Kognícia a umelý život XIV

Mobilný notebook

Andrej Lúčny – Ondrej Mikuláš

Katedra aplikovanej informatiky

FMFI UK Bratislava

lucny@fmph.uniba.sk - ondrej.mikulas@gmail.com

Kognitivizmus

- Myslenie je manipuláciou so symbolmi
- Hľadá sa univerzálny algoritmus myslenia
- V modeli tvora nájdeme modul predstavujúci kogníciu (kognitívny podsystém)
- Model mysle je nezávislý od spôsobu života, tela i prostredia mysliaceho tvora

Postkognitivizmus

- Neveríme, že existuje jednotná reprezentácia sveta
- Neveríme, že existuje univerzálny algoritmus myslenia
- Percepcia sa prekrýva s akciou
- Myslenie považujeme za závislé na tele (stelesnenosť) a na prostredí (situovanosť)

Situovanosť a stelesnenosť

Pri tvorbe riadiaceho systému robota si môžeme pomôcť tým, že ho ušijeme na mieru:

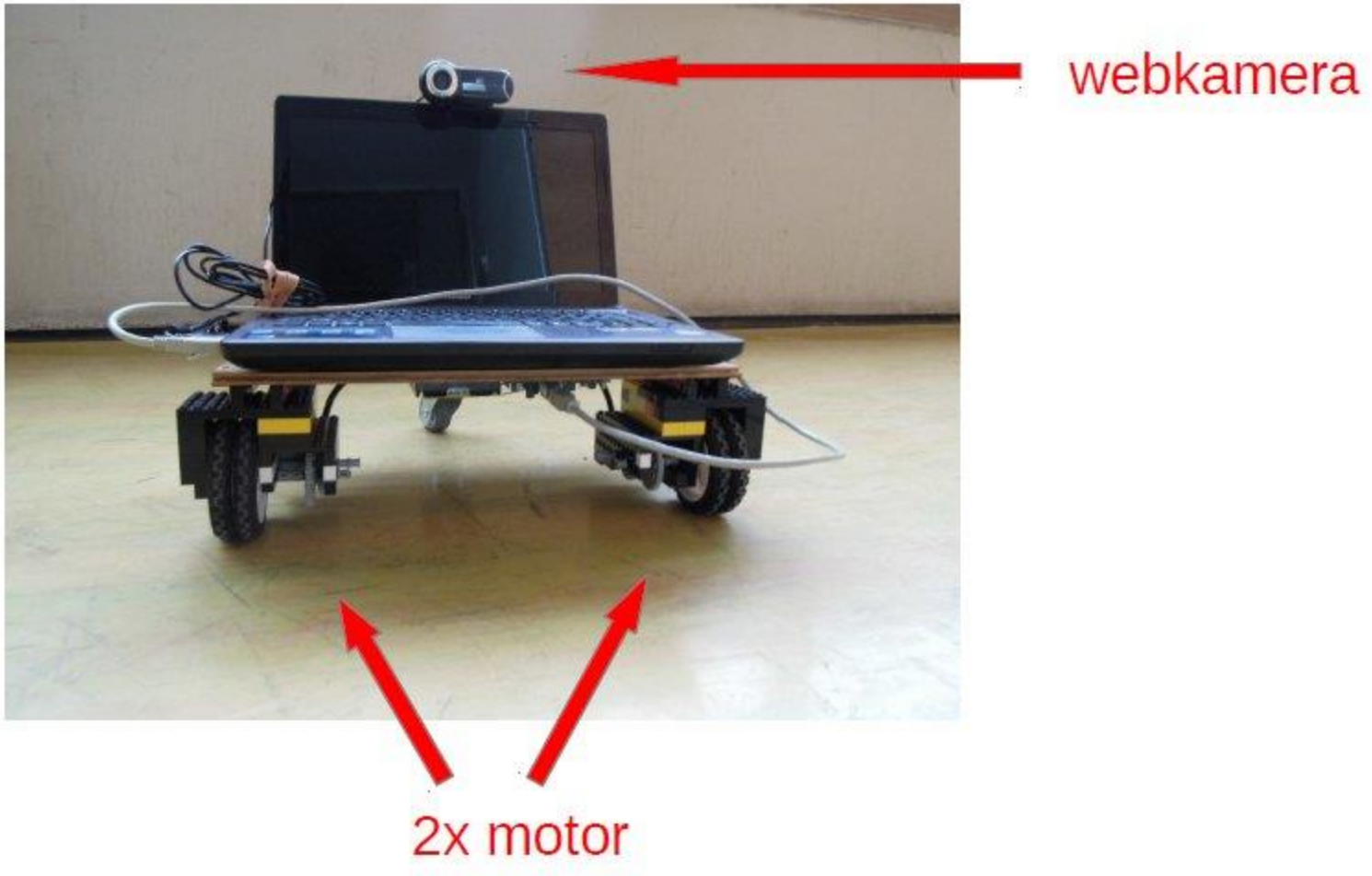
- **prostrediu**, v ktorom robot koná (situovanosť)
- **telu** robota (stelesnenosť)

Úloha

Nekolízny pohyb mobilného robota s dvojkolesovým podvozkom a jednou kamerou v prostredí chodieb na Matfyze

Prostredie

Telo



Senzory a aktuátory

Používame jednoduché senzory a aktuátory, čím kladieme väčšie nároky na riadiaci systém

So sofistikovanejšími senzormi (kamera kinect) a aktuátormi (servomotory s odometriou) je možné vytvoriť 3D model prostredia, vypočíta nekolíznu trasu a potom presne po nej ísť

Živé tvory to ale takto nerobia a nám nejde len o to úlohu vyriešiť, ale vyriešiť ju podobným spôsobom.

Situácia: prejdenie chodbou

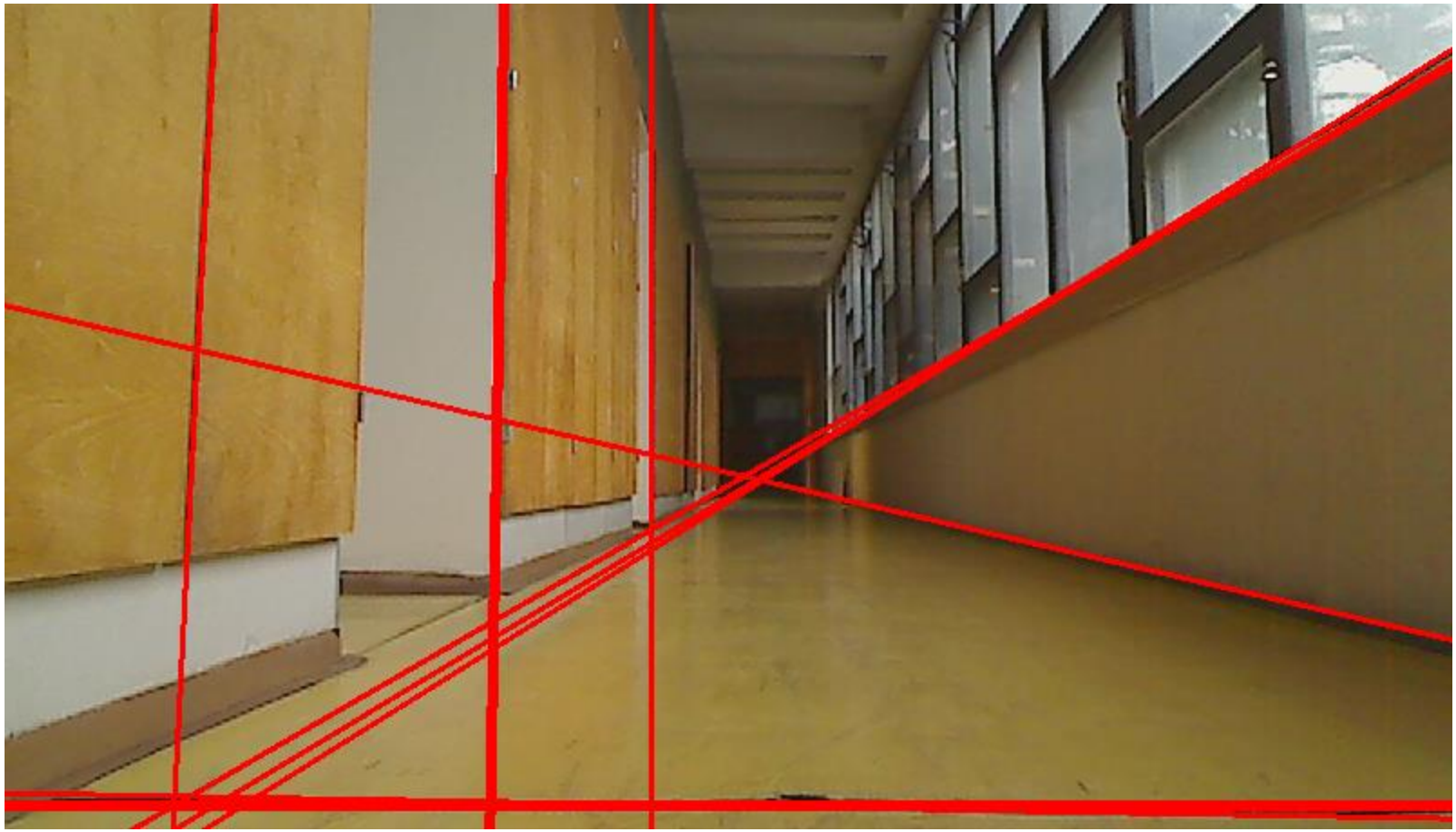
Farebný obraz na čiernobiely

Blur pre potlačenie šumu

Sobelov operátor (1. fáza Canny)

Hrany z algoritmu Canny

Houghova transformácia - priamky

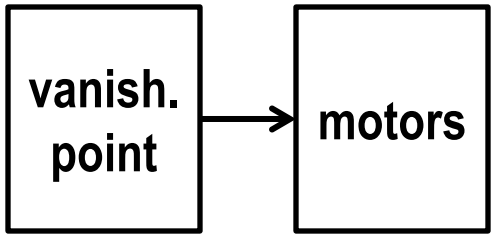


Houghova transformácia - úsečky

Úsečky relevantné pre úbežný bod

Úbežný bod – priesečník

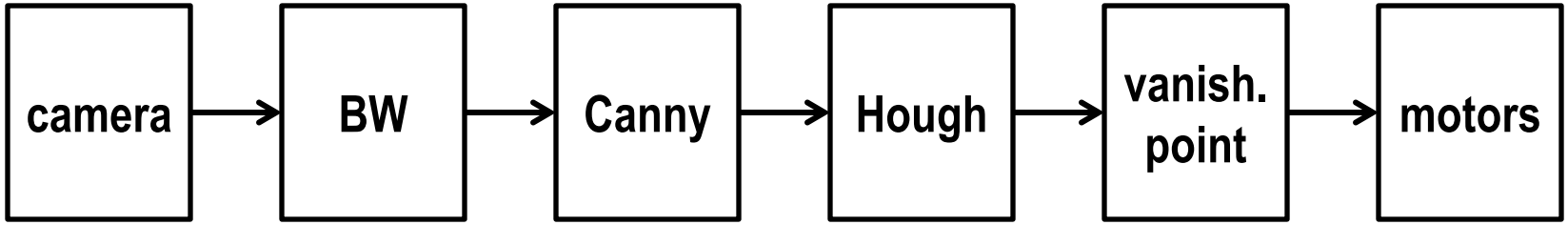
Pohyb robota k úbežnému bodu



Pokiaľ máme úbežný bod príliš na rovnakej strane obrazu, než je na tele robota motor, tento motor zastavíme.

