خودروی خودران هوشمند (لیگ فیزیکی پرو - رده سنی باز)

مسابقات فیراکاپ آزاد ایران ۲۰۲۶

مستندات معرفی تیم داتام داینامیکز - Datam dynamics

سیدحسین حسینی
مسئول نرم‌افزار
دانشگاه تهران مرکز – کارشناسی علوم کامپیوتر
ایمیل: seyedhossein.hosseini@iau.ir

میثم دهقان
مهندس ارشد مکانیک
دانشگاه شریف – ارشد مکانیک
ایمیل: dehghan.meisam@mech.sharif.edu

کیارش نیرالدین‌زاده
مهندس الکترونیک
دانشگاه تهران – ارشد مخابرات میدان
ایمیل: Kiarashny123@gmail.com

ارین امامی
کارشناس مکانیک و راننده حرفه‌ای
علم و فرهنگ – کارشناسی مکانیک
ایمیل: arian.emami2@gmail.com

چکیده

در این مستندات، طراحی و پیاده‌سازی خودروی خودران هوشمند برای مسابقات FIRA RoboWorld Cup ارائه گردید. تیم در مرحله انتقال از نمونه اولیه کلاسیک به نسل جدید یادگیری عمیق End-to-End قرار دارد. نمونه اولیه بر اساس الگوریتم‌های بینایی ماشین OpenCV و کنترل PID پیاده‌سازی شد و در شبیه‌ساز Avis Engine با دقت بیش از ۸۵ درصد در دنبال‌کنندگی خط آزمایش گردید. معماری پیشنهادی برای نسخه پیشرفته از شبکه عصبی چندوظیفه‌ای استفاده می‌کند که قادر به یادگیری مستقیم نگاشت تصویر به دستورات کنترلی، تشخیص و اطاعت از تابلوهای راهنمایی و استدلال زمانی با استفاده از LSTM است. سیستم از معماری توزیع‌شده با پردازش روی لپ‌تاپ و کنترل بلادرنگ روی ESP32 بهره می‌برد. شاسی RC

مقیاس ۱:۱۰ با موتور براش‌لس و سروو دیجیتال به کار گرفته شد. نتایج اولیه نشان‌دهنده آمادگی تیم برای مسابقات است.

کلمات کلیدی: خودروی خودران، یادگیری عمیق End-to-End، تشخیص تابلوی راهنمایی، بینایی ماشین، کنترل PID، شبیه‌ساز FIRA ،Avis Engine

---

1. مقدمه

تیم ما متشکل از چهار عضو با تخصص‌های مکمل است. سیدحسین حسینی به عنوان مهندس و مسئول طراحی معماری نرم‌افزاری، الگوریتم‌های بینایی ماشین و شبکه‌های عصبی عمیق می‌باشد و سابقه کار در پروژه‌های نرم افزاری اینترپرایز را دارد. کیارش نیرالدین‌زاده به عنوان مهندس الکترونیک، مسئول طراحی سخت‌افزار، برنامه‌نویسی میکروکنترلرها و مدیریت برق سیستم است. میثم دهقان به عنوان مهندس ارشد مکانیک، مسئول طراحی و بهینه‌سازی شاسی و تحلیل دینامیک خودرو می‌باشد. آرین امامی به عنوان کارشناس مکانیک و راننده حرفه‌ای، مسئول تست‌های عملکردی و جمع‌آوری داده‌های آموزشی است. این اولین حضور تیم در مسابقات FIRA است.

خودروی ما یک سیستم خودران مقیاس ۱:۱۰ بر پایه شاسی RC است که از دو واحد پردازشی استفاده می‌کند: لپ‌تاپ برای پردازش‌های پیچیده و ESP32 برای کنترل بلادرنگ. دوربین ESP32-CAM تصاویر را با ۲۰-۳۰ فریم بر ثانیه ضبط و از طریق WiFi ارسال می‌کند. سیستم بینایی ماشین خطوط را تشخیص داده و دستورات فرمان و سرعت را تولید می‌کند که به میکروکنترلر ارسال می‌شود. نمونه اولیه کلاسیک آماده و تست شده است و تیم در حال توسعه نسخه پیشرفته با یادگیری عمیق End-to-End می‌باشد.

