Presentation Script

Author: Kumar Anurag
kmranrg@unm.edu

Slide 01 - Title

Good Afternoon, everyone. I am Kumar Anurag from the University of New Mexico, and today I'm excited to present our work on resilient tracking in autonomous vehicles using a framework we call **DRISE** - **Distributionally Robust Input and State Estimation**.
This work is in collaboration with Kasra Azizi and my advisor Prof. Wenbin Wan.

Slide 02 - Introduction, Problem & Solution

Today, autonomous vehicles are becoming more common the road. To keep them safe, they need to know their own condition - like how fast they are going, what direction they're facing, or if they're sliding.
Some of this information is hard to measure directly. For example, the **sideslip angle** - which shows how much the car is sliding sideways, which is very important for safety, but it's hard to measure with normal sensors.
Also, low-cost sensors can give **noisy or wrong data**, especially in bad weather or tricky driving situations.
So, what's the solution?
We use **state estimation**. That means we combine sensor data with a math model of how the vehicle behaves. This helps us guess the hidden values that sensors can't measure directly.

Slide 02 - Introduction, Problem & Solution (contd...)

But this is not always easy. Sometimes the noise is strange, or the environment changes in unexpected ways. So, we need a smarter estimation method - one that works well even when things don't go as planned.

Slide 03 - Baseline: Input and State Estimation (ISE)

One popular method we often use is called **ISE** - Input and State Estimation. It doesn't just estimate the vehicle's internal state. It also tries to guess **unknown inputs** - like wind or tire slip, or uneven roads, these factors affect the vehicle in unexpected ways. So, ISE improves the accuracy by estimating both what we can't see and what we didn't plan for.

But there is a problem.

ISE works well only when the noise follows **perfect Gaussian distribution**. It can easily break down when we get **outliers** - like strange or faulty sensor readings.

That's we need something more **robust**.

Slide 04 - Baseline: Distributionally Robust Estimation (DRE)

The second method we looked at is **DRE** - Distributionally Robust Estimation. It's helpful when we don't know exactly what the sensor noise looks like. For example, if it's not a perfect Gaussian, DRE still works. How does it do that?
Let's look at this picture. The **black dot** in the center is the **nominal distribution** - our best guess. But the real distribution might be different - maybe like this **green point**.
So, DRE builds a **safe zone** around the nominal distribution, called an **ambiguity set** - the dotted circle.
It then prepares for the **worst-case** distribution inside that zone - the **red star**. That way, even if the real noise is strange, our estimates are still safe and reliable.

Slide 04 - Baseline: Distributionally Robust Estimation (DRE) (contd...)

However, DRE has some issues. It **doesn't estimate unknown inputs**, like sudden wind or surface friction.
And it's still **sensitive to outliers** - big errors in sensor readings.

Slide 05 - Problem Formulation

Now, let's talk about the actual problem we are solving.
We describe the vehicle using a **linear time-varying model** - which means the system changes over time, and we write it in terms of matrices like A_k, B_k, C_k.
The first equation models the **next state** of the vehicle, based on: - the **current state** x_k - the **known control input** u_k - like steering or acceleration - an **unknown input** d_k - like wind, road slope, or friction that we didn't model - and some **random noise** w_k
The second equation shows how we get our **measurements** y_k - but again, they can include **sensor noise** v_k, and even **outliers**.
So, we have three big challenges: - there's an unknown input that affects the system - the noise in both process and measurements is uncertain - and we may get bad measurements - like outliers
Our goal is to **estimate the state** x_k **accurately**, even when all these things happen.

Slide 06 - Unknown Input Estimation

The first challenge we tackle is the **unknown input** d_k.
These are forces or effects that we didn't include in our model - like road bumps, slippery surfaces, or even wind.
Most classical estimators (like Kalman Filter) ignore these. But in real-world driving, they can cause **big errors** if we don't handle them.
So, the first step in our **DRISE** algorithm is to **estimate this unknown input**.
We do that using the formula:

$$\hat{d_{k-1}} = M_k(y_k - C_k\hat{x}_k^-)$$

Let me explain it simply: - we compare the actual measurement y_k with what we expected $C_k\hat{x}_k^-$ - that difference is called the **innovation** - it tells us what's off - then we apply a gain matrix M_k to get our best guess of the unknown input
This allows us to handle **unmodeled disturbances**, and helps make our state estimation more accurate.

Slide 06 - Unknown Input Estimation (contd...)

Audience: How M_k is calculated? M_k is not a fixed constant, but is computed dynamically at each time step based on system matrix C_k, state covariance P_k, and measurement covariance R_k. In short, we don't choose manually, it's derived from our model and noise statistics during prediction step of the filter.

Slide 07 - Distributionally Robust Estimation

The second challenge we solve is about **noise uncertainty**.
In real life, the noise in our system and sensors may **not follow the exact Gaussian shape** we assume. It might change over time, or act strangely in some conditions.
If we use a normal Kalman Filter or even ISE, these assumptions can lead to **bad estimates**.
To fix this, we use the idea of **Distributionally Robust Estimation** or DRE.
Here's the key idea: instead of assuming one fixed noise distribution, we prepare for **a whole set of possible noise types**. We call this **ambiguity set**, written as $\mathcal{F}$.
Mathematically, it looks like this:

$$\min_{\text{estimator}} \max_{P \in \mathcal{F}} \mathbb{E}_P[\text{Error}]$$

It means: Choose the estimator that gives the best performance - even if the true noise is the worst one inside our safe zone.