(Inak motor stále ide)

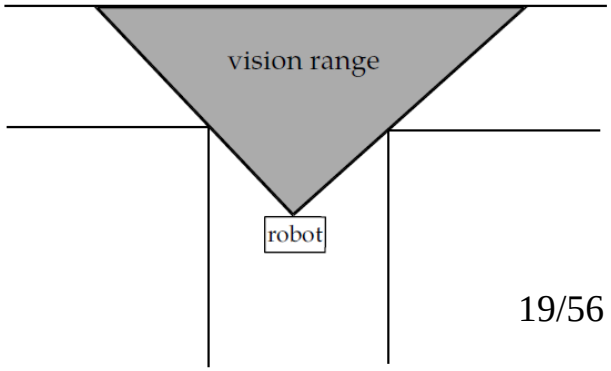
Pohyb robota k úbežnému bodu



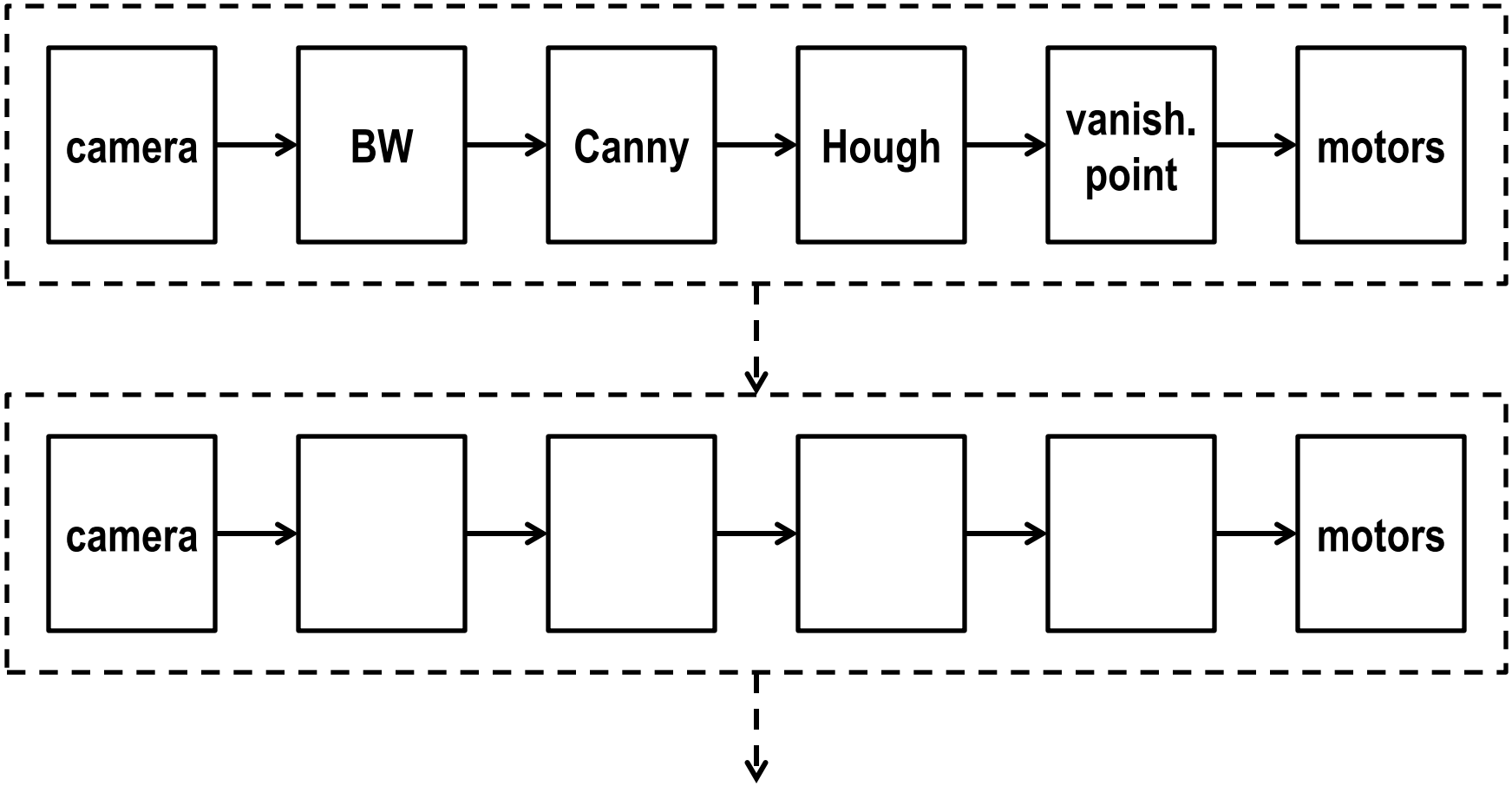
Nadviazaním jednotlivých metód do tzv. pipeline dostávame riadiaci systém, ktorý zvládne úlohu za jednej konkrétnej situácie (prechod chodbou)

Ďalšie správanie

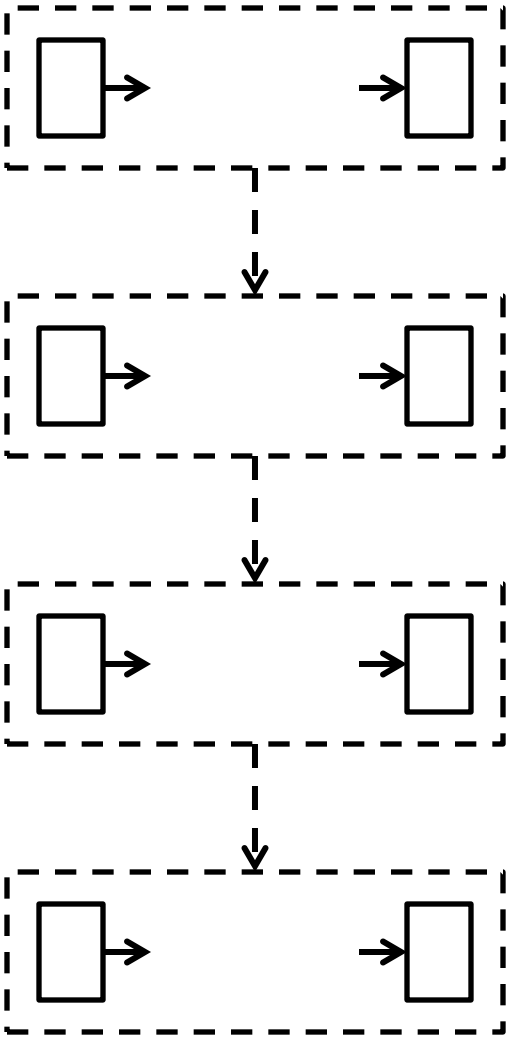
- robot teraz prechádza chodbou
- na konci však stratí úbežný bod a nevie čo si počať
- treba vyriešiť zvládnutie križovatiek chodieb, t.j. zatočenie do susednej chodby
- Ako na to ?

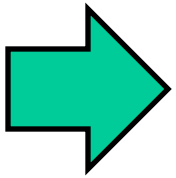


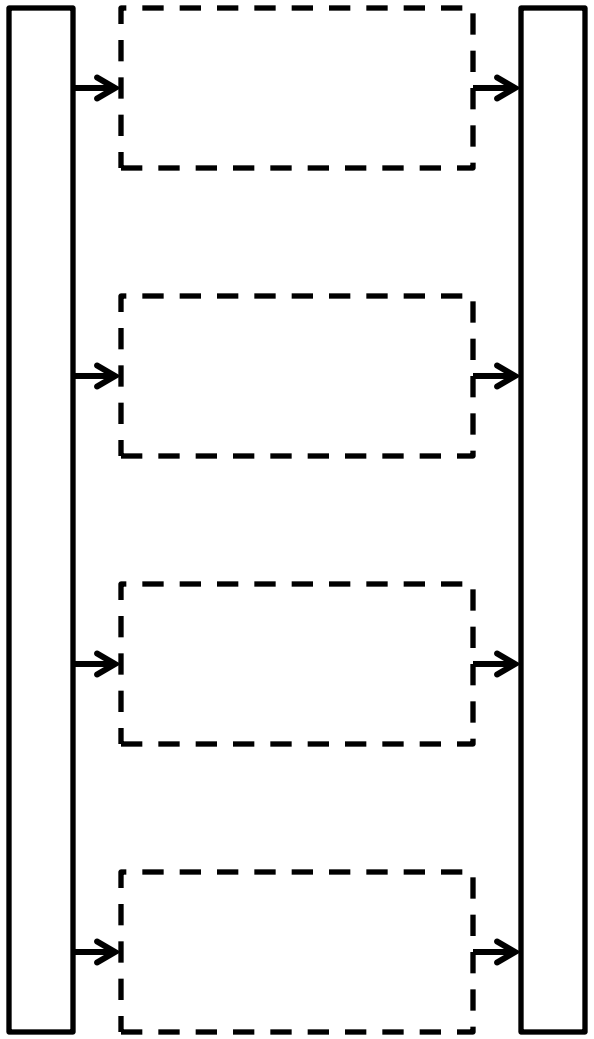
Ako pridať ďalšiu situáciu ?