2. لزوم پیاده‌سازی پروژه، طرح یا ایده

خودروهای خودران یکی از چالش‌برانگیزترین حوزه‌های رباتیک و هوش مصنوعی است. سیستم‌های سنتی بر پایه الگوریتم‌های کلاسیک استوار هستند که نیاز به پیاده‌سازی دستی قوانین برای هر سناریو دارند و انعطاف‌پذیری پایین، هزینه نگهداری بالا و مقیاس‌پذیری محدود دارند. پروژه ما با هدف غلبه بر این محدودیت‌ها از رویکرد یادگیری عمیق End-to-End بهره می‌برد که در آن یک شبکه عصبی به‌طور مستقیم از تصاویر، دستورات کنترلی را یاد می‌گیرد.
این پروژه به نیازهای تحقیق و توسعه در حوزه یادگیری عمیق، آموزش دانشجویان، شرکت در مسابقات بین‌المللی و توسعه دانش فنی قابل انتقال به صنعت پاسخ می‌دهد. قابلیت تشخیص و اطاعت از تابلوهای راهنمایی یکی از چالش‌های اصلی سیستم‌های خودران است که در پروژه ما به‌صورت یکپارچه در معماری شبکه عصبی پیاده‌سازی می‌شود.

3. نمونه‌های مشابه

پروژه‌های مشابهی در سطح بین‌المللی پیاده‌سازی شده‌اند. [1] NVIDIA DAVE-2 اولین سیستم End-to-End موفق برای رانندگی خودکار بود که از CNN ساده برای نگاشت تصویر به فرمان استفاده کرد اما فقط کنترل فرمان را انجام می‌داد و قابلیت تشخیص تابلو نداشت. [2] Waymo سیستم حرفه‌ای تجاری با سنسورهای متعدد است اما هزینه بالا و پیچیدگی زیاد دارد. [3] DonkeyCar پلتفرم متن‌باز برای خودروهای کوچک است که از Behavioral Cloning ساده استفاده می‌کند اما فاقد تشخیص تابلو و استدلال معنایی است. پروژه‌های [4] Udacity از رویکرد کلاسیک با OpenCV و PID استفاده می‌کنند که نیاز به تنظیم دستی دارند.

وجه تمایز پروژه ما استفاده از معماری هیبریدی است که نمونه اولیه کلاسیک را به عنوان baseline ایمن حفظ کرده و به تدریج به سیستم End-to-End ارتقا می‌یابد. معماری Multi-Task Learning ما یک شبکه واحد برای کنترل، تشخیص تابلو و استدلال زمانی دارد. سیستم ما قادر به سوئیچ هوشمند بین کنترل کلاسیک و شبکه عصبی بر اساس سطح اعتماد است که ایمنی را تضمین می‌کند. استفاده از شبیه‌ساز Avis Engine امکان Sim-to-Real transfer را فراهم می‌آورد.

4. نوآوری

نوآوری‌های اصلی پروژه ما عبارتند از:

۱. معماری Multi-Task End-to-End Learning: شبکه عصبی واحد برای سه وظیفه: (الف) کنترل مستقیم فرمان و سرعت از تصویر، (ب) تشخیص و طبقه‌بندی تابلوهای راهنمایی، (ج) استدلال زمانی با LSTM و Attention برای استفاده از تاریخچه مشاهدات و اطاعت از قوانین ترافیکی.

۲. انتقال تدریجی از کلاسیک به End-to-End: شروع با نمونه اولیه کلاسیک (OpenCV + PID) به عنوان baseline عملیاتی و مطمئن، سپس توسعه سیستم End-to-End به‌صورت موازی، و در نهایت یکپارچه‌سازی با مکانیزم سوئیچ هوشمند. این رویکرد ریسک را کاهش داده و امکان مقایسه عملکرد را فراهم می‌آورد.

۳. ایمنی چندلایه: سیستم ایمنی هیبریدی شامل (الف) Confidence-based switching: انتخاب خودکار بین شبکه عصبی و کنترل کلاسیک بر اساس اعتماد، (ب) Safety monitor: نظارت مداوم بر پارامترهای حیاتی مانند سرعت، فاصله از مانع و وضعیت ارتباط، (ج) Emergency fallback: بازگشت فوری به کنترل کلاسیک در شرایط بحرانی.

