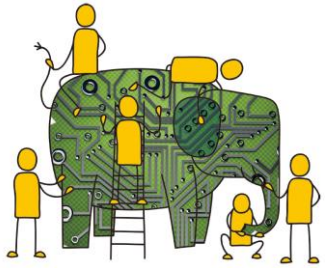
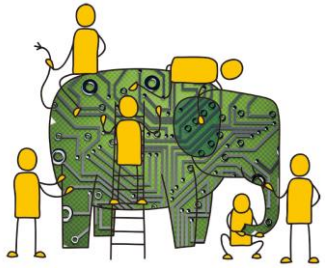


디지털 데믹: 숨겨진 위험들

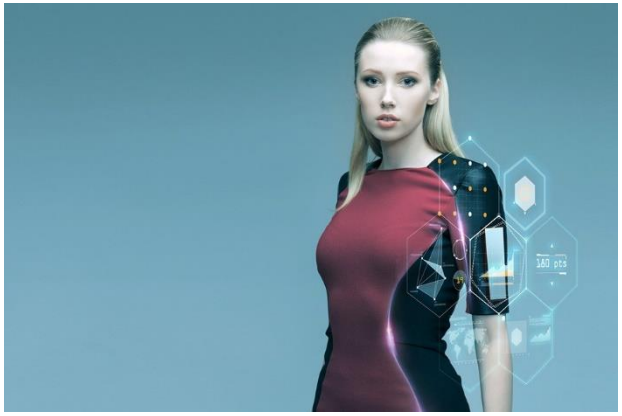
차성덕

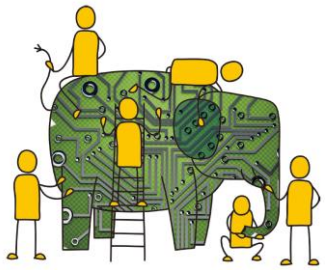
고려대학교 컴퓨터학과

“안전한 시민, 안전한 소프트웨어 ”



소프트웨어가 만드는 " 새로운 세계"





자율주행 자동차

- **On a narrow street, her Tesla automatically tucked in its side mirrors so that it wouldn't scrape other cars.**
- **they're capable of giving bicyclists a wide berth while passing.**
- **…**

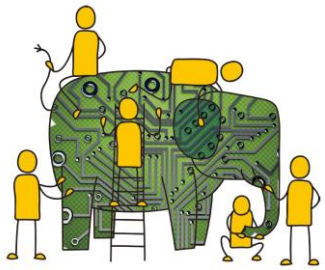
Tesla Is Now Beta-Testing 'Full Self Driving' in Real-World Traffic

Is Tesla FSD truly a Level 5 system if you still have to monitor it, aside from legal liability?

BY JAY RAMEY ◆ OCT 21, 2020

- Tesla rolls out Full Self Driving in limited number of vehicles
- FSD has been promised to offer Level 5 autonomy
- Legal liability remains untested in serious crashes of semi-autonomous vehicles

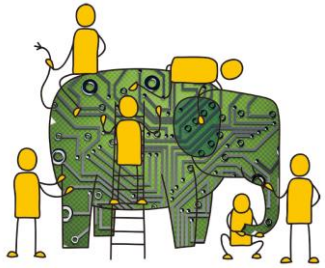
2020년 10월 21일 뉴스



디지털 데믹? 숨겨진 위험?

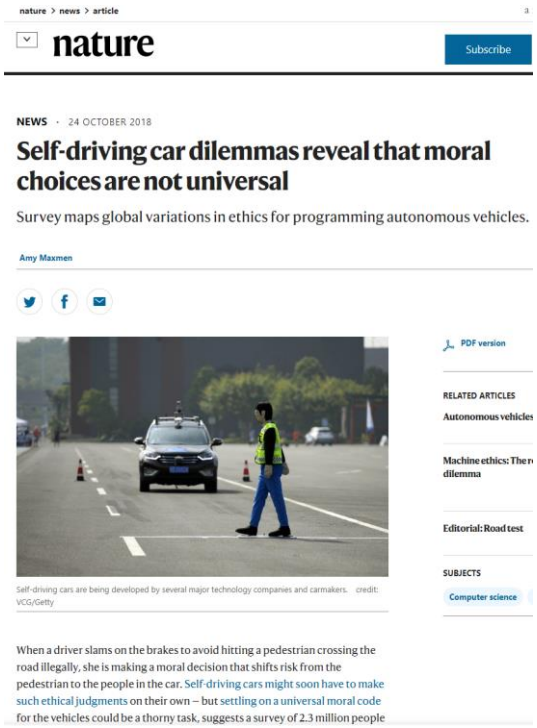
- **Teslas appear to**
  - blow through red lights
  - stop well short of intersections
  - miss turns
  - nearly rear-end a parked car
  - make a turn from the wrong lane
  - speed over speed bumps
- 다른 자율주행 자동차 소프트웨어는 유사한 오류가 없을까?

