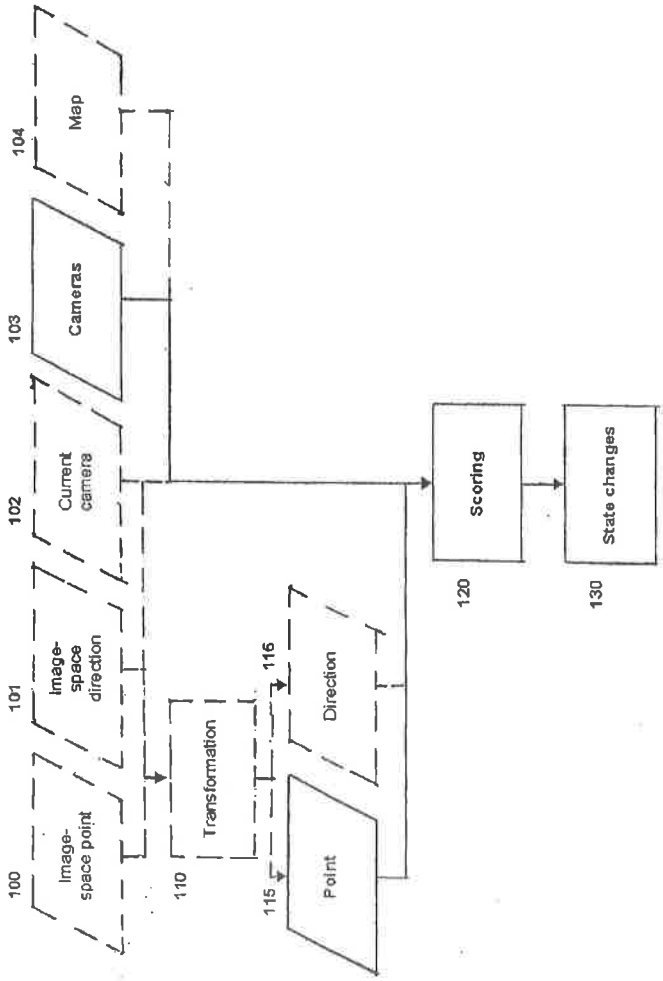


Figure 1 – High-level overview diagram of the navigation algorithm

FIG. 1

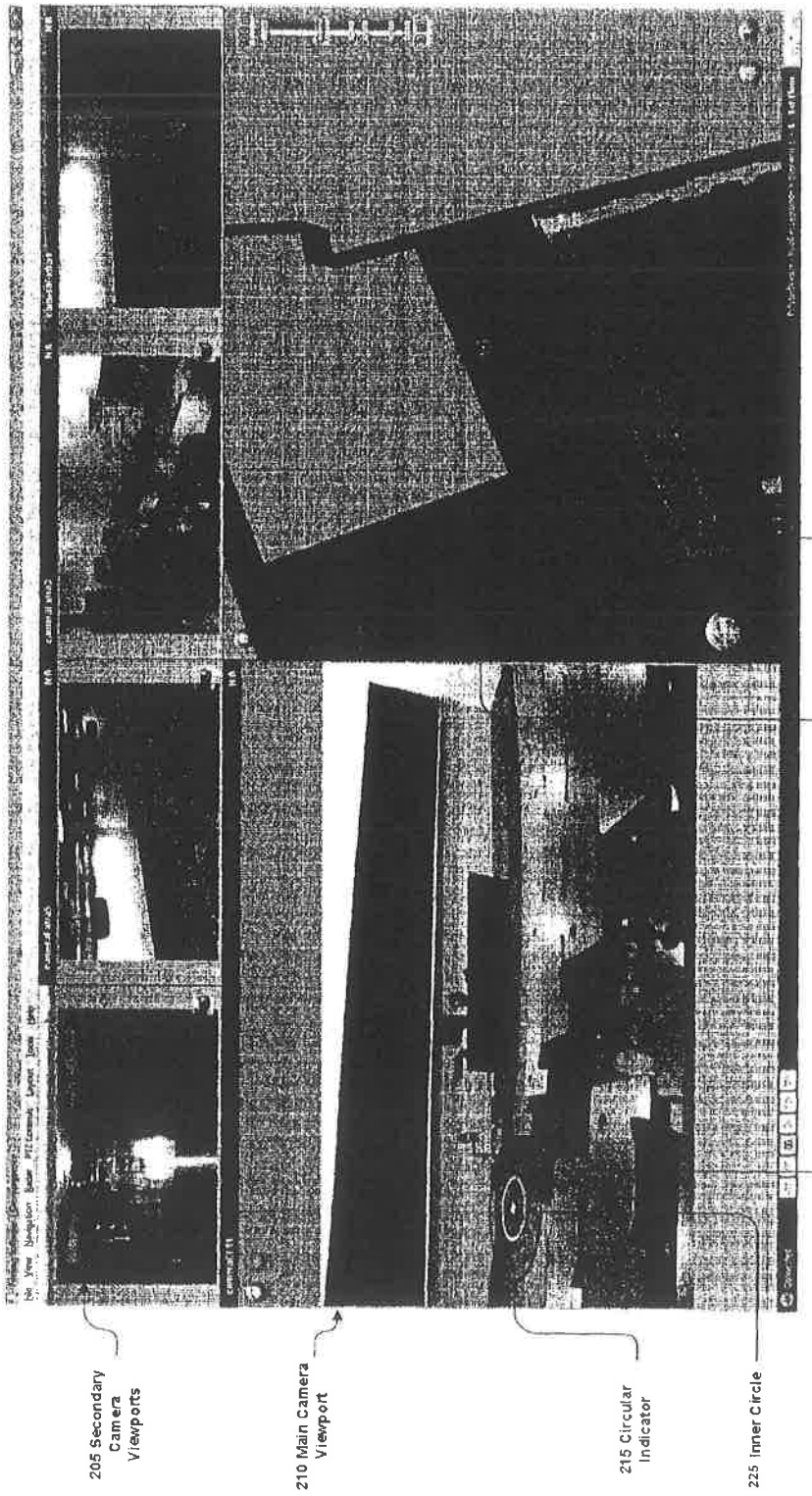


Figure 2 – Screenshot of Omnipresence 3D Software with navigation initiated
FIG. 2

