Transpose[U.Transpose[W].Transpose[V]].-camTransl2}];
pose2d = MapThread[Append, {Transpose[U.Transpose[W].Transpose[V]],
    -Transpose[U.Transpose[W].Transpose[V]].-camTransl2}];
pose2a // MatrixForm
pose2b // MatrixForm
pose2c // MatrixForm
pose2d // MatrixForm
```

Out[4513]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & -20. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

Out[4514]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & 20. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

Out[4515]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & -20. \\ 0. & -1. & 0. & 0. \\ 0. & 0. & -1. & 0. \end{pmatrix}$$

Out[4516]//MatrixForm=

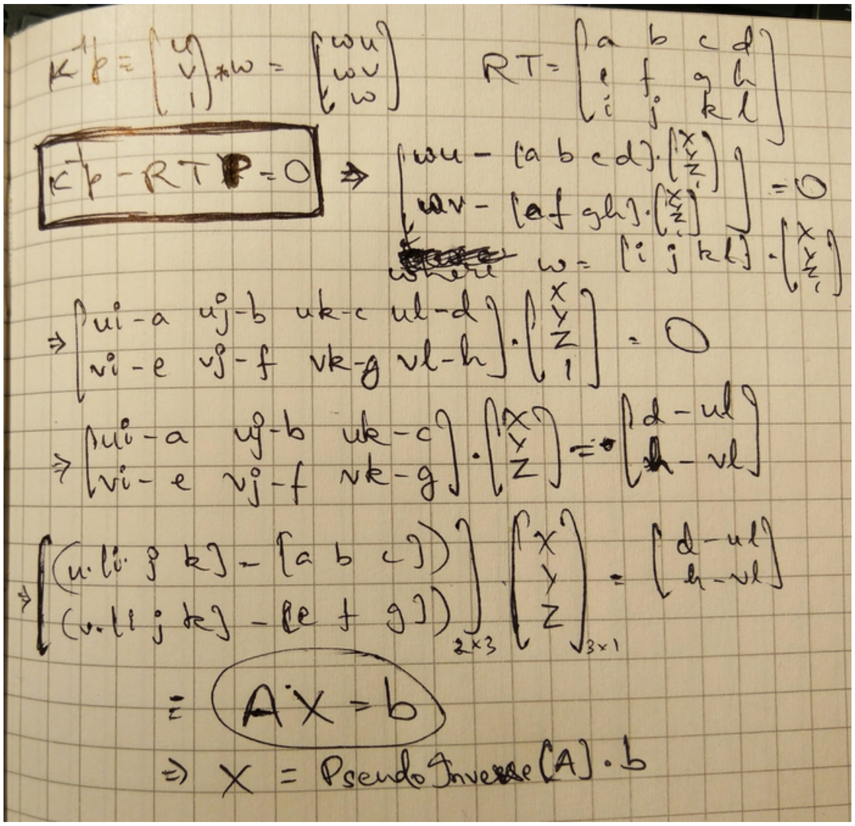
$$\begin{pmatrix} 1. & 0. & 0. & 20. \\ 0. & -1. & 0. & 0. \\ 0. & 0. & -1. & 0. \end{pmatrix}$$

Pose estimation is correct up to a scale factor for translation. I have inserted this scale.
Now let’s triangulate all points with the estimated pose.