2020년 10월 30일 뉴스



디지털 데믹과 숨겨진 위험?

- 기술적인 어려움뿐 아니라 윤리적인, 법률적인 문제까지 포함



nature > news > article

nature

Subscribe

NEWS · 24 OCTOBER 2018

Self-driving car dilemmas reveal that moral choices are not universal

Survey maps global variations in ethics for programming autonomous vehicles.

Amy Maxmen

PDF version

RELATED ARTICLES

Autonomous vehicles

Machine ethics: The r dilemma

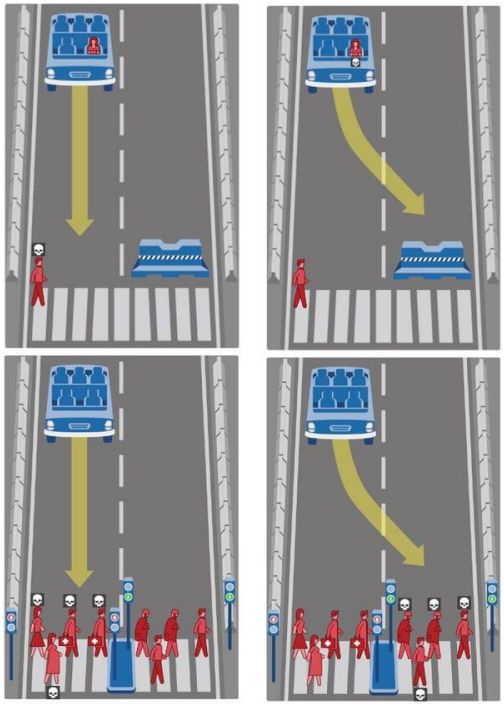
Editorial: Road test

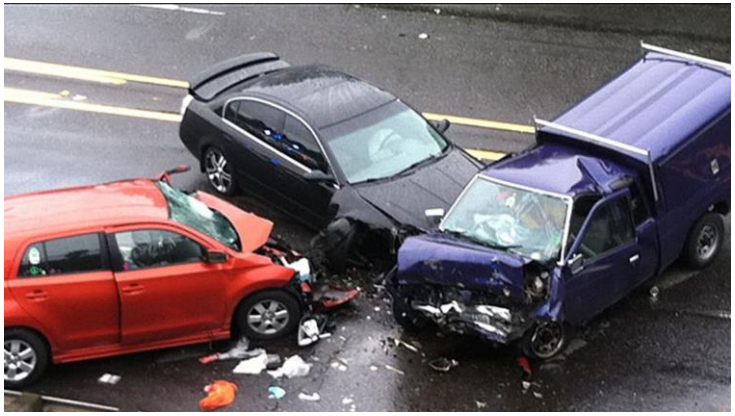
SUBJECTS

Computer science

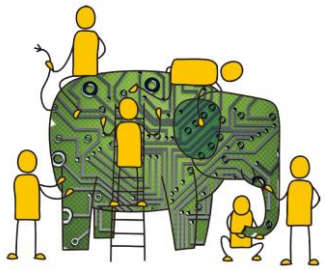
Self-driving cars are being developed by several major technology companies and carmakers. credit: VCG/Getty

When a driver slams on the brakes to avoid hitting a pedestrian crossing the road illegally, she is making a moral decision that shifts risk from the pedestrian to the people in the car. Self-driving cars might soon have to make such ethical judgments on their own – but settling on a universal moral code for the vehicles could be a thorny task, suggests a survey of 2.3 million people