۴. تشخیص و اطاعت یکپارچه از تابلوها: برخلاف سیستم‌هایی که تشخیص تابلو را به‌صورت جداگانه انجام می‌دهند، معماری ما head اختصاصی برای تشخیص تابلو در شبکه عصبی اصلی دارد و Reward function در مرحله آموزش Reinforcement Learning شامل جریمه سنگین برای عدم اطاعت از تابلوهاست که باعث می‌شود agent یاد بگیرد به‌صورت طبیعی از قوانین پیروی کند.

۵. Sim-to-Real Transfer با Domain Adaptation: آموزش کامل در محیط شبیه‌ساز Avis Engine و سپس Fine-tuning با تکنیک‌های Domain Adaptation - برای کاهش شکاف واقعیت روی ربات واقعی.

ساختار و طراحی مکانیکی ربات

مشخصات شاسی

خودروی ما بر پایه شاسی RC مقیاس ۱:۱۰ طراحی شده است. این شاسی به دلیل استانداردسازی، در دسترس بودن قطعات یدکی، دینامیک مناسب و وزن پایین انتخاب گردید.

مشخصات کلی شاسی:

- مقیاس: ۱:۱۰
- فاصله محوری (Wheelbase): 258 mm
- عرض مسیر چرخ (Track Width): 160 mm
- فاصله از زمین (Ground Clearance): 10 mm
- جنس شاسی: پلاستیک

سیستم فرمان

سروو موتور:

- مدل: TSU-03
- محدوده حرکت: ±45 درجه
- گشتاور: **3.5 kg·cm** (در ولتاژ 6.0 V)
- سرعت: **0.17 s / 60°** (در ولتاژ 6.0 V)
- ولتاژ کاری: ۴.۸-۶.۰ ولت
- نوع:**آنالوگ (Pulse-width control / analog servo)**

مکانیزم فرمان:

- نوع هندسه: Ackermann
- حداکثر زاویه چرخش چرخ‌ها: 45 درجه
- جنس بازوهای فرمان: پلاستیک

سیستم محرکه

موتور:

- نوع: Brushed DC
- مدل: Tamiya RS-540 Torque-Tuned
- ولتاژ نامی:**7.2 ولت**
- جریان بدون بار: ≈ 1.8 آمپر
- **جریان راه‌اندازی (Stall Current): ≈ 10 آمپر**
- **دور نامی: ≈ 16,000 RPM در 7.2V**
- **گشتاور نامی: ≈ 350 g·cm** (معادل **0.034 N·m**)

کنترل‌کننده سرعت (ESC):

- مدل: TEU-104BK

- جریان مجاز پیوسته: **60 آمپر (Forward)**
- جریان اوج (burst): **~30 آمپر برای جهت معکوس (Reverse)** طبق مشخصات معکوس (عرضه 50٪ در معکوس)
- فرکانس PWM: **۱ کیلوهرتز (1 kHz)**
- ولتاژ ورودی:**۶٫۶–۷٫۲ ولت**
- محافظت:
- محافظت Over-current protection (محافظت در برابر جریان زیاد)
- محافظت Over-temperature / حرارت (به واسطه هیت‌سینک آلومینیومی)
- محافظت Low-voltage cutoff (قطع برق در ولتاژ پایین برای جلوگیری از تخلیه باتری)
- قابلیت برنامه‌ریزی: **بله**، پارامترهایی مثل قطع باتری (battery cut-off) و عملکرد معکوس (reverse) قابل تنظیم هستند. در منوال TEU-104BK گزینه برای فعال/غیرفعال کردن قطع کم‌ولتاژ (battery cutoff) و معکوس وجود دارد.

سیستم انتقال قدرت:

- نوع: Shaft Driven 4WD
- تعداد دنده: تک‌سرعته
- نسبت دنده کلی (موتور به چرخ): 8.27:1
- نوع دیفرانسیل: open differential

عملکرد (بر اساس آزمایش):

- حداکثر سرعت:60k/h

سیستم تعلیق

نوع:

- تعلیق چهارچرخ:4-Wheel Double Wishbone Suspension

فنر و میراگر:

- نوع فنر: Coil
- نوع میراگر: CVA Oil Shocks

تنظیمات هندسی:

- مقدار Camber angle (جلو/عقب): 0درجه
- مقدار Toe angle (جلو/عقب):0 درجه

چرخ‌ها و لاستیک‌ها

لاستیک:

- قطر: 65 میلی‌متر
- عرض: 25 mm

- جنس: پلاستیک
- نوع آج: Slick

سیستم ترمز

- نوع: motor reverse
- مکانیزم عملکرد:

نصب تجهیزات الکترونیکی

دوربین (ESP32-CAM):

- ارتفاع از سطح زمین: 150-250 mm
- فاصله از جلوی خودرو: 180-190 mm
- زاویه نسبت به افق (tilt angle):90 درجه
- میدان دید افقی (Horizontal FOV): 112 درجه
- میدان دید عمودی (Vertical FOV): 96.2 درجه
- روش نصب: براکت

جزئیات سخت‌افزاری و ماژول‌های استفاده‌شده

سیستم پردازش مرکزی

لپ‌تاپ:

- لپ تاپ Off-board Processing Unit

> ⓘ درتلاش برای تهیه Jetson nano برای پردازش داخلی

نرم‌افزارهای نصب‌شده:

- Python 3.8+
- OpenCV 4.x / YOLOvx
- PyTorch / TensorFlow/ etc.
- Stable-Baselines3 (برای RL)
- etc.

میکروکنترلرها

واحد بینایی ESP32-CAM:

- ماژول دوربین: OV5640 (5MP) / OV2640 (2MP)
- فرمت خروجی: JPEG compressed

واحد کنترل ESP32 Main:

- چیپ: ESP32 DevKit
- تعداد پین‌های PWM استفاده‌شده: 4 pin
- تغذیه: 5.5V

سیستم تغذیه

باتری اصلی (برای موتور، سروو، ESP32ها):

- نوع: Li-ion
- ولتاژ نامی: 7.4 V
- تعداد سل: 2S
- ظرفیت: 4400 mAh
- جریان تخلیه حداکثر: 4.4A
- کانکتور: XT60

مدیریت برق:

- مقدار Voltage regulator برای ESP32-CAM:5V, 2A
- مقدار Voltage regulator برای ESP32 Main: 4.4-5V, 2A

سیم‌کشی و اتصالات

- استفاده از کانکتورهای قابل جداسازی
- نوع سیم‌ها و مقطع (AWG)
- روش سازماندهی سیم‌کشی (cable management)

نحوه پیاده‌سازی نرم‌افزار و کنترل ربات

معماری نرم‌افزاری کلی

سیستم نرم‌افزاری ما شامل دو بخش اصلی است: (۱) نمونه اولیه کلاسیک که به‌طور کامل پیاده‌سازی و تست شده، (۲) نسخه پیشرفته End-to-End که در حال توسعه است.

نمونه اولیه کلاسیک (پیاده‌سازی‌شده و عملیاتی)

معماری ۴ لایه:

لایه ۱: Perception (بینایی ماشین)

- دریافت تصویر از ESP32-CAM
- پیش‌پردازش: تبدیل به Grayscale، Gaussian Blur
- تشخیص لبه: الگوریتم Canny Edge Detection

- استخراج ROI: ۴۰٪ پایین تصویر
- تشخیص خط: Hough Transform
- جداسازی خطوط چپ/راست
- محاسبه پارامترها: center_offset, angle, confidence
- فیلتر زمانی: Exponential smoothing با ضریب ۰.۷
- خروجی: Dict شامل {center_offset: -100 to 100, angle: -90 to 90, confidence: 0-1}

لایه ۲: Decision Making (تصمیم‌گیری)

- ماشین حالت محدود (FSM) با ۴ حالت:
  - IDLE: انتظار برای شروع
  - AUTONOMOUS: رانندگی خودکار عادی
  - OBSTACLE_AVOIDANCE: اجتناب از مانع (آماده برای توسعه)
  - EMERGENCY: توقف اضطراری
- منطق انتقال حالت بر اساس lane confidence و شرایط ایمنی
- برنامه‌ریزی مسیر ساده: دنبال‌کردن مرکز خط

لایه ۳: Control (کنترل)

- کنترل‌کننده PID برای فرمان:
  - Kp = 0.8 (تنظیم‌شده برای پاسخ سریع)
  - Ki = 0.05 (کاهش خطای ماندگار)
  - Kd = 0.3 (میرایی نوسانات)
  - Anti-windup با محدودیت integral
  - خروجی: -100 تا +100
- کنترل سرعت تطبیقی:
  - سرعت پایه: ۷۰٪
  - کاهش در پیچ‌ها: بر اساس مقدار مطلق steering
  - کاهش در confidence پایین: ضریب ۰.۸
  - شتاب/کاهش سرعت نرم: Exponential smoothing