- Triangulation

In[2188]:=
```
(* 5. Find the correct pose matrix out of the 4 options *)
(* by triangulating pixel points acc to the pose matrices *)
(* and checking if they are on the correct side of both cameras *)
pose1 = MapThread[Append,
   {IdentityMatrix[3, WorkingPrecision → MachinePrecision], {0., 0., 0.}}];
```

https://perception.inrialpes.fr/Publications/1997/HS97/HartleySturm-cviu97.pdf




```
In[2258]:= makeA[KinvPpoint_, RT_] := {KinvPpoint[[1]] * RT[[3, 1 ;; 3]] - RT[[1, 1 ;; 3]],
              KinvPpoint[[2]] * RT[[3, 1 ;; 3]] - RT[[2, 1 ;; 3]]};
            makeB[KinvPpoint_, RT_] := {RT[[1, 4]] - KinvPpoint[[1]] * RT[[3, 4]],
              RT[[2, 4]] - KinvPpoint[[2]] * RT[[3, 4]]};
```

```
In[4476]:= (* Triangulate the first point using pose2a *)
            pose2 = pose2a;
            A = Join[makeA[KinvP1[[1]], pose1], makeA[KinvP2[[1]], pose2]];
            A // MatrixForm
```

Out[4478]//MatrixForm=

$$\begin{pmatrix} -1. & 0. & -0.283401 \\ 0. & -1. & 0.469291 \\ -1. & 0. & -0.396589 \\ 0. & -1. & 0.469291 \end{pmatrix}$$

```
In[4479]:= b = Join[makeB[KinvP1[[1]], pose1], makeB[KinvP2[[1]], pose2]];
            b // MatrixForm
```

Out[4480]//MatrixForm=

$$\begin{pmatrix} 0. \\ 0. \\ -20. \\ 0. \end{pmatrix}$$

In[4481]:=
```
X = PseudoInverse[A].b;
X // MatrixForm
```

Out[4482]//MatrixForm=

$$\begin{pmatrix} -50.0761 \\ 82.9223 \\ 176.697 \end{pmatrix}$$

In[4483]:=
```
(* See if the 3D point lies to the front of both cameras *)
(* Consider the image plane of the camera. Eqn of a plane is n.(x-n)=0,
where n is the normal vector to the plane and x is a point on the plane. *)
(* For any other point,
n.(x-n) is +ve if point is in front of the plane, and -ve if behind *)
planeVal[p3D_, pose_] := Module[{n}, (
    n = Inverse[pose[[1 ;; 3, 1 ;; 3]]].{0, 0, 1};
    n.(p3D - pose[[1 ;; 3, -1]] - n)
   )];
planeVal[X, pose1]
planeVal[X, pose2]
```

Out[4484]= 175.697

Out[4485]= 175.697

In[4486]:=
```
pose2 = pose2b;
X = PseudoInverse[Join[makeA[KinvP1[[1]], pose1], makeA[KinvP2[[1]], pose2]]].
   Join[makeB[KinvP1[[1]], pose1], makeB[KinvP2[[1]], pose2]];
planeVal[X, pose1]
planeVal[X, pose2]
```

Out[4488]= -177.697

Out[4489]= -177.697

In[4490]:=
```
pose2 = pose2c;
X = PseudoInverse[Join[makeA[KinvP1[[1]], pose1], makeA[KinvP2[[1]], pose2]]].
   Join[makeB[KinvP1[[1]], pose1], makeB[KinvP2[[1]], pose2]];
planeVal[X, pose1]
planeVal[X, pose2]
```

Out[4492]= -30.4122

Out[4493]= 28.4122

In[4494]:=
```
pose2 = pose2d;
X = PseudoInverse[Join[makeA[KinvP1[[1]], pose1], makeA[KinvP2[[1]], pose2]]].
   Join[makeB[KinvP1[[1]], pose1], makeB[KinvP2[[1]], pose2]];
planeVal[X, pose1]
planeVal[X, pose2]
```

Out[4496]= 28.4122

Out[4497]= -30.4122

Thus, we can see that pose2a is the correct pose value for Camera 2.

In[4499]:=
```
pose2a // MatrixForm
R2.T2
```

Out[4499]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & -20. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

Out[4500]= TransformationFunction[$\left(\begin{array}{ccc|c} 1 & 0 & 0 & -20 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$]

We select pose2a by checking which pose has both points to the front of their cameras.