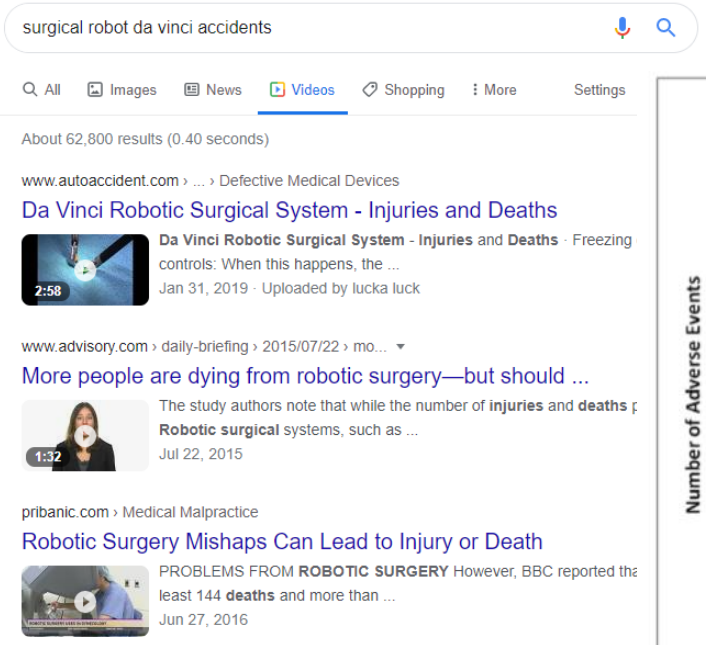


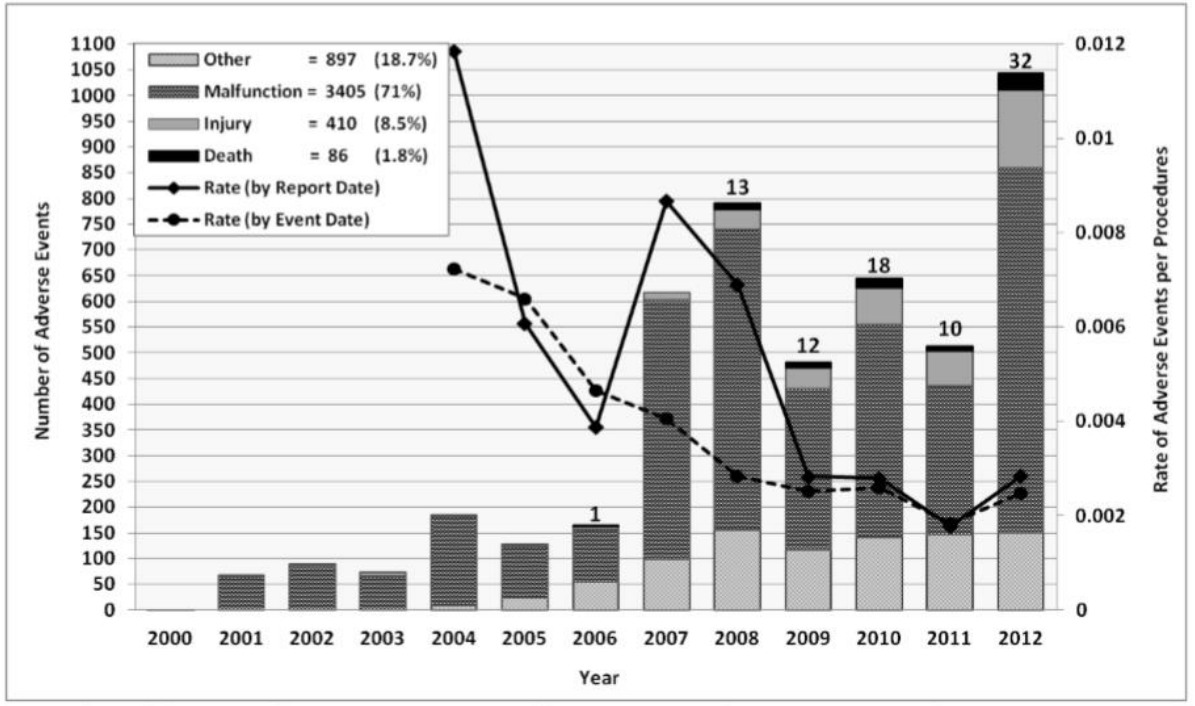
2018년 10월 24일 뉴스



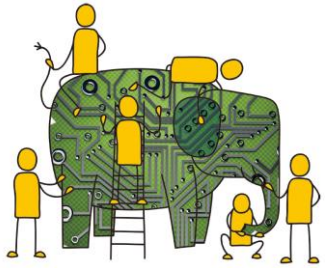
소프트웨어 재난?

- 소프트웨어의 올바른 작동에 의존적인 사회는 피할 수도, 거스를 수도 없는 추세
  - 의료장비, 전력망, 자율자동차, 드론, … 등 사회 거의 전 분야





FDA에 보고된 다빈치 로봇수술 시스템의 사고 사례



소프트웨어 재난 사례

- 사소해 보이는 소프트웨어 오류가 때로는 심각한 피해를 초래함
  - 예: Ariane 5 로켓 폭발

HOME / BLOG / A SPACE ERROR: 370.000.000 $ FOR AN I...

A space error: 370.000.000 $ for an integer overflow

Aleksey Statsenko
Articles: 6

Sept. 2, 2016
#Cpp

Contents

Start. 37 seconds of flight. KaBOOM! 10 years and 7 billion dollars are turning into dust.

