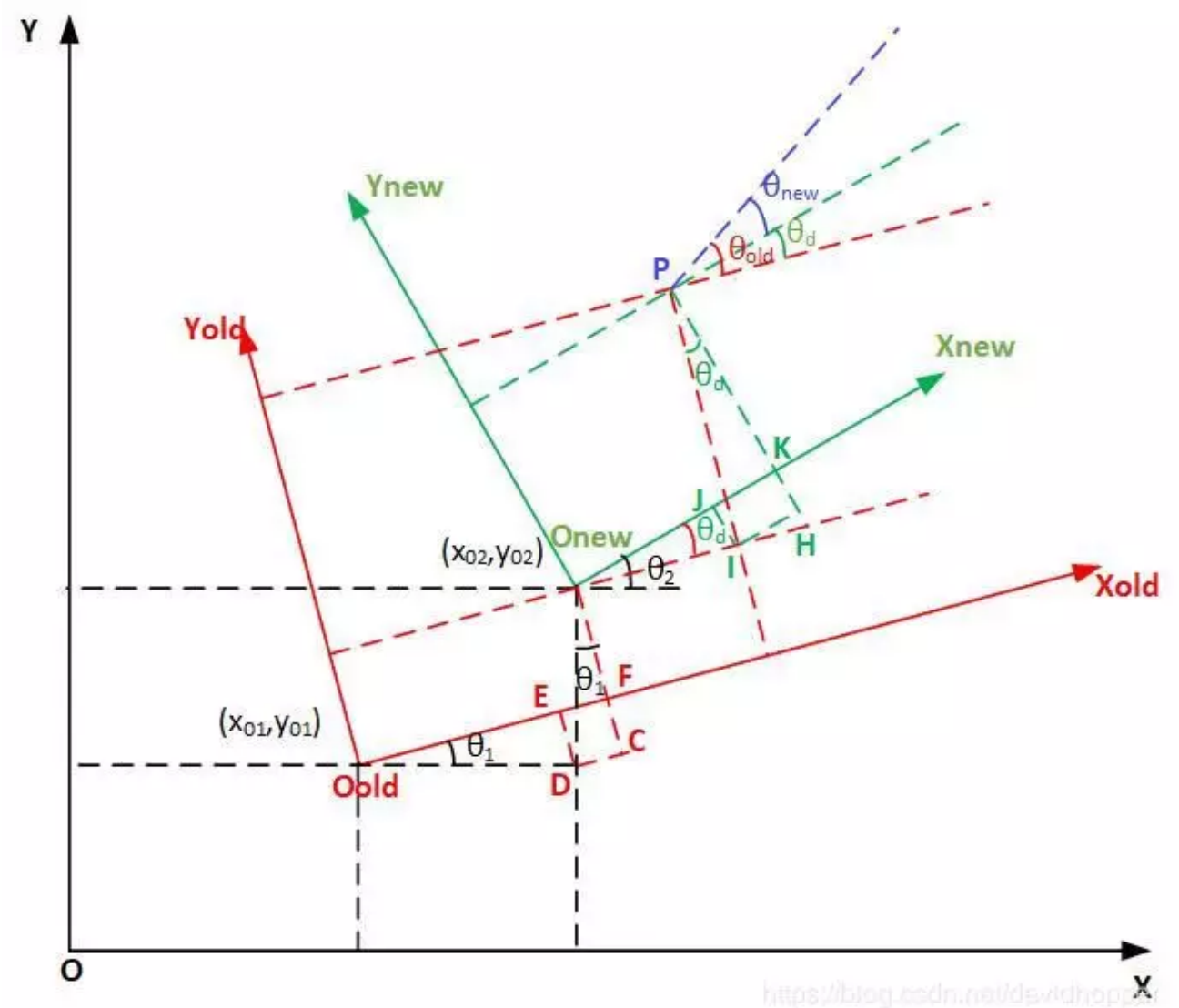
开发者说丨Apollo项目导航模式下的坐标转换研究

01

坐标变换公式

问题描述

如下图所示，XOY是**ENU全局坐标系**，$X_{old}O_{old}Y_{old}$与$X_{new}O_{new}Y_{new}$是**FLU车身坐标系**。已知坐标原点O_{old}在坐标系XOY中的坐标为$(x_{o1}, y_{o1}, \theta_1)$，$O_{new}$在坐标系XOY中的坐标为$(x_{o2}, y_{o2}, \theta_2)$。P点在前一帧车身坐标系$X_{old}O_{old}Y_{old}$中的坐标为$(x_{old}, y_{old}, \theta_{old})$，求解P点在当前帧车身坐标系$X_{new}O_{new}Y_{new}$中的坐标为$(x_{new}, y_{new}, \theta_{new})$。