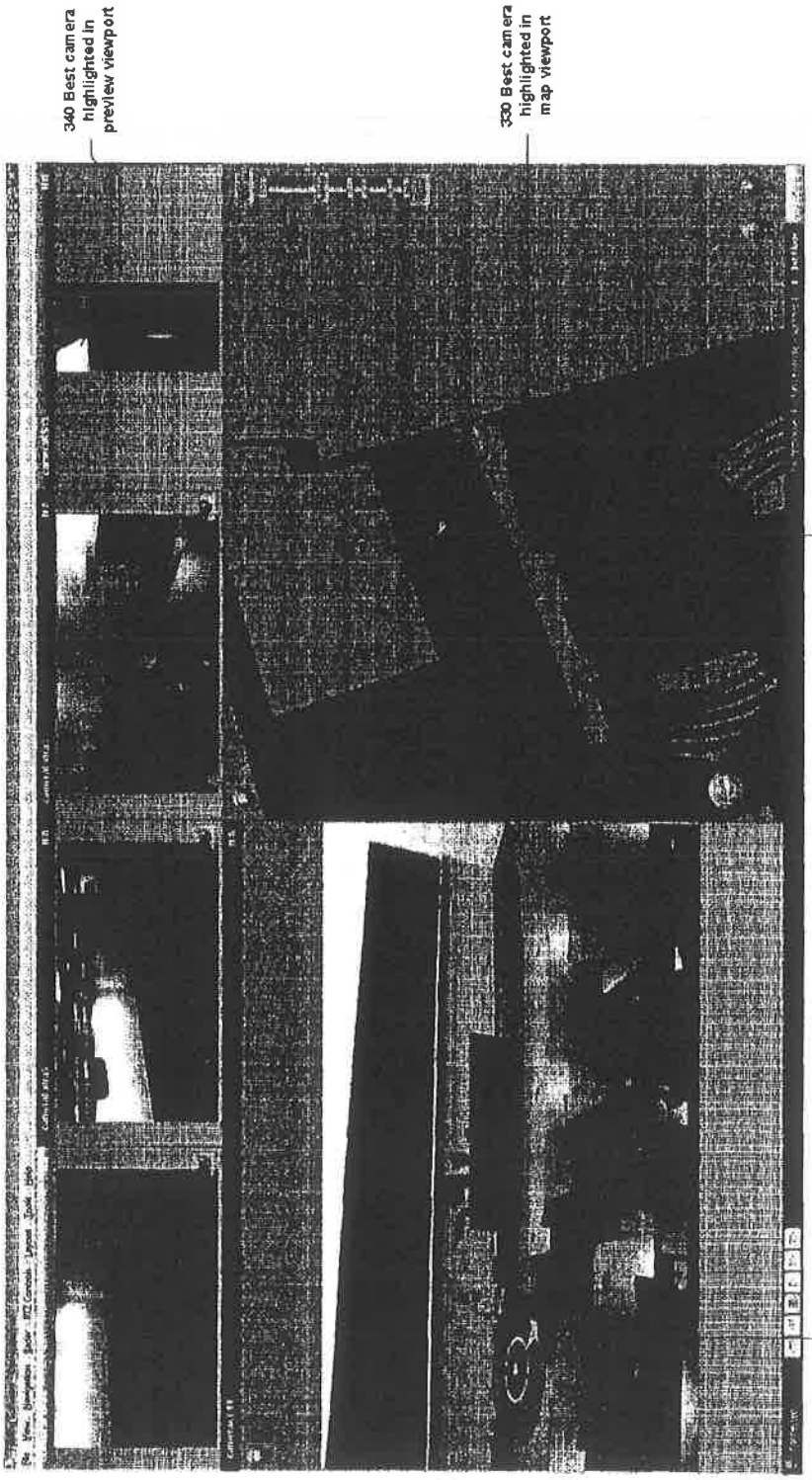


Figure 3 – Screenshot of Omnipresence 3D Software with navigation direction specified

FIG. 3

```
Float minSearchDistance <- 15.0 # Meters
Float maxSearchDistance <- 50.0 # Meters
Float optimalDistance <- 10.0 # Meters.
Float samplePointCount <- 300 # Number of samples taken on line segments.
Float[] directionToleranceAngles <-  [every 1.5 degree between -15.0 and 15.0 degrees]

Float UpdateCamerasScore(Point clickedPoint, Vector clickedDirection, Float
clickedDirectionMagnitude)
{
  # Reset the accumulated camera scores.
  foreach (Camera c in sceneSurveillanceCameras)
  {
    if (c.IsSelected)
      c.Score <- -100000;
    else
      c.score <- 0
  }

  # Vary the length of the search segments following the magnitude with which the
  # user has dragged in the clicked direction. The magnitude is in [0, 1] range.
  Float searchDistance <- minSearchDistance + clickedDirectionMagnitude *
    (maxSearchDistance - minSearchDistance)

  # Accumulate scores for a fan of line segments following the clicked direction.
  foreach (Angle angle in directionToleranceAngles)
  {
    # Prepare one line segment that will be scored.
    Vector searchDirection <- clickedDirection.RotatedBy(angle, Axis::Z)
    Point searchEndPoint <- clickedPoint + searchDirection * searchDistance
    LineSegment searchSegment(clickedPoint, searchEndPoint)

    foreach (Point samplePoint in searchSegment.GetPointsAlong(samplePointCount))
    {
      Float sampleDistance <- ||samplePoint - clickedPoint||