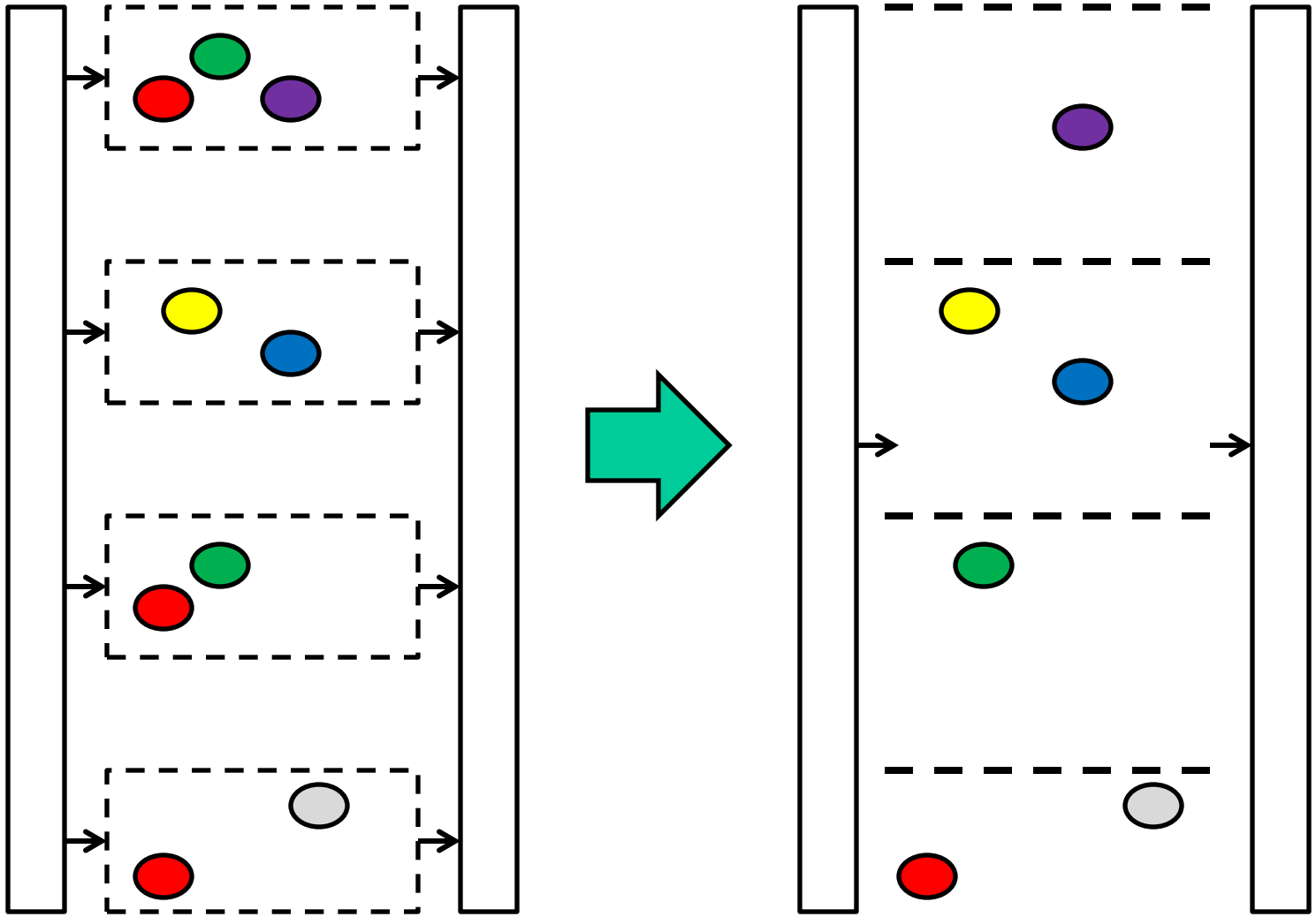
Behaviorálna robotika



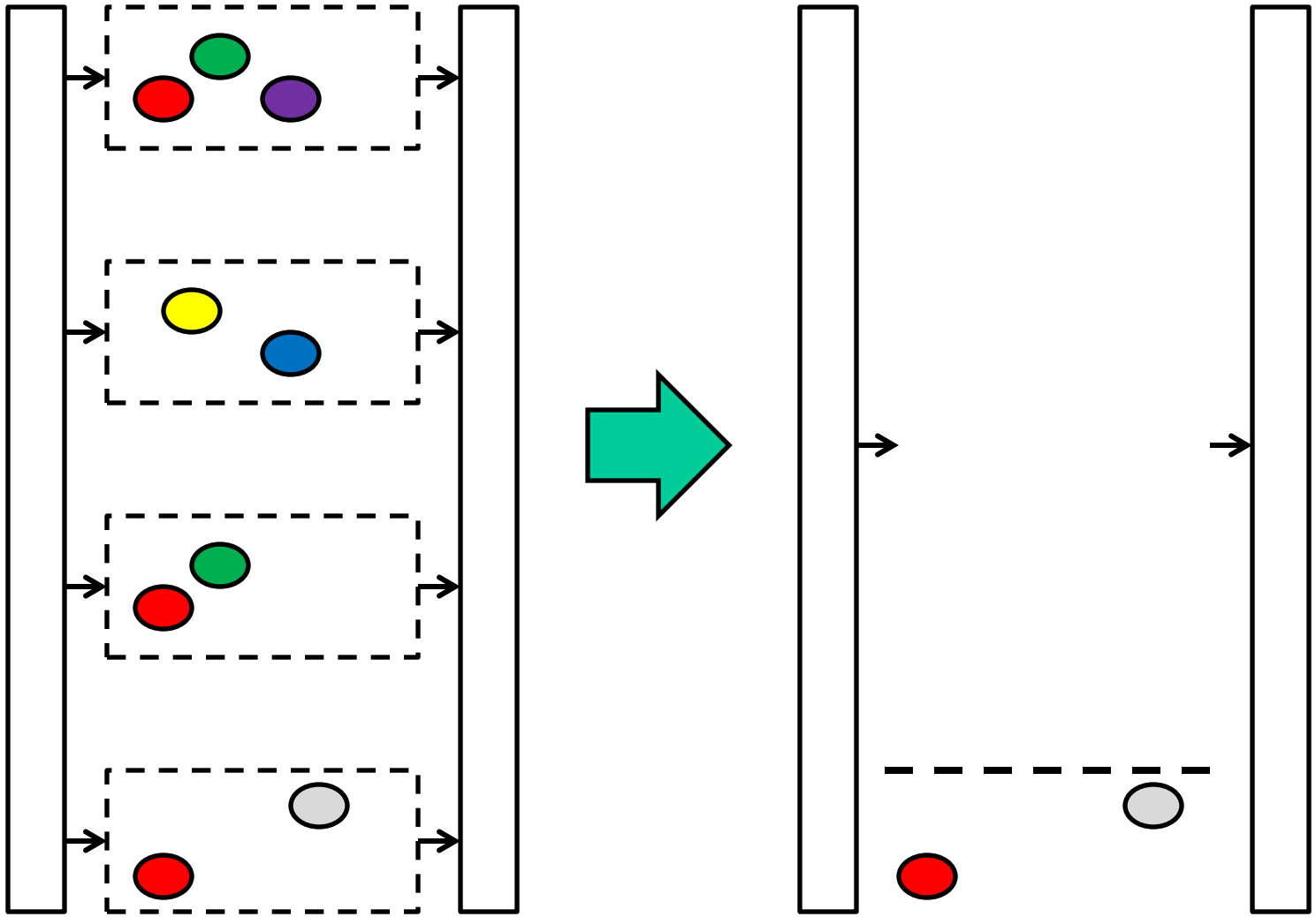




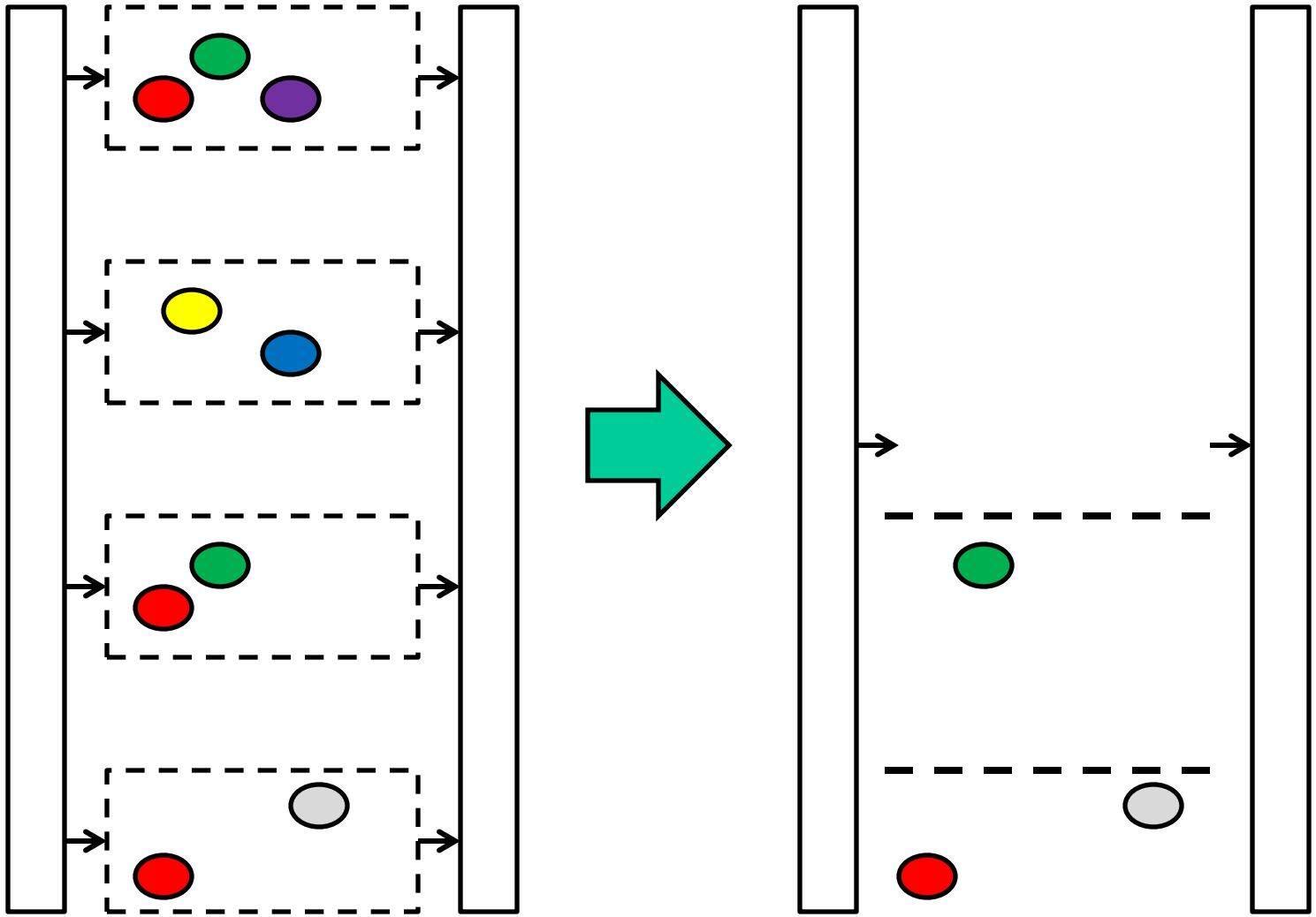
Subsumpčná architektúra



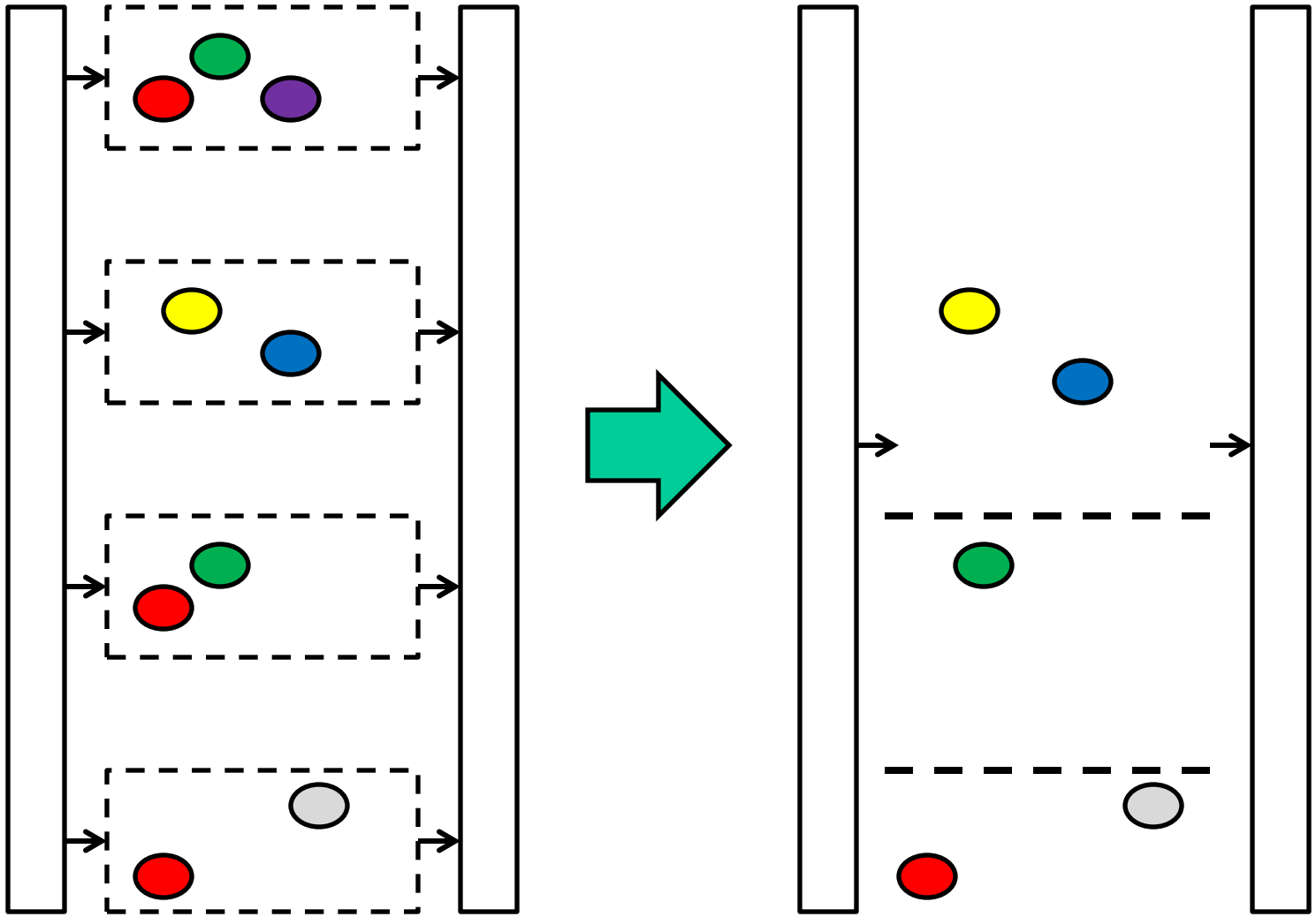
Subsumpčná architektúra



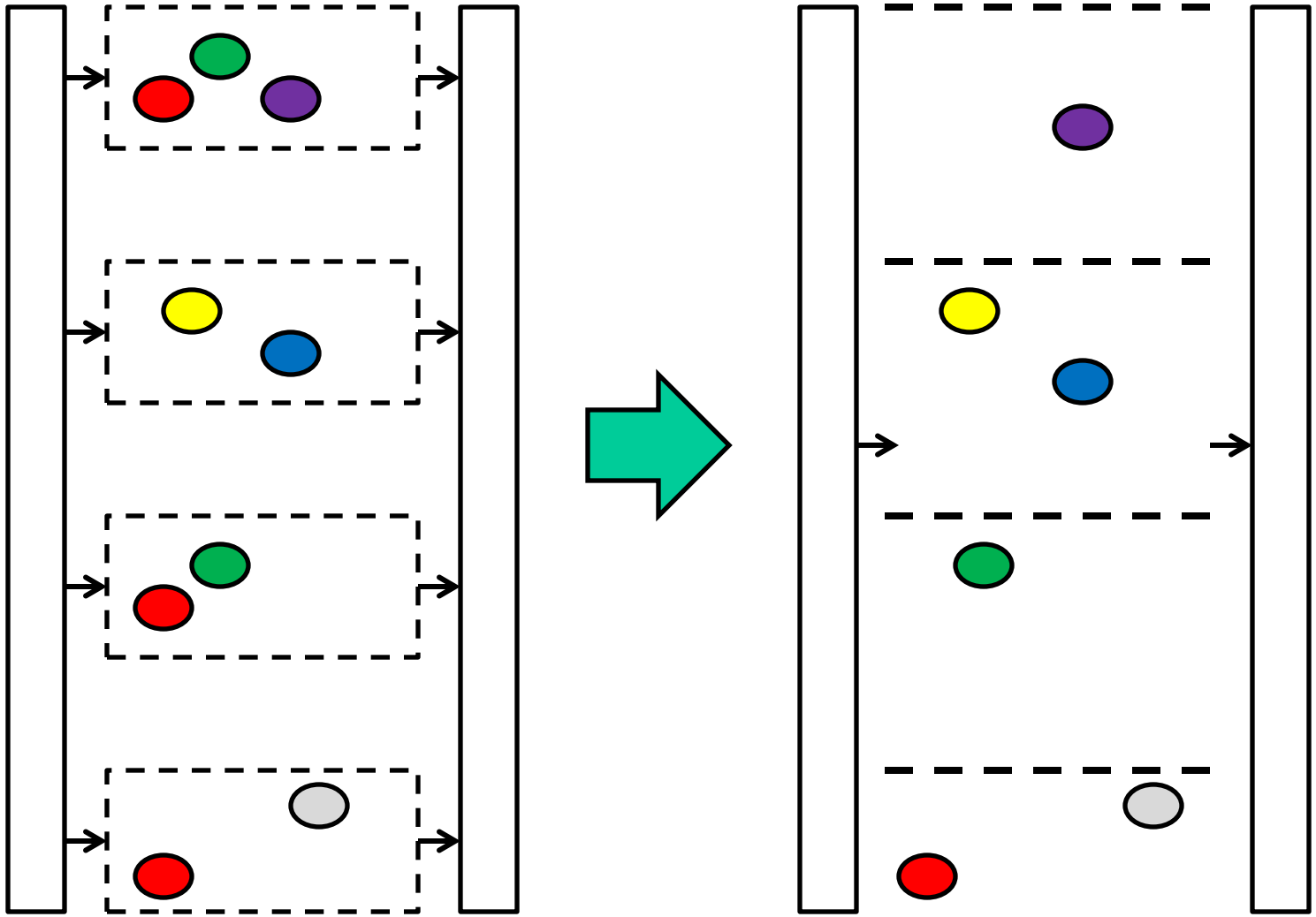
Subsumpčná architektúra



Subsumpčná architektúra



Subsumpčná architektúra



Motivácia

- Typickými vlastnosťami živých organizmov sú paralelizmus a hierarchia
- Táto hierarchia je založená skôr na regulácii než aktivácii

Ak chirurgicky prerušíme miechu úhora pri mozgu, neprestane plávať svojimi typickými sínuoidnými pohybmi, ale naopak pláva stále a úplne pravidelne.

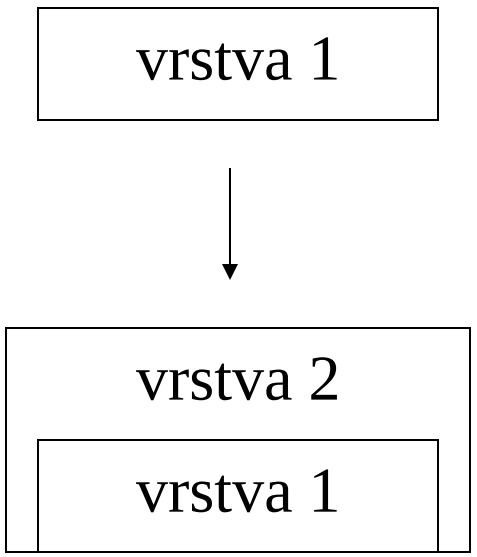