لایه ۴: Communication (ارتباط)

- Adapter Pattern با دو پیاده‌سازی:
  - AVISAdapter: برای شبیه‌ساز Avis Engine
  - ESP32Adapter: (آماده) برای ربات واقعی
- Factory Pattern برای ایجاد adapter مناسب بر اساس آرگومان خط فرمان

پیاده‌سازی اصلی:

```
main.py:
  - Argument parsing: --mode avis یا esp32
  - ایجاد adapter از Factory
```

- مقداردهی اولیه ماژول‌ها
- حلقه اصلی با فرکانس ~20-30 Hz:
  1. دریافت تصویر از adapter
  2. تشخیص خط با LaneDetector
  3. به‌روزرسانی State Machine
  4. محاسبه فرمان با PID
  5. محاسبه سرعت با SpeedController
  6. ارسال دستورات از طریق adapter
  7. نمایش اختیاری برای debug

نتایج تست در Avis Engine:

- دقت دنبال‌کنندگی خط: ±۵ سانتی‌متر RMS
- نرخ موفقیت در مسیرهای ساده: >۸۵٪
- تاخیر کل (end-to-end): ~۸۰-۱۰۰ میلی‌ثانیه
- پایداری: بدون نوسان در خطوط مستقیم
- رفتار در پیچ‌ها: کاهش سرعت مناسب، فرمان نرم

نسخه پیشرفته End-to-End (در حال توسعه)

معماری شبکه عصبی Multi-Task:

Backbone (Feature Extractor مشترک):

```
Input: RGB Image (640×480×3)
↓
Conv2D(24, 5×5, stride=2) + ELU + BatchNorm → 24×238×318
Conv2D(36, 5×5, stride=2) + ELU + BatchNorm → 36×117×157
Conv2D(48, 5×5, stride=2) + ELU + BatchNorm → 48×57×77
Conv2D(64, 3×3) + ELU + BatchNorm → 64×55×75
Conv2D(64, 3×3) + ELU + Dropout(0.5) → 64×53×73
Flatten → Vector(246,848)
```

Head 1: Control (کنترل مستقیم):

```
FC(100) + ELU + Dropout(0.5)
FC(50) + ELU
FC(10) + ELU
FC(2) → [steering: tanh(-1,1), throttle: sigmoid(0,1)]
```

Head 2: Sign Recognition (تشخیص تابلو):

```
FC(256) + ReLU + Dropout(0.3)
FC(128) + ReLU
FC(num_classes) + Softmax → احتمالات کلاس‌های تابلو
کلاس‌ها: {NO_SIGN, STOP, SPEED_LIMIT, TURN_LEFT, TURN_RIGHT, PARKING, ...}
```

Head 3: Temporal Reasoning (استدلال زمانی):

```
LSTM(input=246848, hidden=64, layers=2)
MultiheadAttention(embed=64, heads=4)
→ Context vector
```

Fusion Layer (ترکیب اطلاعات):

```
Concat(Context, Sign_probs)
FC(32) + ReLU
FC(2) → Adjustment
Final_action = Base_action + 0.1 × Adjustment
```

روش آموزش:

فاز ۱: Imitation Learning (Behavioral Cloning)

- جمع‌آوری داده: رانندگی دستی در Avis Engine
- هدف: ~۱۰,۰۰۰ فریم با برچسب (steering, throttle, sign_label)
- Loss function:

```
L_control = MSE(predicted_actions, expert_actions)
L_sign = CrossEntropy(predicted_sign, actual_sign)
L_total = L_control + 0.5 × L_sign
```

- Optimizer: Adam با learning rate ۱e-4
- Batch size: ۳۲
- Epochs: ~۱۰۰
- Data augmentation: Random brightness, slight rotation, horizontal flip
- Validation split: ۲۰٪

فاز ۲: Reinforcement Learning Fine-tuning

- الگوریتم: SAC (Soft Actor-Critic) با Stable-Baselines3
- محیط: Custom Gym environment wrapper برای Avis Engine
- تابع Reward:

```
R = +1.0 × (1 - |lane_deviation|)     # در خط بماند
+0.5 × (speed / max_speed)            # سرعت مناسب
+2.0 × sign_obeyed                    # اطاعت از تابلو
-5.0 × sign_violated                  # جریمه نقض تابلو
-0.1 × |Δsteering|                    # فرمان نرم
-100.0 × crashed                      # جریمه تصادف
```