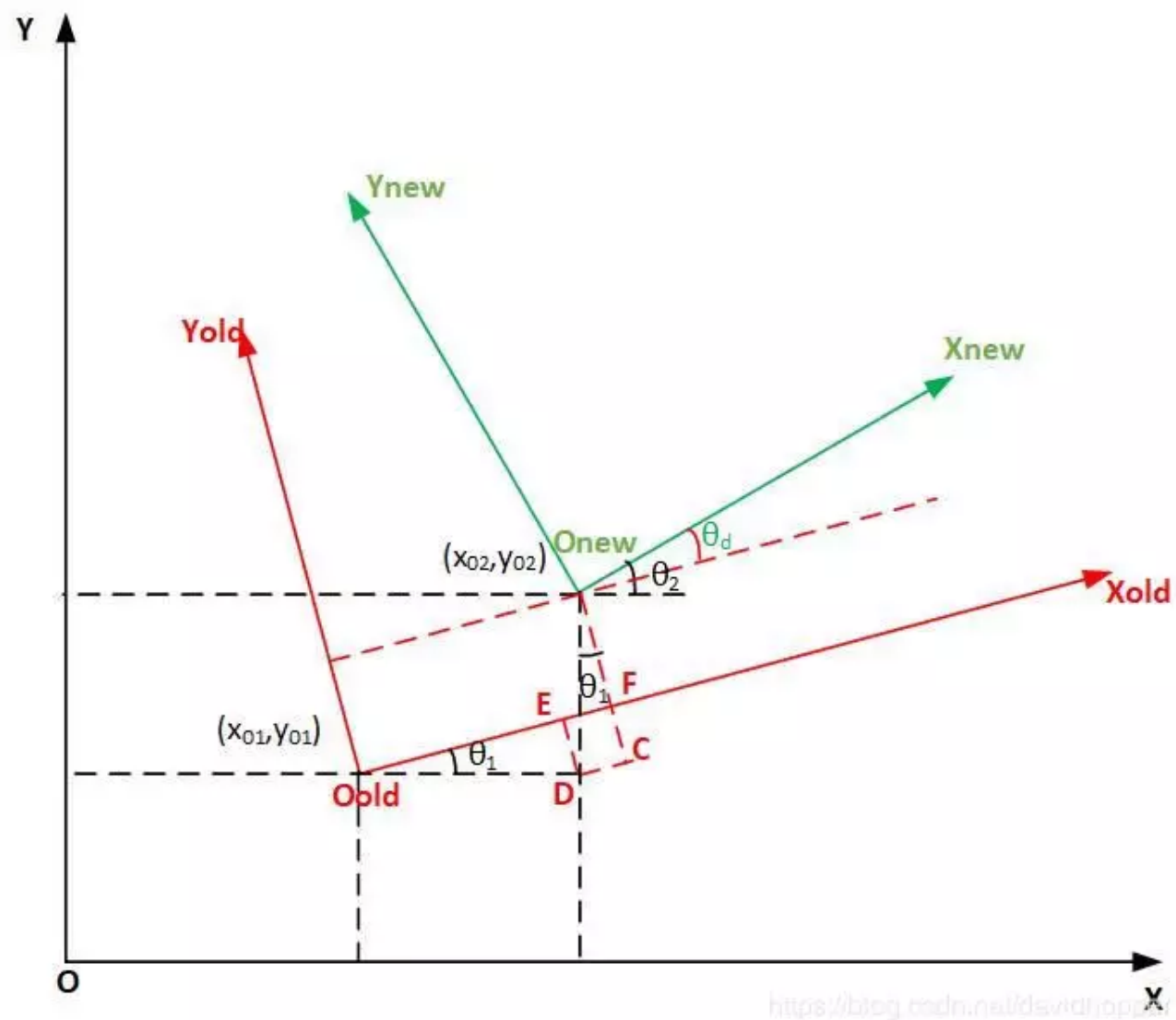
III 公式推导

如下图所示，当前帧坐标原点 O_{new} 在前一帧**车身坐标系** $X_{old}O_{old}Y_{old}$ 中的坐标 (x_d, y_d, θ_d) 可通过下述表达式计算：