Motivácia

- Štruktúra živých organizmov je výsledkom fylogenetickej evolúcie
- Ich anatómia je podobná lebo ich ontogenéza prebieha v podobných fázach ako fylogenéza

Subsumpcia

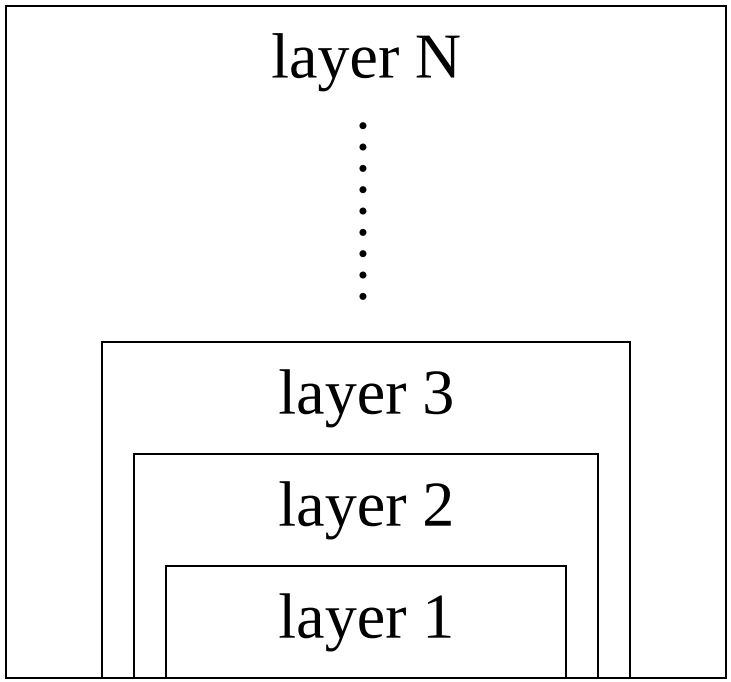
Je založená na evolučnom fakte, že komplexné riadenie pozorované v súčasnosti, má vždy pôvod v jednoduchších predkoch

Vzťah medzi predkom a potomkom je tu však zjednodušený a to tak, že sa predpokladá, že potomok obsahuje presne ten istý riadiaci mechanizmus ako jeho predok, iba k nemu ešte niečo navyše pridáva



Zjednodušenie evolúcie

- T.j. mechanizmus potomka zahrňuje (angl. *subsume*) kompletný mechanizmus jeho predkov
- Na základe toho sa tento princíp volá *subsumption*.