In[4501]:=
```
poses2 = {pose2a, pose2b, pose2c, pose2d};
X = Table[
   PseudoInverse[Join[makeA[KinvP1[[1]], pose1], makeA[KinvP2[[1]], poses2[[i]]]]].
    Join[makeB[KinvP1[[1]], pose1], makeB[KinvP2[[1]], poses2[[i]]]], {i,
    Length[poses2]}];
signs = Table[{planeVal[X[[i]], pose1], planeVal[X[[i]], poses2[[i]]]},
   {i, Length[poses2]}];
For[i = 1, i <= Length[signs], i++, If[signs[[i, 1]] > 0 && signs[[i, 2]] > 0, Break[]]];
i
poses2[[i]] // MatrixForm
R2.T2
```

Out[4505]= 1

Out[4506]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & -20. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

Out[4507]= TransformationFunction[$\left(\begin{array}{ccc|c} 1 & 0 & 0 & -20 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$]

In[4517]:=
```
pose2 = pose2a;
(* 6. Triangulate all points *)
Xs = Table[
   PseudoInverse[Join[makeA[KinvP1[[i]], pose1], makeA[KinvP2[[i]], pose2]]].Join[
     makeB[KinvP1[[i]], pose1], makeB[KinvP2[[i]], pose2]], {i, Length[KinvP1]}];
Xs[[1 ;; 4]] // MatrixForm
points3D[[1 ;; 4]] // MatrixForm
```

Out[4519]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \\ -33.575 & 99.5875 & 170.74 \\ 25.6058 & 61.6854 & 179.754 \end{pmatrix}$$

Out[4520]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \\ -33.575 & 99.5875 & 170.74 \\ 25.6058 & 61.6854 & 179.754 \end{pmatrix}$$

Thus, triangulation is also correct!

Putting in all into one function:

In[5600]:=
```
SLAM[K1_, camImgPoints1_, K2_, camImgPoints2_, camTransl_, realPose1_] := Module[
  {KinvP1, KinvP2, line, ma, u, v, w, Ess, U, W, V, Pa, Pb, Pc, Pd, poses2, makeA,
   makeB, planeVal, triangulate, findPose2Idx, idealPose1, pose2, Xs}, (
   (* 1. K^(-1).p and K^(-1).q *)
   KinvP1 = Transpose[
      Inverse[K1].Transpose[Append[#, 1.] & /@ camImgPoints1]][[All, 1 ;; 2]];
   KinvP2 = Transpose[Inverse[K2].Transpose[
        Append[#, 1.] & /@ camImgPoints2]][[All, 1 ;; 2]];
   (* 2. Solve for E using KinvP and KinvQ *)
   (* Construct the line equation considering KinvP.E.KinvQ=0 *)
   line[{x1_, y1_}, {x2_, y2_}] =
    Coefficient[{x1, y1, 1}.{{e1, e2, e3}, {e4, e5, e6}, {e7, e8, e9}}.{x2, y2, 1},