$$x_d = O_{old}E + EF = O_{old}E + DC = (x_{o2} - x_{o1})\cos\theta_1 + (y_{o2} - y_{o1})\sin\theta_1 \quad (1)$$

$$y_d = O_{new}C - FC = O_{new}C - ED = (y_{o2} - y_{o1})\cos\theta_1 - (x_{o2} - x_{o1})\sin\theta_1 \quad (2)$$

$$\theta_d = \theta_2 - \theta_1 \quad (3)$$

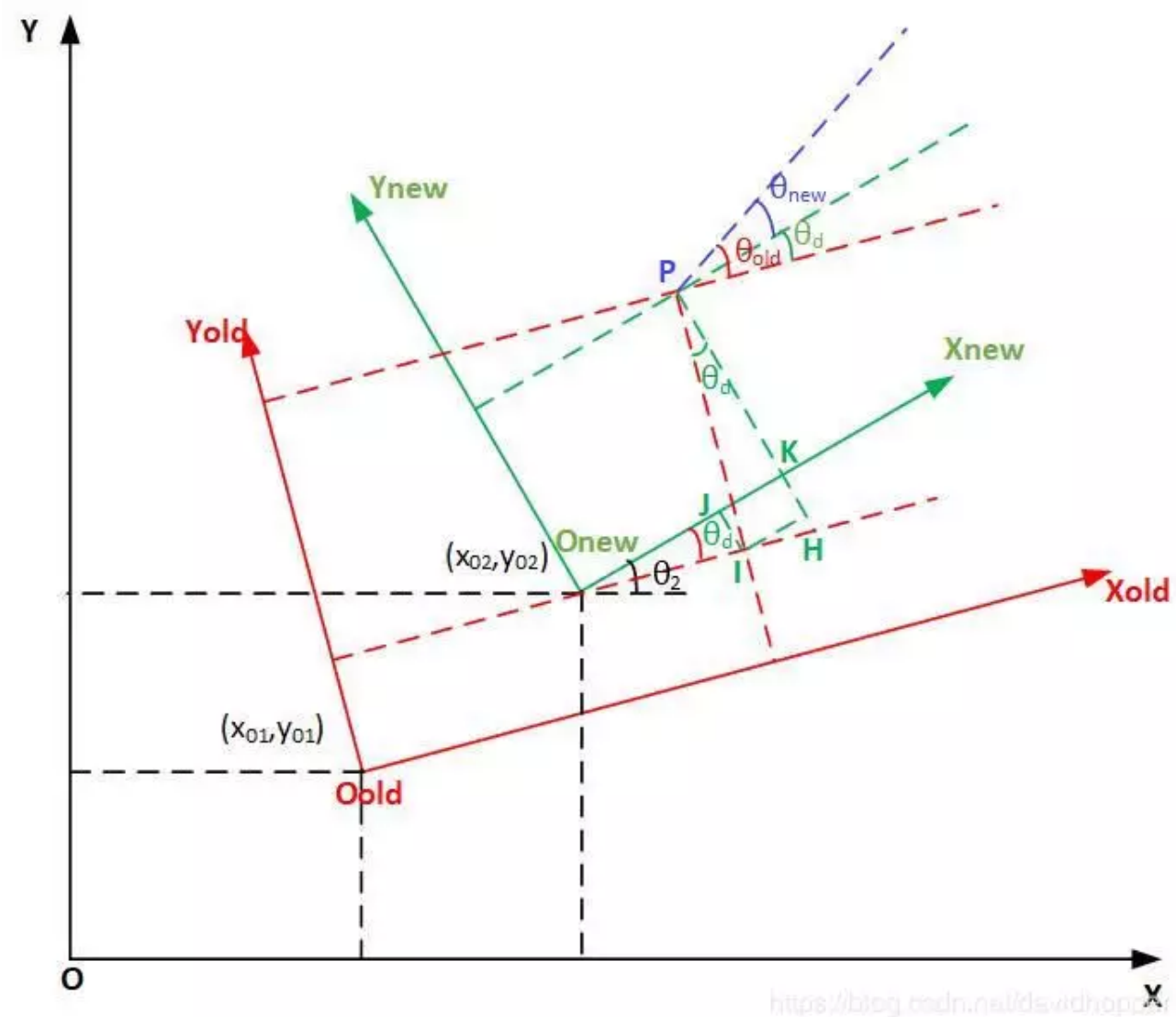


如下图所示，P点在当前帧车身坐标系$X_{new}O_{new}Y_{new}$中的坐标$(x_{new},y_{new},\theta_{new})$可通过下述表达式计算：