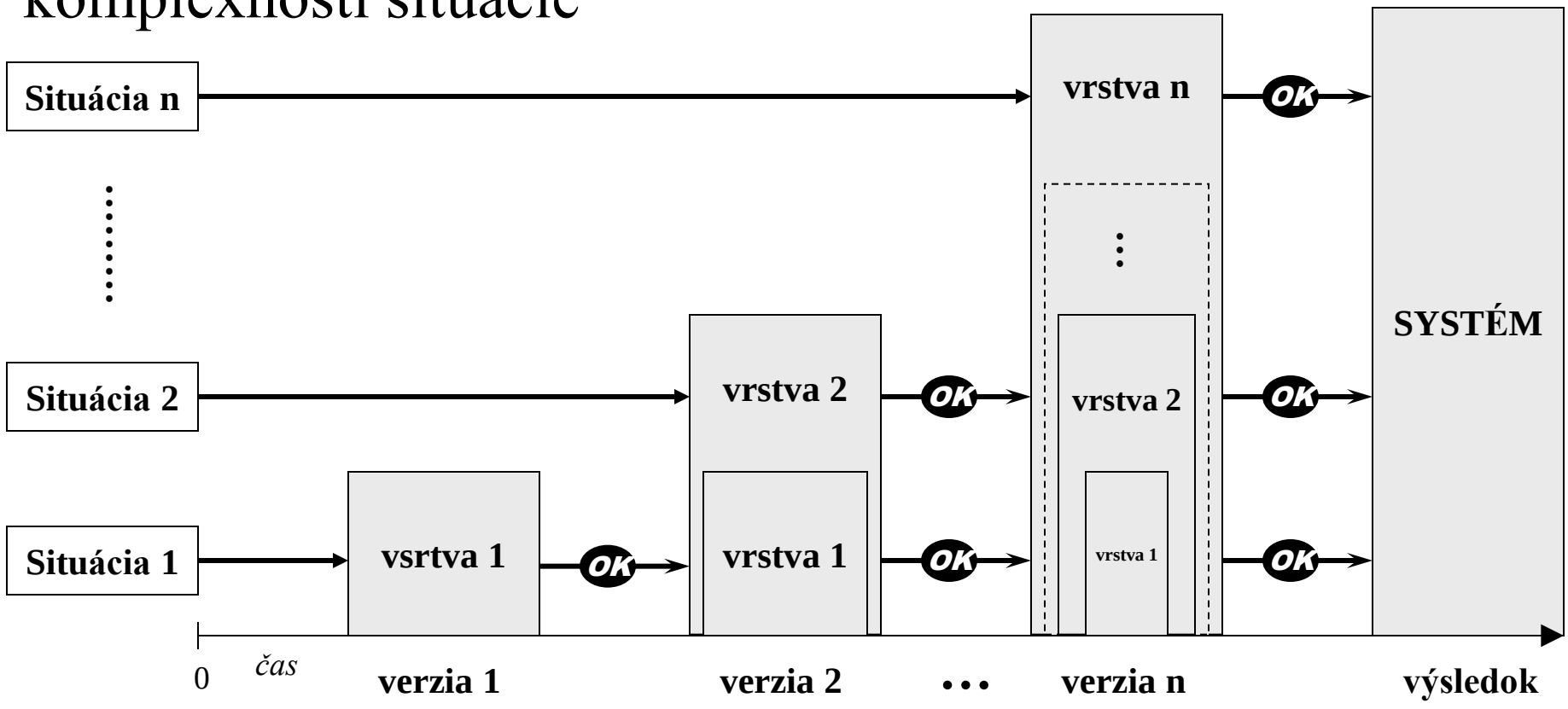


Vývoj pomocou subsumpcie

- Najprv navrhneme vhodné a hlavne dostatočné senzory a aktuátory
- Potom si predstavíme postupnosť evolučných krokov, ktoré by mohli viesť k želanému riadeniu a ktoré začínajú z jednoduchého základu
- Potom postupne vyvinieme štruktúry riadenia zodpovedajúce týmto krokom, pričom jednoduchšiu predchádzajúcu verziu obohacujeme pridaním novej vrstvy
- Od pridaných vrstiev očakávame, že poskytnú novú funkcionalitu, ale nepoškodia tú ktorá už je implementovaná

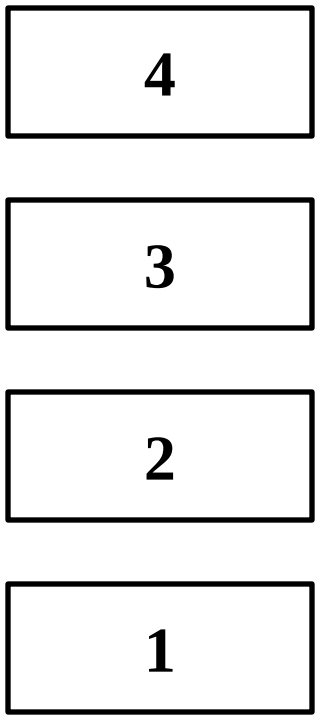
Vývoj pomocou subsumpcie

Situovanosť: evolučným krokom zodpovedá stupňovanie komplexnosti situácie

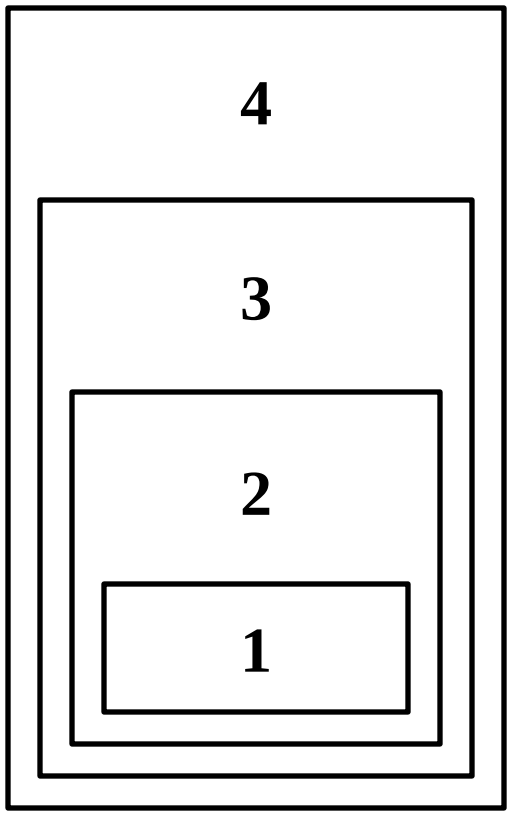


Štruktúrovanie situácii

Klasická robotika



Behaviorálna robotika



Základný implementačný problém

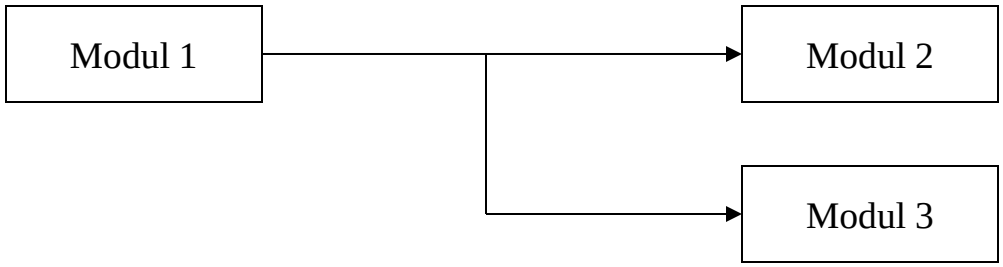
- Ako však môže hierarchicky nižšia – teda evolučne neskoršia - vrstva pôsobiť na hierarchicky vyššiu – teda evolučne skoršiu ?
- Veď pri implementácii tej nižšej sme takúto možnosť neuvažovali a teda sme nevybudovali žiadne rozhranie, ktorým by mohla vyššia vrstva na nižšiu vplývať

Subsumpcia a modularita

- Riešenie: systém musí mať takú modularitu, pri ktorej je toto rozhranie implicitne prítomné

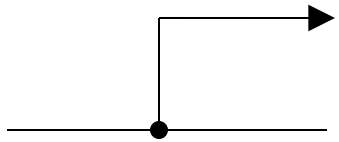
Subsumpčná architektúra

- Systém sa skladá z autonómych modulov (Augmented Finite State Automata)
- Tie sú prepojené komunikačným vedením, po ktorom sú prenášané správy

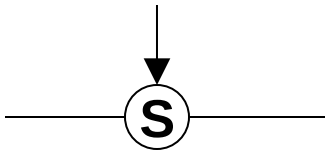


Subsumpčná architektúra

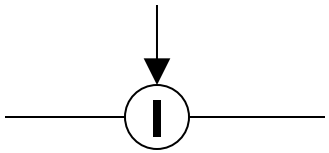
- Vyššia vrstva môže s nižšou manipulovať pomocou:
  - odpočúvania