     {e1, e2, e3, e4, e5, e6, e7, e8, e9}];
   (* Make line equations from the pairs of points *)
   ma = Table[line[KinvP1[[i]], KinvP2[[i]]], {i, 1, Length[KinvP1]}];
   (* Solve for the variables (Essential matrix)
    by taking the last eigenvector of this matrix *)
   {u, w, v} = SingularValueDecomposition[ma];
   Ess = Partition[v[[All, -1]], 3] // Chop;
   (* 3. Find the SVD of E *)
   {U, W, V} = SingularValueDecomposition[Ess];
   (* 4. Estimate 4 possible poses of second camera,
   assuming pose of first camera is ideal *)
   W = {{0, -1, 0}, {1, 0, 0}, {0, 0, 1}};
   Pa = MapThread[Append,
```

```
  {Transpose[U.W.Transpose[V]], Transpose[U.W.Transpose[V]].-camTransl}];
Pb = MapThread[Append, {Transpose[U.W.Transpose[V]],
    -Transpose[U.W.Transpose[V]].-camTransl}];
Pc = MapThread[Append, {Transpose[U.Transpose[W].Transpose[V]],
    Transpose[U.Transpose[W].Transpose[V]].-camTransl}];
Pd = MapThread[Append, {Transpose[U.Transpose[W].Transpose[V]],
    -Transpose[U.Transpose[W].Transpose[V]].-camTransl}];
poses2 = {Pa, Pb, Pc, Pd};
(* Multiply the poses with realPose1 to get actual poses *)
(*poses2=(Append[realPose1,{0.,0.,0.,1}].Append[#,{0.,0.,0.,1.}])[[1;;3]]&/@
    poses2;*)
(* 5. Find the correct pose matrix out of the 4 options *)
(* Triangulation *)
makeA[KinvPpoint_, RT_] := {KinvPpoint[[1]] * RT[[3, 1 ;; 3]] - RT[[1, 1 ;; 3]],
  KinvPpoint[[2]] * RT[[3, 1 ;; 3]] - RT[[2, 1 ;; 3]]};
makeB[KinvPpoint_, RT_] := {RT[[1, 4]] - KinvPpoint[[1]] * RT[[3, 4]],
  RT[[2, 4]] - KinvPpoint[[2]] * RT[[3, 4]]};
planeVal[p3D_, K_, pose_] := Module[{n}, (
    n = Inverse[pose[[1 ;; 3, 1 ;; 3]]].{0, 0, K[[1, 1]]};
    n.(p3D + pose[[1 ;; 3, -1]] - n)
  )];
triangulate[KinvPpoint1_, RT1_, KinvPpoint2_, RT2_] := Module[{A, b}, (
    A = Join[makeA[KinvPpoint1, RT1], makeA[KinvPpoint2, RT2]];
    b = Join[makeB[KinvPpoint1, RT1], makeB[KinvPpoint2, RT2]];
    PseudoInverse[A].b
  )];
findPose2Idx[KinvPpoint1_, RT1_, KinvPpoint2_, poses2_] :=
 Module[{X, signs, p, poseP}, (
    X = Table[triangulate[KinvPpoint1, RT1, KinvPpoint2, poses2[[i]]],
      {i, Length[poses2]}];
    signs = Table[{planeVal[X[[i]], K1, RT1], planeVal[X[[i]], K2, poses2[[i]]]},