$$x_{new} = O_{new}J + JK = O_{new}J + IH = (x_{old} - x_d)cos\ \theta_d + (y_{old} - y_d)sin\ \theta_d \qquad (4)$$

$$y_{new} = PH - KH = PH - JI = (y_{old} - y_d)cos\ \theta_d - (x_{old} - x_d)sin\ \theta_d \qquad (5)$$

$$\theta_{new} = \theta_{old} - \theta_d \qquad (6)$$



02

坐标变换代码

坐标变换代码见modules/planning/navi_planning.cc中的NaviPlanning::RunOnce函数，具体代码如下：

```
void NaviPlanning::RunOnce(const LocalView& local_view,
                           ADCTrajectory* const trajectory_pb) {
  // ...
  auto vehicle_config =
      ComputeVehicleConfigFromLocalization(*local_view_.localization_estimate);
```

```
  if (last_vehicle_config_.is_valid_ && vehicle_config.is_valid_) {
    auto x_diff_map = vehicle_config.x_ - last_vehicle_config_.x_;
    auto y_diff_map = vehicle_config.y_ - last_vehicle_config_.y_;

    auto cos_map_veh = std::cos(last_vehicle_config_.theta_);
    auto sin_map_veh = std::sin(last_vehicle_config_.theta_);

    auto x_diff_veh = cos_map_veh * x_diff_map + sin_map_veh * y_diff_map;
    auto y_diff_veh = -sin_map_veh * x_diff_map + cos_map_veh * y_diff_map;

    auto theta_diff = vehicle_config.theta_ - last_vehicle_config_.theta_;

    TrajectoryStitcher::TransformLastPublishedTrajectory(
        x_diff_veh, y_diff_veh, theta_diff, last_publishable_trajectory_.get())
  }
  // ...
}
```

<左右滑动以查看完整代码>

其中的

NaviPlanning::ComputeVehicleConfigFromLocalization函数代码为：

```
NaviPlanning::VehicleConfig NaviPlanning::ComputeVehicleConfigFromLocalization(
    const localization::LocalizationEstimate& localization) const {
  NaviPlanning::VehicleConfig vehicle_config;

  if (!localization.pose().has_position()) {
    return vehicle_config;
  }

  vehicle_config.x_ = localization.pose().position().x();
  vehicle_config.y_ = localization.pose().position().y();

  const auto& orientation = localization.pose().orientation();

  if (localization.pose().has_heading()) {
    vehicle_config.theta_ = localization.pose().heading();
  } else {
    vehicle_config.theta_ = common::math::QuaternionToHeading(
        orientation.qw(), orientation.qx(), orientation.qy(), orientation.qz())
  }

  vehicle_config.is_valid_ = true;
  return vehicle_config;
}
```

TrajectoryStitcher::TransformLastPublishedTrajectory 函数位于文件modules/planning/common/trajectory_stitcher.cc中，代码如下：

```
void TrajectoryStitcher::TransformLastPublishedTrajectory(
    const double x_diff, const double y_diff, const double theta_diff,
    PublishableTrajectory* prev_trajectory) {
  if (!prev_trajectory) {
    return;
  }

  // R^-1
  double cos_theta = std::cos(theta_diff);
  double sin_theta = -std::sin(theta_diff);

  // -R^-1 * t
  auto tx = -(cos_theta * x_diff - sin_theta * y_diff);
  auto ty = -(sin_theta * x_diff + cos_theta * y_diff);

  std::for_each(prev_trajectory->begin(), prev_trajectory->end(),
                [&cos_theta, &sin_theta, &tx, &ty,
                 &theta_diff](common::TrajectoryPoint& p) {
                  auto x = p.path_point().x();
                  auto y = p.path_point().y();
                  auto theta = p.path_point().theta();

                  auto x_new = cos_theta * x - sin_theta * y + tx;
                  auto y_new = sin_theta * x + cos_theta * y + ty;
                  auto theta_new =
                      common::math::NormalizeAngle(theta - theta_diff);

                  p.mutable_path_point()->set_x(x_new);
                  p.mutable_path_point()->set_y(y_new);
                  p.mutable_path_point()->set_theta(theta_new);
                });
}
```