  - supresie

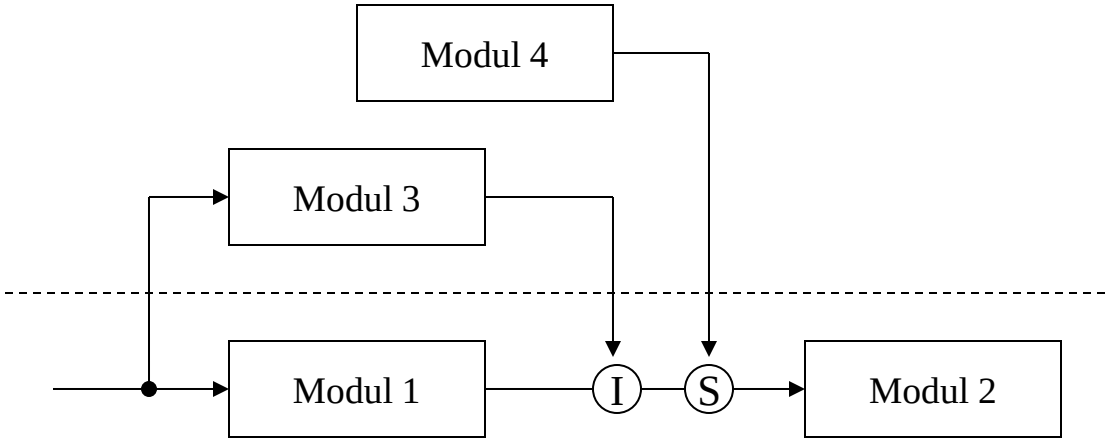


  - inhibície

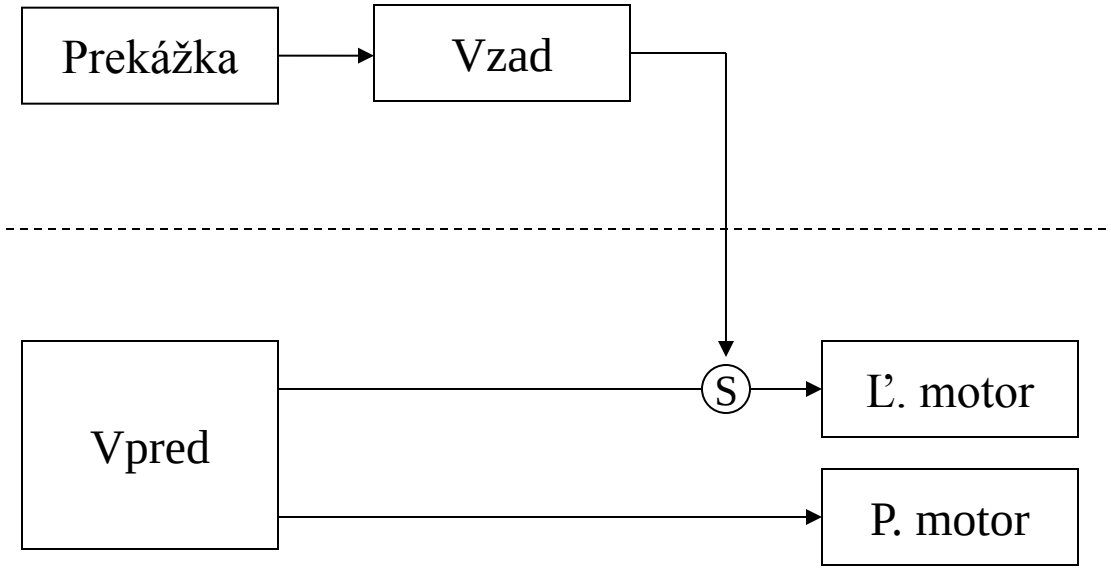


Subsumpčná architektúra

- Pri pridaní novej vrstvy použijeme tieto mechanizmy na zavedenie kooperácie

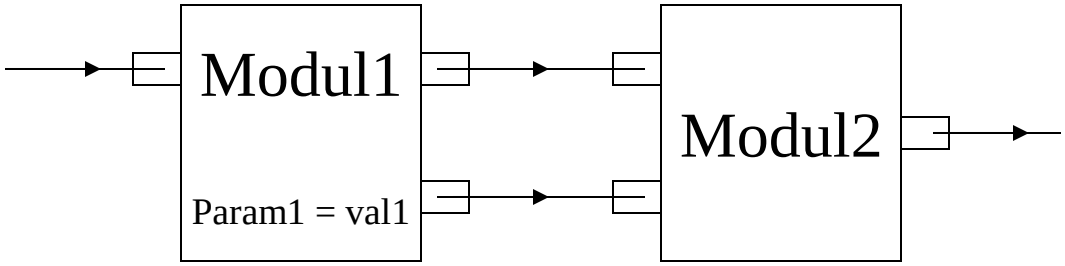


Jednoduchý príklad



Tradičná implementácia

- riadenie sa skladá z modulov s pevne definovanými vstupmi a výstupmi a nastaviteľnými parametrami
- skladanie systému sa realizuje vhodným prepojením vstupov a výstupov

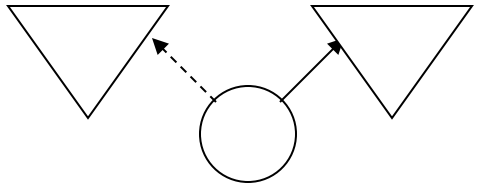


- vykonávanie kódu, ktorý v module prepočíta vstupy na výstupy má na starosti plánovač

Problémy tradičnej implementácie

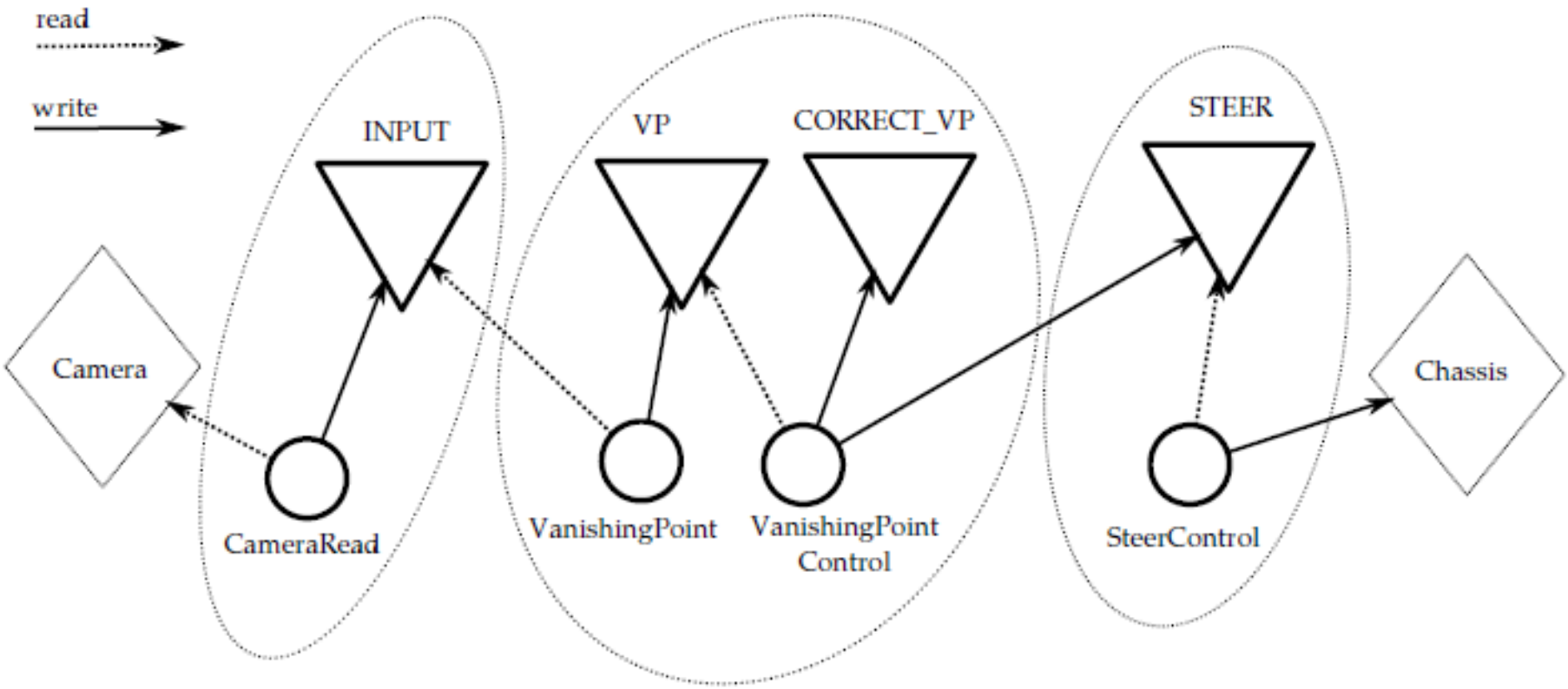
- keď boli kombinované pomalé moduly s rýchlymi
- keď mali vstupy do modulu rôznu frekvenciu
- keď bolo treba dynamicky meniť počet vstupov a výstupov
- keď bolo treba časovo ohraničiť platnosť nejakého údaju
- keď bolo treba zabezpečiť netriviálnu koordináciu medzi paralelnými subsystémami

Architektúra Agent-Space



- spojenia medzi modulmi nahradíme za pomenované dáta na „čiernej tabuli“ (space), tzv. bloky, ktoré realizujú nepriamu komunikáciu medzi agentami
  - agenty ich môžu čítať, zapisovať a mazať na základe ich pomenovania
  - pri zápise môže agent pre blok definovať jeho časovú platnosť a prioritu

Naša úloha podľa Agent-Space



Späť k úlohe

- Pokiaľ sa úbežný bod stratí, robot pôjde vpred naslepo (to ešte urobí základné správanie), pričom sa začne otáčať (nemá krk, takže sa musí otáčať celý)
- Keď sa pritom nejaký úbežný bod, otáčanie ustúpi a robot bude pôjde za novým úbežným bodom

Ako pokračovať na slepo

- Ako zariadime, že keď úbežný bod stratíme, robot pôjde naslepo vpred ?
- Pri stratégii, že podľa úbežného bodu motory zastavujeme (preferujeme reguláciu pred aktiváciou), zariadi sa to samo.

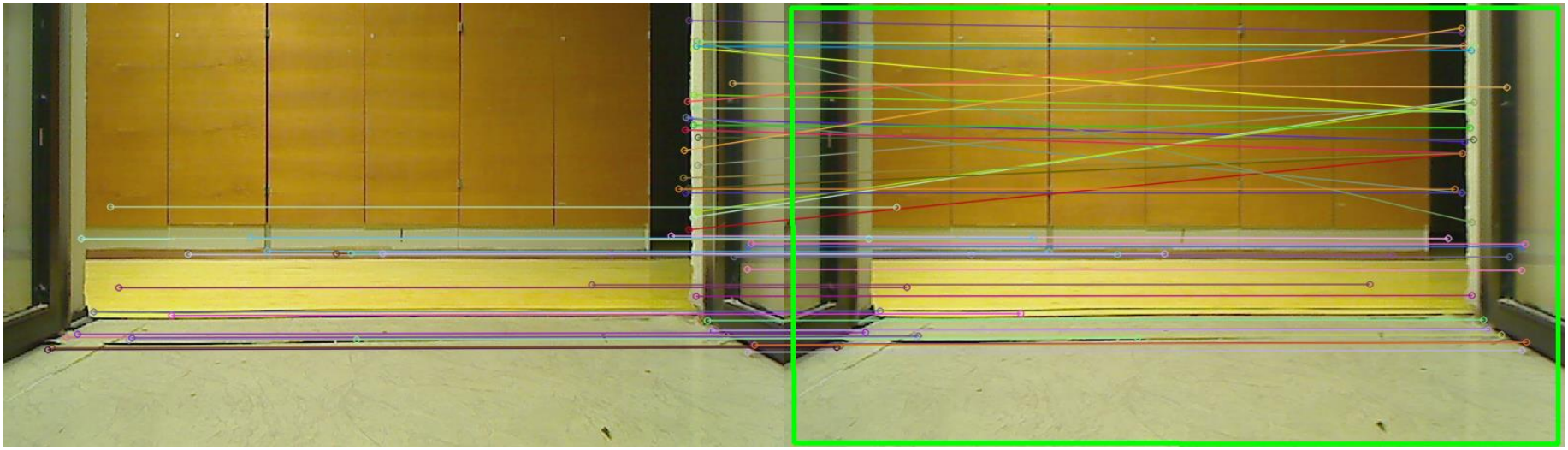
Ako sa otáčať

- Na základe straty úbežného bodu sa aktivuje nové správanie, ktoré sa raz za určitý čas votrie do základného správania (t.j. do pohybu naslepo vpred)
- Pomocou supresie (prioritného zápisu) umlčíme pôvodné príkazy pre motory a točíme sa na mieste na jednu či na druhú stranu (protichodným pohybom motorov)