      {i, Length[poses2]}];
    (*Print[signs];*)
    poseP = 1;
    For[p = 1, p <= Length[signs],
     p++, If[signs[[p, 1]] > 0 && signs[[p, 2]] > 0, (poseP = p;
       Break[])]];
    (*Print[poseP];*)
    poseP
  )];
idealPose1 = MapThread[Append,
  {IdentityMatrix[3, WorkingPrecision → MachinePrecision], {0., 0., 0.}}];
pose2 = poses2[[Commonest[Table[findPose2Idx[KinvP1[[i]], idealPose1,
       KinvP2[[i]], poses2], {i, Length[KinvP1]}], 1][[1]]]];
(* Correct pose2 with realPose1 *)
```

```
    pose2 =
     (Append[pose2, {0., 0., 0., 1.}].Append[realPose1, {0., 0., 0., 1}])[[1 ;; 3]];
    (* 6. Triangulate all pixel points *)
    Xs = Table[triangulate[KinvP1[[i]],
       realPose1, KinvP2[[i]], pose2], {i, Length[KinvP1]}];
    {pose2, Xs}
   )];
```

In[5601]:=
```
pose1 = MapThread[Append,
   {IdentityMatrix[3, WorkingPrecision → MachinePrecision], {0., 0., 0.}}];
{pose2, X3D} = SLAM[K1, camImgPoints1, K2, camImgPoints2, camTransl2, pose1];
pose2 // MatrixForm
R2.T2
X3D[[1 ;; 4]] // MatrixForm
points3D[[1 ;; 4]] // MatrixForm
```

Out[5603]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & -20. \\ 0. & 1. & 0. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

Out[5604]= TransformationFunction$\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & -20 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$

Out[5605]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \\ -33.575 & 99.5875 & 170.74 \\ 25.6058 & 61.6854 & 179.754 \end{pmatrix}$$

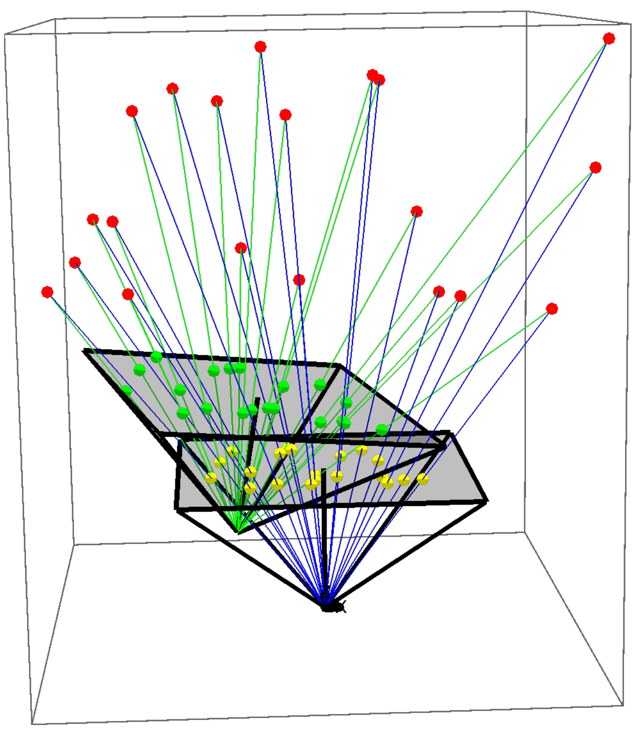
Out[5606]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \\ -33.575 & 99.5875 & 170.74 \\ 25.6058 & 61.6854 & 179.754 \end{pmatrix}$$

Example 2 - Translation in X,Y,Z + Rotation of camera

In[5404]:=
```
aovDeg3 = aovDeg1;
camTransl3 = {-30, 10, 25};
camPos3 = camPos1 + camTransl3;
camRotAngleDeg3 = 20;
camRotAxis3 = {1, 2 , 4};
camFull3 = camera[imX, imY, aovDeg3, camPos3, camRotAngleDeg3, camRotAxis3];
cam3 = camFull3[[1]];
f3 = camFull3[[2]];
K3 = camFull3[[3]];
R3 = camFull3[[4]];
T3 = camFull3[[5]];
camCapturePoints3 = camCapture[points3D, cam3, f3];