      # Calculate the base score for this sample point. The maximum score is at
      # "optimalDistance" along the search segment. It decreases quadratically from
      # this distance. Range is [0, 1].
      Float baseScore <- 1 - (sampleDistance - optimalDistance)^2 / searchDistance^2
```

Figure 4a. The Fast Track camera scoring algorithm.

```
        # Update each camera's score for this sample point.
        foreach (Camera camera in sceneSurveillanceCameras)
        {
          # The camera should score points if it can see this sample point. This
          # is done in Omnipresence by sampling a world position map texture generated
          # from the camera's point of view and comparing whether the sample point is
          # closer than the point on the map.

          if (camera.IsPointVisible(samplePoint))
          {
            # Calculate the score multiplier ratio based on the distance of the sample
            # point to the camera. The maximum ratio is at "optimalDistance" from the
            # camera. It decrease linearly from this distance. Range is [-infinity, 1].
            Float cameraDistanceToSamplePoint <- ||camera.Position -  samplePoint||
            Float cameraSampleProximityScore <- 1 - Abs(cameraDistanceToSamplePoint -
              optimalDistance) / searchDistance

            camera.Score <- camera.Score + baseScore * cameraSampleProximityScore;
          }
        }
      }
    }
  }
```

Figure 4b. The Fast Track camera scoring algorithm.