Ako sa otáčať

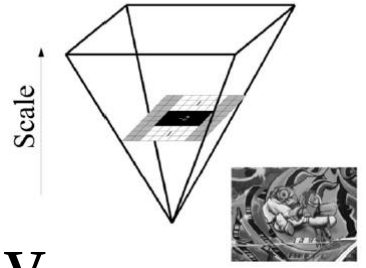
- Smerom do strany možno otáčanie načasovať, t.j. stačí merať čas
- Motory sú ale nepresné, takže vrátiť sa týmto spôsobom späť do správneho smeru je problém
- Používame preto vizuálnu informáciu

Ako zistiť, že sa pohľad vrátil

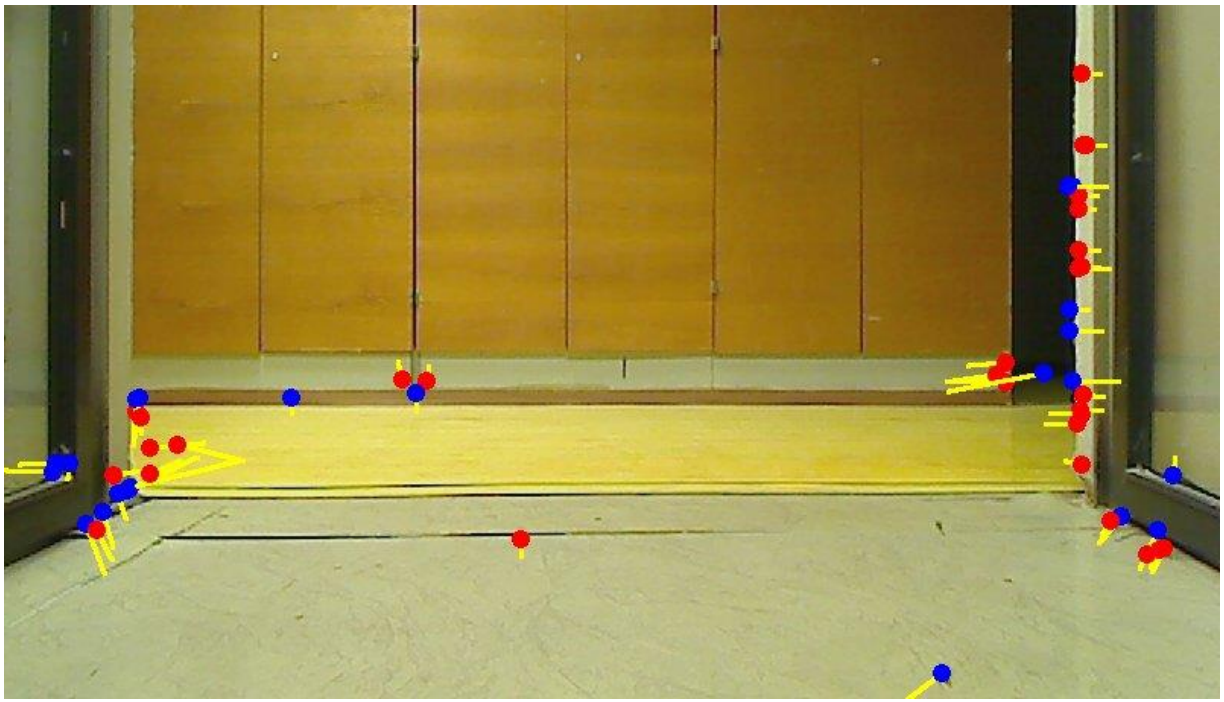


- Potrebujeme z dvoch obrázkov povedať, či ide o ten istý pohľad, respektíve akému otočeniu zodpovedá rozdiel

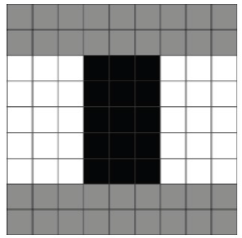
SURF (speeded up robust features)



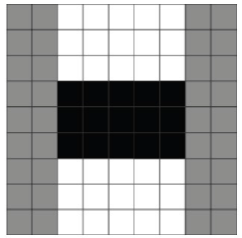
- vyberie význačné body, napr. rohy, škvrny, ...



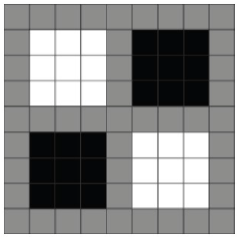
dxx =



dyy =

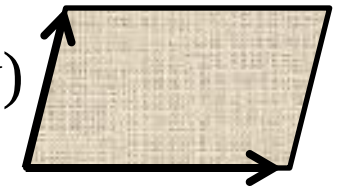


dxy =

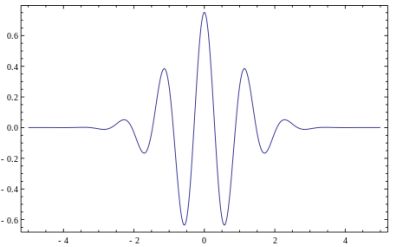


- detektor bodov – lokálne maximá hessiánu v trojrozmenom priestore podvzorkovaného obrazu

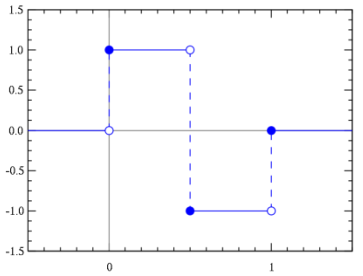
(dxx,dxy)



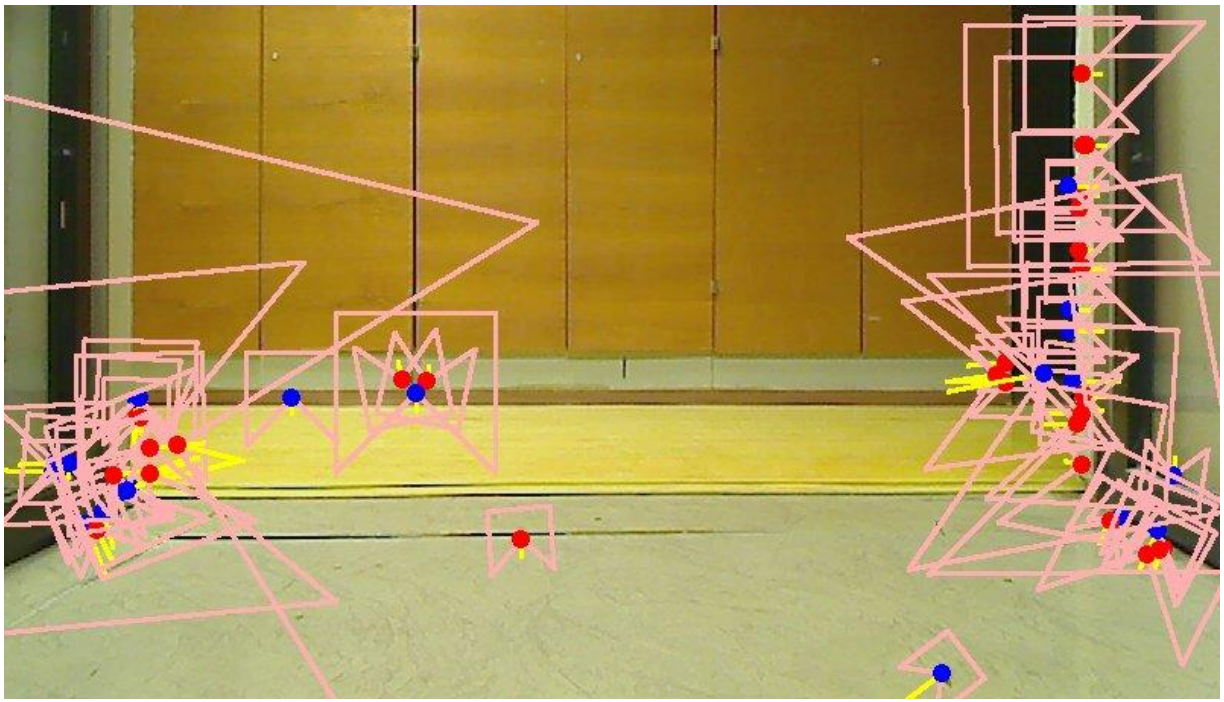
(dxy,dyy)

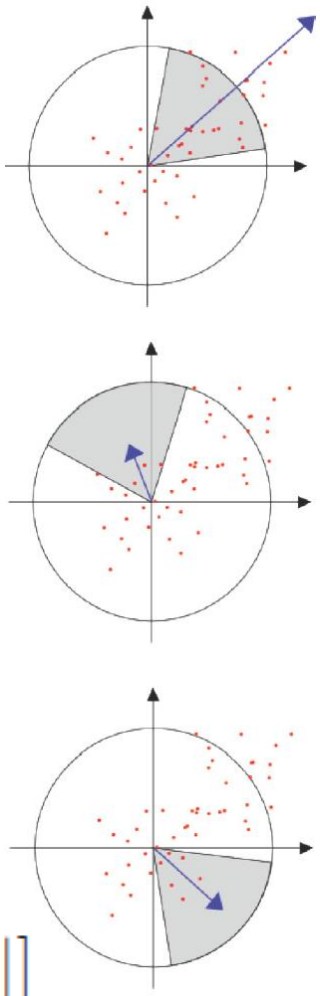


SURF - deskriptor



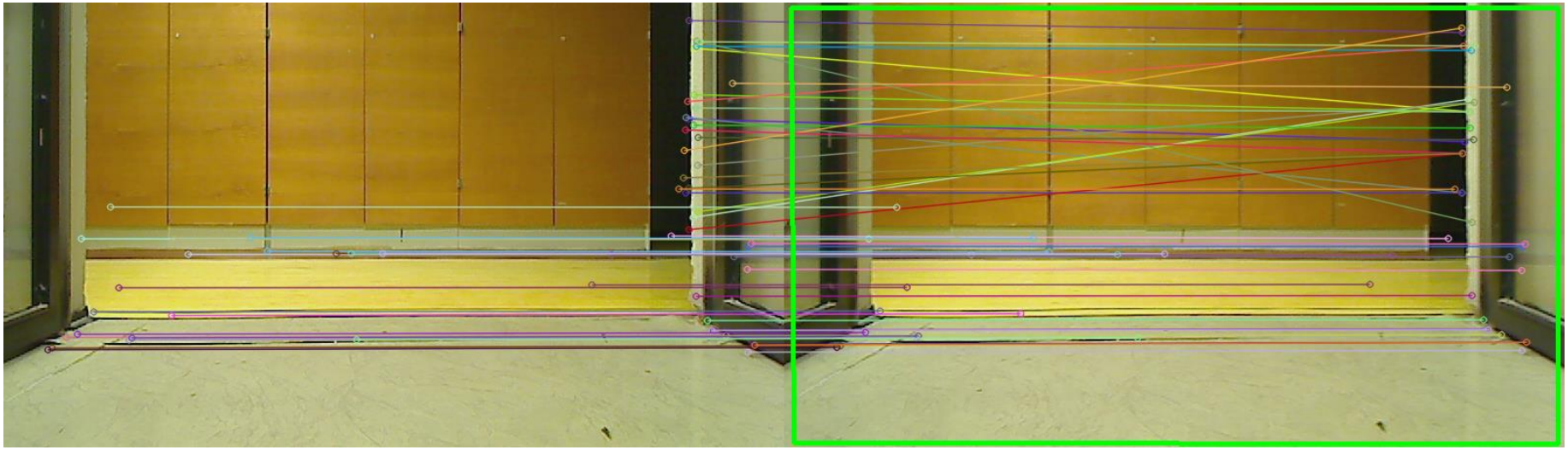
- dá orientovaný popis okolia každého bodu