camImgPoints3 = camCapturePoints3[[1]];
camPointsDeproj3 = camCapturePoints3[[2]];
Show[axes[],
 Graphics3D[{Red, PointSize[0.02], Point[#]}] & /@ points3D,
 Graphics3D[{Green, PointSize[0.02], Point[#]}] & /@ camPointsDeproj3,
 Graphics3D[{Green, Line[{camPos3, #}] & /@ points3D}],
 showCam[cam3, f3],
 Graphics3D[{Yellow, PointSize[0.02], Point[#]}] & /@ camPointsDeproj1,
 Graphics3D[{Blue, Line[{camPos1, #}] & /@ points3D}],
 showCam[cam1, f1]]
```

Out[5418]=



In[5607]:=
```
{pose3, X3D} = SLAM[K1, camImgPoints1, K3, camImgPoints3, camTransl3, pose1];
pose3 // MatrixForm
N[Composition[R3, T3]]
X3D[[1 ;; 2]] // MatrixForm
points3D[[1 ;; 2]] // MatrixForm
```

Out[5608]//MatrixForm=

$$\begin{pmatrix} 0.942564 & 0.304283 & -0.137783 & 28.6787 \\ -0.292796 & 0.95118 & 0.0976092 & -20.7359 \\ 0.160757 & -0.0516607 & 0.985641 & -19.3017 \end{pmatrix}$$

Out[5609]= TransformationFunction$\left[\left(\begin{array}{ccc|c} 0.942564 & 0.304283 & -0.137783 & 28.6787 \\ -0.292796 & 0.95118 & 0.0976092 & -20.7359 \\ 0.160757 & -0.0516607 & 0.985641 & -19.3017 \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$

Out[5610]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \end{pmatrix}$$

Out[5611]//MatrixForm=

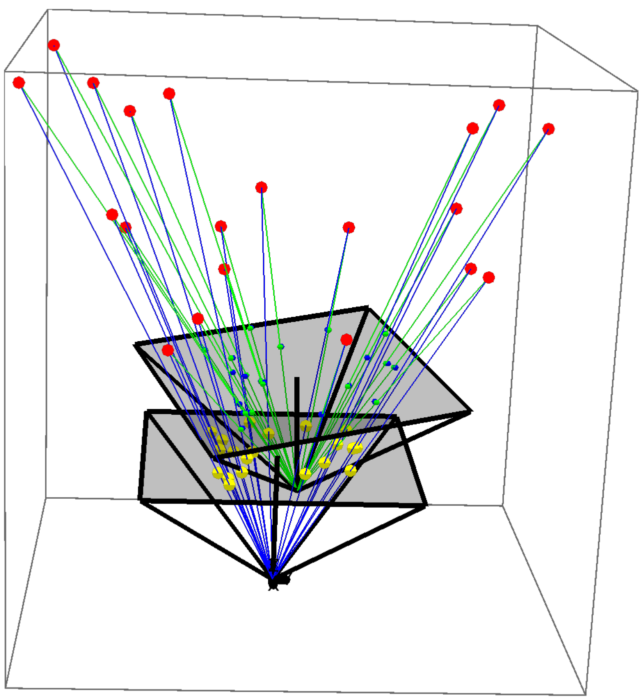
$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \end{pmatrix}$$

It works!

Example 3 - Noisy pixel values

In[5430]:=
```
noisyCamImgPoints3 = camImgPoints3 + RandomReal[{-1, 1}, {Length[camImgPoints3], 2}];
noisyCamPointsDeproj3 =
  InverseFunction[cam3][f3 * Append[#, 1.] & /@ noisyCamImgPoints3];
Show[axes[],
 Graphics3D[{Red, PointSize[0.02], Point[#]}] & /@ points3D,
 Graphics3D[{Green, PointSize[0.01], Point[#]}] & /@ camPointsDeproj3,
 Graphics3D[{Blue, PointSize[0.01], Point[#]}] & /@ noisyCamPointsDeproj3,
 Graphics3D[{Green, Line[{camPos3, #}] & /@ points3D}],
 showCam[cam3, f3],
 Graphics3D[{Yellow, PointSize[0.02], Point[#]}] & /@ camPointsDeproj1,
 Graphics3D[{Blue, Line[{camPos1, #}] & /@ points3D}],
 showCam[cam1, f1]]
```

Out[5432]=



In[5612]:=
```
{pose3, X3D} = SLAM[K1, camImgPoints1, K3, noisyCamImgPoints3, camTransl3, pose1];
pose3 // MatrixForm
N[R3.T3]