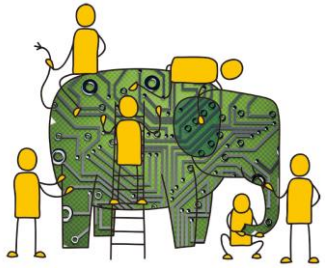
Why visibility matters—the Ariane 5 crash

- Velocity was represented as a 64-bit float
- A conversion into a 16-bit signed integer caused an overflow
- The current velocity of Ariane 5 was too high to be represented as a 16-bit integer
- Error handling was suppressed for performance reasons

*Source: http://moscova.inria.fr/~levy/talks/10enslongo/enslongo.pdf

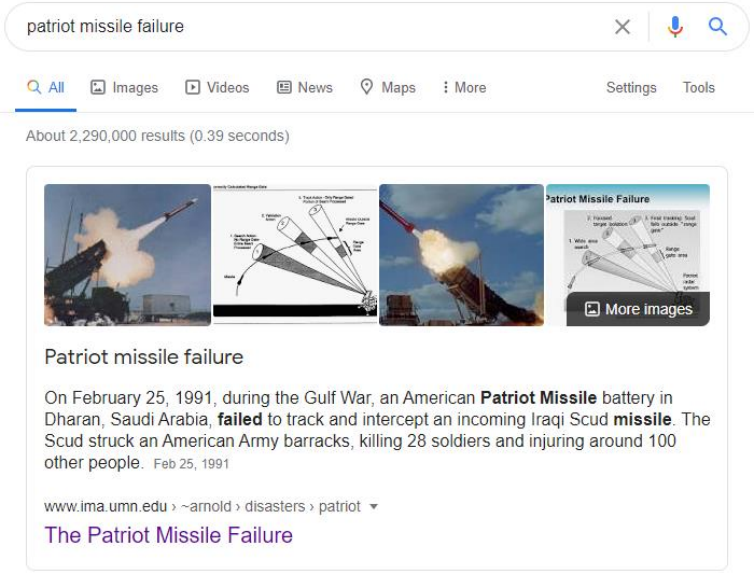
```
-- Vertical velocity bias as measured by sensor
L_M_BV_32 :=
   TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
   G_M_INFO_DERIVE(T_ALG.E_BV));
-- Check, if measured vertical velocity bias ban be
-- converted to a 16 bit int. If so, then convert
if L_M_BV_32 > 32767 then
   P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
   P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
   P_M_DERIVE(T_ALG.E_BV) :=
         UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
-- Horizontal velocity bias as measured by sensor
-- is converted to a 16 bit  int without checking
P_M_DERIVE(T_ALG.E_BH) :=
   UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH) *
   G_M_INFO_DERIVE(T_ALG.E_BH)));
```

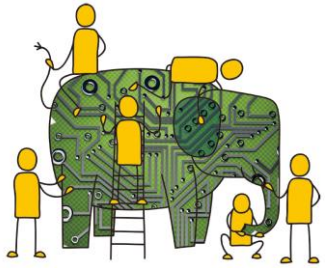
```
L_M_BV_32 := TDB.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
                                G_M_INFO_DERIVE(T_ALG.E_BV));
if L_M_BV_32 > 32767 then
   P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
   P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
   P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M
end if;

P_M_DERIVE(T_ALG.E_BH) := UC_16S_EN_16NS (TDB.T_ENTIER_16S
                                ((1.0/C_M_LSB_BH) *
                                G_M_INFO_DERIVE(T_ALG.E_BH)))
end LIRE_DERIVE;
```



디지털 데믹과 소프트웨어 재난

- 모든 소프트웨어 재난이 반드시 소프트웨어의 오류로 인한 것은 아님을 기억해야 함
  - 패트리어트 미사일의 요격실패는 개발시 가정된 작동환경과 실제 작전환경의 차이로 인해 발생한 오류
- 오류가 없는 "완벽한" 소프트웨어의 개발은 불가능하고 비현실적인 기대





결론

- 디지털 데믹으로 발생할 수 있는 소프트웨어 재난은 우리 모두의 문제
  - 누구에게나, 아무때나 발생할 수 있음
  - 피할 수도 없고, 거스를 수도 없는 시대의 흐름
  - “안전한 시스템”을 요구하는, 그리고 이에 필요한 비용을 지불하는 “깨어있는 소비자”
- "적절한 수준 " 의 안전성 확보를 법제화하는 것이 필요
- 소프트웨어 공학기술의 개발, 안전성 검증 및 평가에 필요한 "국가적인 투자" 필요
  - 안전한 사회 구축을 위한 필수 비용
- 소프트웨어 기반 안전산업의 활성화 및 국제적인 경쟁력 확보