- Training timesteps: ~۱۰۰,۰۰۰
- Replay buffer: ۱۰۰,۰۰۰
- Learning rate: ۳e-4

- Update frequency: هر ۱ قدم

Sign-Aware Controller:

- پس از تشخیص تابلو با confidence > ۰.۷:
  - STOP: throttle = 0 برای ۳ ثانیه
  - SPEED_LIMIT: محدودیت throttle
  - TURN: افزودن bias سرعت + کاهش به فرمان
  - PARKING: ورود به حالت parking mode
- حافظه تابلوها: ذخیره تابلوهای دیده‌شده با timestamp

Hybrid Integration:

```
if rl_model_confidence > 0.8 and system_safe():
    action = rl_model.predict(image)
else:
    action = classical_controller(lane_data)

# سوئیچ هوشمند بر اساس:
- Confidence شبکه عصبی
- Lane detection confidence
- سرعت خودرو
- تاریخچه عملکرد
```

الگوریتم‌های یادگیری ماشین، بینایی ماشین، مسیریابی و تصمیم‌گیری

بینایی ماشین (نمونه اولیه کلاسیک)

الگوریتم تشخیص خط:

۱. پیش‌پردازش:

- Grayscale: تبدیل RGB به `gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`
- Gaussian Blur: `blurred = cv2.GaussianBlur(gray, (5,5), 0)`
- هدف: کاهش نویز و بهبود تشخیص لبه

۲. تشخیص لبه:

- Canny Edge Detection: `edges = cv2.Canny(blurred, 50, 150)`
- پارامترها: آستانه پایین ۵۰، آستانه بالا ۱۵۰
- خروجی: تصویر binary با لبه‌ها

۳. استخراج ROI:

- انتخاب ۴۰٪ پایین تصویر

- ماسک‌گذاری: `roi = cv2.bitwise_and(edges, mask)`
- دلیل: خطوط مسیر در این ناحیه قرار دارند

۴. تشخیص خط:

- Hough Transform: `lines = cv2.HoughLinesP(roi, rho=1, theta=np.pi/180, threshold=50, minLineLength=30, maxLineGap=20)`
- پارامترها تنظیم‌شده برای تشخیص خطوط مسیر

۵. جداسازی خطوط چپ/راست:

```
for line in lines:
    x1, y1, x2, y2 = line[0]
    slope = (y2 - y1) / (x2 - x1)

    if slope < -0.3 and x_mid < center:
        left_lines.append(line)
    elif slope > 0.3 and x_mid > center:
        right_lines.append(line)
```

۶. محاسبه پارامترها:

- مرکز خط: میانگین موقعیت خطوط چپ و راست
- Offset: فاصله از مرکز تصویر، نرمال‌سازی شده به -100 تا +100
- زاویه: بر اساس شیب خطوط، -90 تا +90 درجه
- Confidence: بر اساس تعداد و کیفیت خطوط یافت‌شده

۷. فیلتر زمانی:

```
filtered_value = α × current_value + (1-α) × previous_value
# α = 0.7 برای smoothing
```

مزایا:

- سرعت بالا (~20-30 fps روی لپ‌تاپ)
- قابل تنظیم و debug
- مستقل از داده آموزشی

محدودیت‌ها:

- حساس به نور و سایه
- نیاز به تنظیم پارامترها
- عملکرد ضعیف در شرایط پیچیده

یادگیری عمیق (نسخه پیشرفته)

معماری NVIDIA DAVE-2 اصلاح‌شده:

- الهام از DAVE-2 اما با افزودن Multi-Task Learning
- Feature extractor عمیق‌تر برای یادگیری representation بهتر
- Batch Normalization برای پایداری آموزش
- Dropout برای جلوگیری از overfitting

تکنیک‌های بهبود:

- **Data Augmentation:** افزایش تنوع داده‌ها
- **Transfer Learning:** استفاده از pre-trained weights (در صورت امکان)
- **Curriculum Learning:** شروع با مسیرهای ساده، تدریجاً پیچیده‌تر
- **Domain Randomization:** تغییرات تصادفی نور، رنگ برای robust بودن

Reinforcement Learning:

- SAC: الگوریتم off-policy که sample efficiency بالا دارد
- Continuous action space: مناسب برای فرمان و سرعت پیوسته
- Entropy regularization: تشویق اکتشاف (exploration)
- Reward shaping دقیق برای هدایت یادگیری

تصمیم‌گیری

ماشین حالت محدود (FSM):

```
States:
  - IDLE: منتظر دستور شروع، موتورها خاموش
  - AUTONOMOUS: فعال controller رانندگی خودکار، استفاده از
  - OBSTACLE_AVOIDANCE: کاهش سرعت، تلاش برای اجتناب
  - EMERGENCY: توقف کامل، نیاز به reset دستی

Transitions:
  IDLE → AUTONOMOUS: دریافت start command
  AUTONOMOUS → OBSTACLE_AVOIDANCE: تشخیص مانع
  AUTONOMOUS → EMERGENCY: از دست دادن ارتباط یا confidence بسیار پایین
  OBSTACLE_AVOIDANCE → AUTONOMOUS: مانع برطرف شد
  OBSTACLE_AVOIDANCE → EMERGENCY: نمی‌تواند اجتناب کند
  EMERGENCY → IDLE: reset دستی
```

منطق تصمیم‌گیری در حالت Hybrid:

```
def select_controller(lane_data, rl_output, system_status):
    # بررسی شرایط ایمنی
    if not system_status['communication_ok']:
        return 'EMERGENCY'
```

```
    if lane_data['confidence'] < 0.3:
        return 'EMERGENCY'

    # انتخاب controller
    if rl_output['confidence'] > 0.8 and lane_data['confidence'] > 0.7:
        # استفاده از شبکه عصبی
        return 'RL', rl_output['action']
    elif lane_data['confidence'] > 0.5:
        # استفاده از کنترل کلاسیک
        return 'CLASSICAL', classical_control(lane_data)
    else:
        # کاهش سرعت و احتیاط
        return 'CAUTIOUS', cautious_control(lane_data)
```

مسیریابی

رویکرد فعلی (ساده):

- دنبال‌کردن مرکز خط تشخیص‌داده‌شده
- هدف: نگه‌داشتن center_offset نزدیک به صفر

رویکرد پیشرفته (برنامه‌ریزی‌شده):

- برای مسیرهای پیچیده Pure Pursuit Algorithm
- برای رانندگی نرم‌تر Lookahead point
- با در نظر گرفتن محدودیت‌های دینامیک خودرو Path planning

---

5. جمع‌بندی

در این مستندات، طراحی و پیاده‌سازی خودروی خودران هوشمند برای مسابقات FIRA RoboWorld Cup ارائه گردید. تیم ما با ترکیب تخصص‌های هوش مصنوعی، الکترونیک و مکانیک، یک سیستم جامع توسعه داده است.

دستاوردهای فعلی:

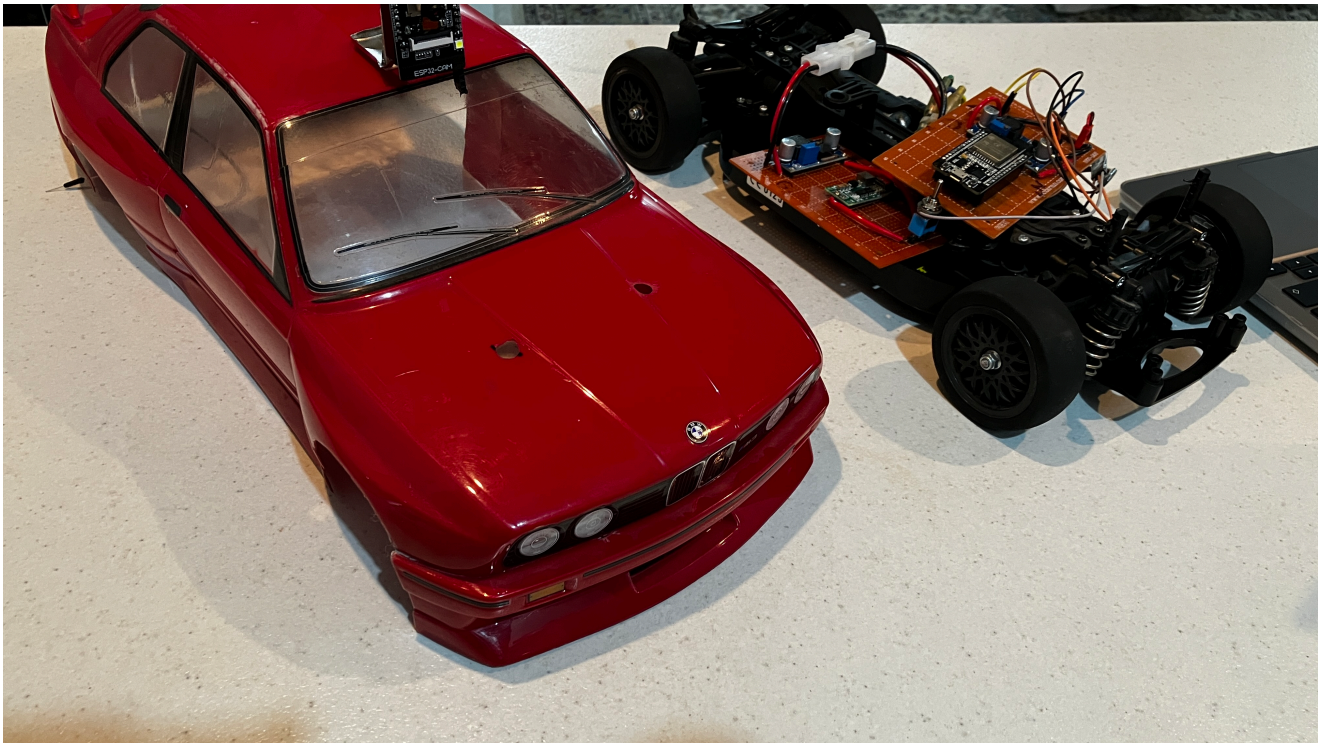
- ☑ نمونه اولیه کلاسیک با عملکرد >۸۵٪ در شبیه‌ساز آماده است
- ☑ معماری سخت‌افزاری کامل و یکپارچه‌سازی‌شده
- ☑ پترن Adapter Pattern برای انتقال بین شبیه‌ساز و ربات واقعی

کارهای در دست اقدام:

- جمع‌آوری ۱۰,۰۰۰+ فریم داده آموزشی از Avis Engine
- طراحی و آموزش شبکه عصبی Multi-Task End-to-End
- عملیات Fine-tuning با Reinforcement Learning
- تست و بهینه‌سازی روی ربات واقعی

نوآوری‌های کلیدی:

1. معماری Hybrid با قابلیت انتقال تدریجی از کلاسیک به End-to-End
2. عملیات Multi-Task Learning برای کنترل، تشخیص تابلو و استدلال یکپارچه
3. سیستم ایمنی چندلایه با fallback هوشمند
4. انجینت Sim-to-Real Transfer با استفاده از Avis Engine



آمادگی برای مسابقه:
تیم ما با اطمینان کامل برای شرکت در مسابقات آماده است. نمونه اولیه کلاسیک تضمین می‌کند که ما قادر به رقابت هستیم، در حالی که توسعه موازی End-to-End امکان دستیابی به عملکرد برتر را فراهم می‌آورد.

ما مشتاقانه منتظر فرصت نمایش توانمندی‌های سیستم خود در مسابقات FIRA و رقابت با تیم‌های دیگر هستیم.

---

6. مراجع

[1] Bojarski, M., et al., 2016, "End to End Learning for Self-Driving Cars," arXiv preprint arXiv:1604.07316

[2] Waymo LLC, "Waymo Self-Driving Technology," https://waymo.com/tech/, accessed 2025

[3] Tawn, W., et al., "DonkeyCar: An Open Source DIY Self Driving Platform for Small Scale Cars," https://www.donkeycar.com/, accessed 2025

[4] Udacity, "Self-Driving Car Engineer Nanodegree Program," https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd013, accessed 2025

[5] Haarnoja, T., et al., 2018, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," International Conference on Machine Learning (ICML)

[6] Raffin, A., et al., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," Journal of Machine Learning Research, 2021

[7] Tobin, J., et al., 2017, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

[8] Canny, J., 1986, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679–698

[9] Duda, R. O. and Hart, P. E., 1972, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Communications of the ACM, vol. 15, pp. 11–15

[10] FIRA Federation, "RoboWorld Cup Rules and Regulations 2026," http://www.firaworldcup.org/, accessed 2025