X3D[[1 ;; 2]] // MatrixForm
points3D[[1 ;; 2]] // MatrixForm
```

Out[5613]//MatrixForm=

$$\begin{pmatrix} 0.942473 & 0.309585 & -0.126104 & 28.3309 \\ -0.297394 & 0.948784 & 0.106609 & -21.0749 \\ 0.15265 & -0.0629735 & 0.986272 & -19.4476 \end{pmatrix}$$

Out[5614]= TransformationFunction[
$$\left(\begin{array}{ccc|c} 0.942564 & 0.304283 & -0.137783 & 28.6787 \\ -0.292796 & 0.95118 & 0.0976092 & -20.7359 \\ 0.160757 & -0.0516607 & 0.985641 & -19.3017 \\ \hline 0. & 0. & 0. & 1. \end{array}\right)$$
]

Out[5615]//MatrixForm=

$$\begin{pmatrix} -50.8335 & 83.0572 & 179.305 \\ 42.9775 & -32.4163 & 113.534 \end{pmatrix}$$

Out[5616]//MatrixForm=

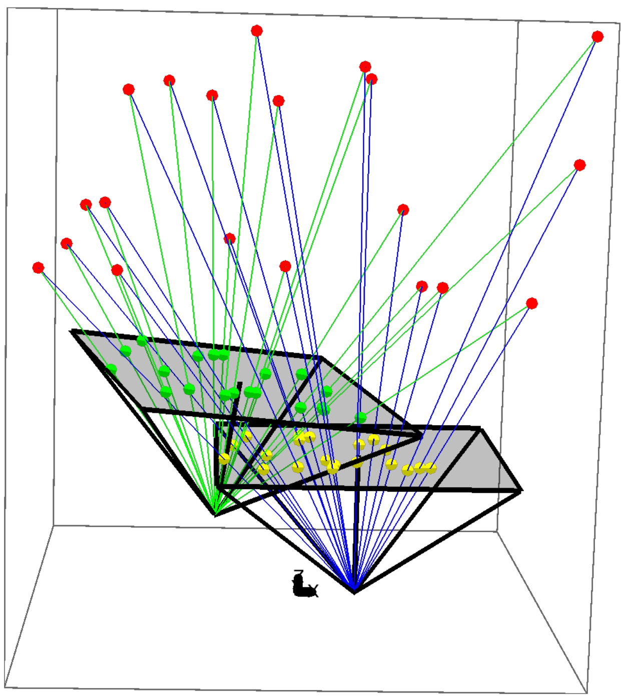
$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \end{pmatrix}$$

Good with noise too!

Example 4 - Non-origin camera positions

In[5528]:=
```
Show[axes[],
 Graphics3D[{Red, PointSize[0.02], Point[#]}] & /@ points3D,
 Graphics3D[{Green, PointSize[0.02], Point[#]}] & /@ camPointsDeproj3,
 Graphics3D[{Green, Line[{camPos3, #}] & /@ points3D}],
 showCam[cam3, f3],
 Graphics3D[{Yellow, PointSize[0.02], Point[#]}] & /@ camPointsDeproj2,
 Graphics3D[{Blue, Line[{camPos2, #}] & /@ points3D}],
 showCam[cam2, f2]]
```

Out[5528]=



In[5623]:=
```
{pose3From2, X3D} =
  SLAM[K2, camImgPoints2, K3, camImgPoints3, camPos3 - camPos2, pose2];
pose3From2 // MatrixForm
N[Composition[R3, T3]]
X3D[[1 ;; 2]] // MatrixForm
points3D[[1 ;; 2]] // MatrixForm
```

Out[5624]//MatrixForm=

$$\begin{pmatrix} -0.531233 & 0.599043 & 0.599116 & -36.9053 \\ 0.531757 & 0.786269 & -0.314667 & 15.9567 \\ 0.659565 & -0.151422 & 0.736237 & 2.89526 \end{pmatrix}$$

Out[5625]= TransformationFunction$\left[\left(\begin{array}{ccc|c} 0.942564 & 0.304283 & -0.137783 & 28.6787 \\ -0.292796 & 0.95118 & 0.0976092 & -20.7359 \\ 0.160757 & -0.0516607 & 0.985641 & -19.3017 \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$

Out[5626]//MatrixForm=

$$\begin{pmatrix} 4.16946 & 18.7326 & 39.9168 \\ -126.566 & 193.296 & -702.305 \end{pmatrix}$$

Out[5627]//MatrixForm=

$$\begin{pmatrix} -50.0761 & 82.9223 & 176.697 \\ 43.8513 & -31.4558 & 114.289 \end{pmatrix}$$