- Používa Haarov waveletový filter
- 64 hodnotový vektor popisujúci 4x4 regiónov okolia bodu $[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|]$

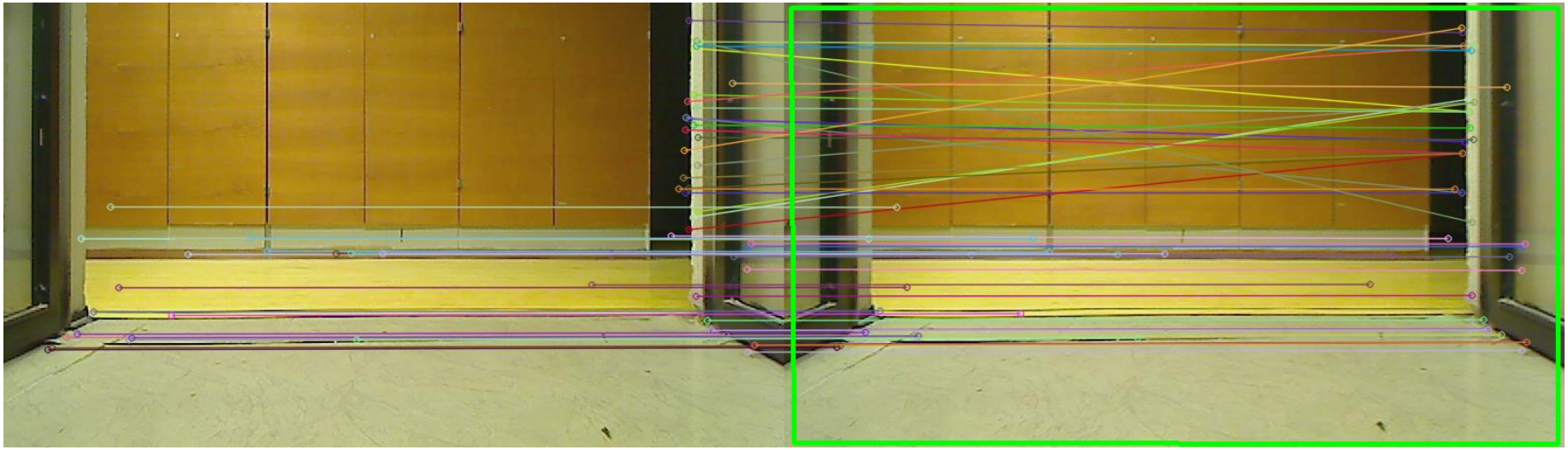
Feature point matching



- Pravdepodobnostný algoritmus, ktorý páruje body v dvoch obrazoch podľa podobnosti ich deskriptorov

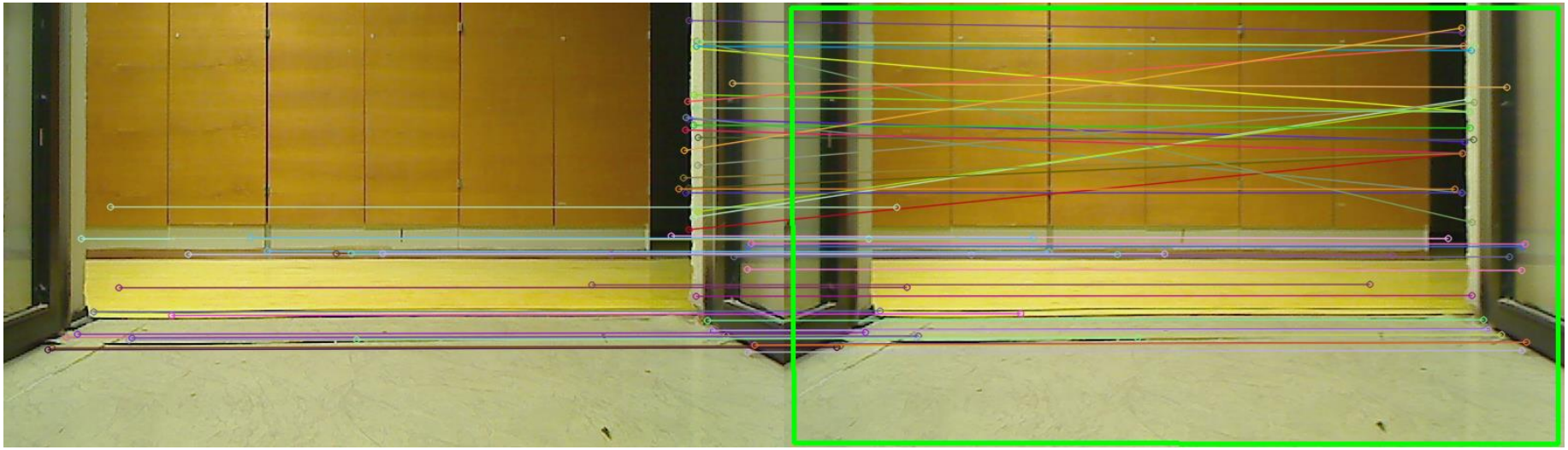
Projektívna transformácia

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



- Hľadáme parametre transformácie, ktorá zobrazí sadu bodov na body k nim spárované
- Algoritmus RANSAC (RANdom SAmple Consensus)

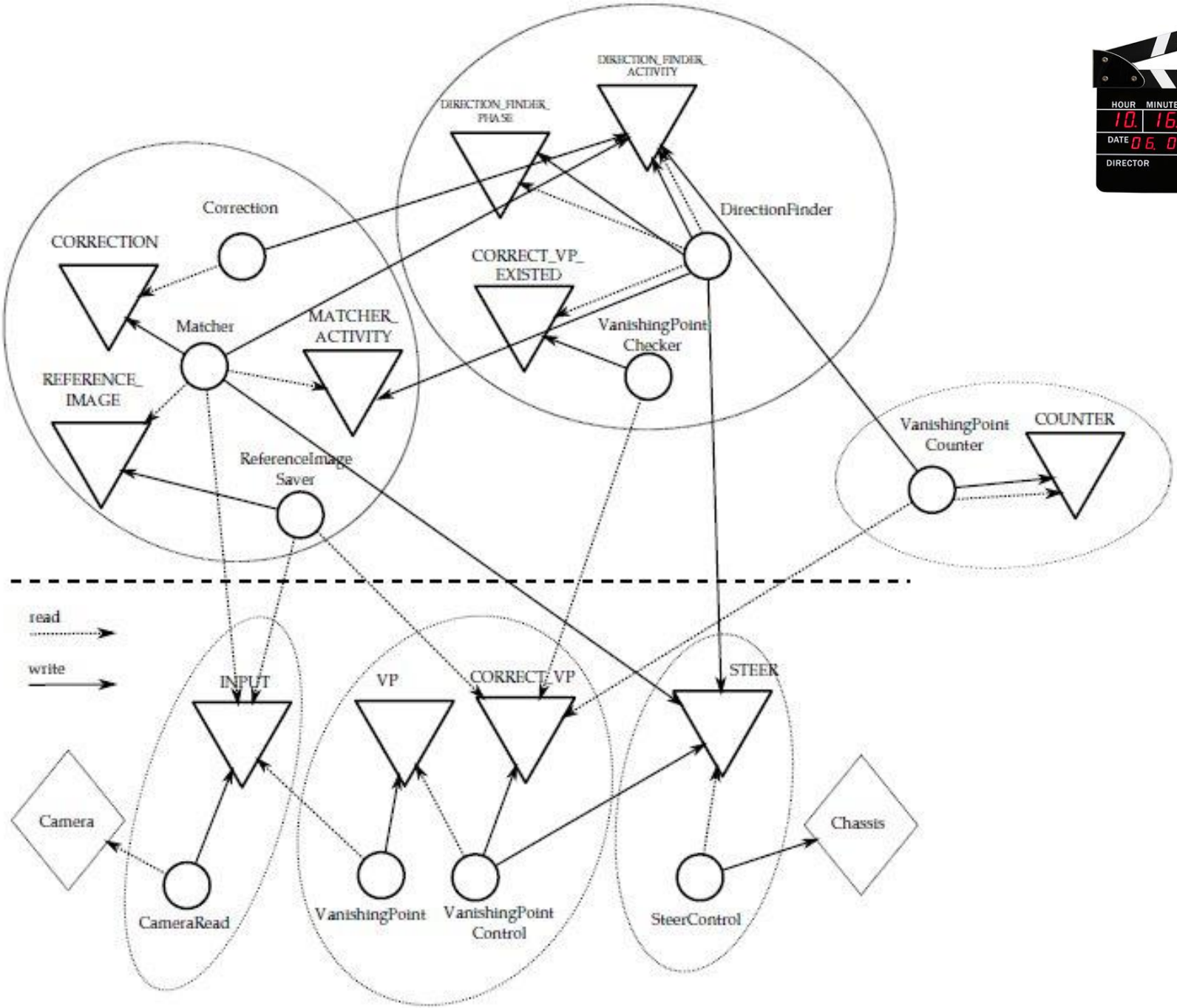
Identifikácia rovnakého pohľadu



- Na základe umiestnenia rohov jedného obrazu na druhom sa rozhodneme, či ide ten istý pohľad

Počítanie úbežných bodov

- Prvý nájdený úbežný bod nie je optimálny
- Nájdené body preto počítame a až po určitom počte a rozhodneme „vypnúť“ otáčanie


DIRECTION_FINDER_ ACTIVITY
DIRECTION_FINDER_ PHASE
DirectionFinder
Correction
CORRECTION
CORRECT_VP_ EXISTED
Matcher
MATCHER_ ACTIVITY
VanishingPoint Checker
REFERENCE_ IMAGE
ReferenceImage Saver
VanishingPoint Counter
COUNTER
read
write
INPUT
VP
CORRECT_VP
STEER
Camera
Chassis
CameraRead
VanishingPoint
VanishingPoint Control
SteerControl

Ďakujem za pozornosť !

Kognícia a umelý život XIV

Mobilný notebook

Andrej Lúčny – Ondrej Mikuláš

Katedra aplikovanej informatiky

FMFI UK Bratislava

lucny@fmph.uniba.sk – ondrej.mikulas@gmail.com