Slide 07 - Distributionally Robust Estimation (contd...)

This makes the estimation more **reliable**, even when our noise model isn't perfect.

Slide 08 - Robust Update

The final challenge we deal with is **outliers** in the sensor data.
Sometimes a sensor might give a strange reading - maybe from a glitch, reflection, or communication delay.
These **bad values** can make the estimator jump or drift away from the true state.
To fix this, we use a concept from **robust statistics** called the **influence function**.
The goal is to **limit how much each measurement can affect the estimate** - especially when that measurement looks suspicious.
We use a special function called a **Huber-type clipping function**, written as $\psi(\mu)$. It behaves like this: - if the measurement difference μ is small, we use it normally. - but if μ is too big - meaning it might be an outlier - we cap it at a limit K
So large deviations don't completely break the estimate. This makes the update step much **more robust**, especially in tough real-world conditions.

Slide 08 - Robust Update (contd...)

Audience: How K is evaluated? The value of K is tuned empirically. Generally, we set value of K to 1.5 *to* 2 *times the standard deviation of the innovation (μ - measurement residual).*

Slide 09 - DRISE Framework Block Diagram

Now that we've seen the three core building blocks, let's look at how they all come together in the full **DRISE framework**.
On the left, we see **CARLA**, the simulator we use to model the autonomous vehicle. It gets: - control inputs u_k from the controller - affects from unknown inputs d_k - and disturbances from process noise w_k
The vehicle sends back sensor measurements y_k, which are also affected by measurement noise v_k.
On the right side is our **DRISE estimator** - it takes in the control input u_k and the measurements y_k, and processes them in 4 steps: 1. Prediction: Predict where the vehicle should be, based on the last state and control input 2. Unknown Input Estimation: Use sensor feedback to estimate unmodeled forces d_k 3. Time Update: Refine the state using the predicted unknown input 4. Robust Measurement Update: Using a clipped influence function to finalize the estimate

Slide 10 - The DRISE Algorithm Cycle

1. Prediction: First, we predict the next state based on the last estimated state x_{k-1}, and the known control input u_{k-1}. This gives us a rough idea of where the vehicle should be.
2. Unknown Input Estimation: Next, we estimate the unknown input d_{k-1} - for example, disturbances like wind or road slope. We do this by comparing what the sensors saw with what we expected, and using the gain matrix M_k.
3. Time Update: We refine the predicted state using the estimated unknown input. This helps us correct the state before even using the new measurement.
4. Robust Measurement Update: Finally, we apply a robust correction using the actual measurement. If the measurement seems reliable, we update normally. But if it looks like an outlier, we limit its effect using a clipped influence function $\psi(\cdot)$.

Slide 11 - Simulation Setup

Now, let's talk about the simulation setup we used to test the DRISE framework.
We use a standard **kinematic bicycle model** - it's a simplified way to simulate how a car moves.
This includes states like **position, velocity, and yaw angle**.
The **inputs** to the system are: - Steering angle - Acceleration
We also introduce an **unknown input** - a time-varying signal, which mimics things like road slope or friction changes that we don't model directly.
For realism, we include: - Process noise covariance Q_k, which adds uncertainty in how the system evolves. - Measurement noise covariance R_k, which affects the sensor readings.
And we make the test more challenging by injecting **outliers** and **noise deviations**.

Slide 11 - Simulation Setup (contd...)

Finally, we compare our method - **DRISE** - with three baselines: - Kalman Filter - Input and State Estimation - Distributionally Robust Estimation This setup helps us understand how well DRISE performs under tough and realistic conditions.

Slide 12 - CARLA Simulation Environment

To test how well DRISE works in real-world-like conditions, we used CARLA simulator.
CARLA is a powerful **open source platform** made for autonomous vehicle research.
It lets us simulate driving in **realistic urban environments**, with buildings, traffic, trees and even road textures.
It also provides detailed models for **sensors and vehicle dynamics**, so we can see how estimation performs under realistic conditions.
Most importantly, CARLA allows us to introduce **uncertainties** like unexpected forces.
So, this setup gives us a **challenging and believable** environment for evaluating our framework.

Slide 13, 14, and 15 - Results

First, this plot shows the **state estimation error** - how far off the estimators are from the true vehicle state. Here, we can see that the Kalman Filter performs the worst. It diverges when the noise or inputs don't match the assumptions. But DRISE clearly outperforms all, with the lowest overall error.
In the second plot, we show how well each method estimates **unknown inputs** - like wind, slope, or forces. **DRISE again shows much better accuracy**.
In the third plot, we show how well the **vehicle follows a planned 2D path**. **DRISE produces the closest math to the reference trajectory**.
Conclusion: This shows combining robustness and unknown input-state estimation really improves accuracy.

Slide 16 - Future Work

In future work, we also plan to explore different ways of defining ambiguity set - specifically using Wasserstein and KL divergence-based methods.

Slide 17 - Thank You

This brings me to the end of my presentation. I'd be happy to take questions now.