Gymnázium Jana Keplera
Parléřova 2, Praha 6
obor vzdělání Gymnázium 79-41-K/81

Maturitní práce z informatiky

# Vývoj chůze pomocí neuroevolučních algoritmů

Jan Bouček
vedoucí práce: Ing. Tomáš Obdržálek
Praha 2017

Prohlašuji, že jsem jediným autorem této maturitní práce a všechny citace, použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium Jana Keplera, Praha 6, Parléřova 2, oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 24. 3. 2017 Jan Bouček

# Obsah

# Vývoj chůze pomocí neuroevolučních algoritmů

Jan Bouček

## 1 Anotace

**česky:**
Jeden z problémů moderní robotiky je chůze, čehož se tato práce na rozdíl od konvenčních metod snaží dosáhnout pomocí strojového učení, konkrétně neuroevolučního algoritmu s nepřímým kódováním. Ten za pomoci techniky HyperNEAT generuje simulaci primitivního mozku, který chůzi robota ovládá.
**english:**
One of the challenges in modern robotics is robotic walking, which in this paper, unlike conventional methods, is achieved by machine learning, specifically using a neuroevolutionary algorithm with indirect encoding. The algorithm uses the HyperNEAT technique to generate a simulation of a primitive brain, which controls the gait of the robot.

## 2 Úvod

Jeden z cílů dnešní robotiky je konstruovat roboty, kteří jsou všestranně využitelní díky své mobilitě. Toho lze dosáhnout různými způsoby, pro svou jednoduchost se často používá jízda, která však neumožňuje spolehlivé zdolávání členitého terénu, nebo pohyb přímo ve strukturách stavěných člověkem, zejména pak zdolávání schodů.

Tato práce se inspiruje čtyřnohými kráčivými roboty firmy Boston Dynamics[1], kteří dosahují stabilní chůze exaktními metodami.[2] Podobné chůze by mělo být možné dosáhnout i pomocí organičtějších metod. Neuroevoluční algoritmy dokážou nabídnout přirozené řešení i velmi komplexních problémů, či dokonce znovuobjevit řešení, které v přírodě už existuje, proto je možné generovat čtyřnohý pohyb i tímto způsobem.[3]

Cílem projektu pak je dosáhnout podobnou metodou stabilní, vzpřímené a repetitivní chůze vpřed na simulaci čtyřnohého robota.

## 3 Teorie

Chůzi lze definovat jakožto sérii stavů kráčivého tělesa, kde každý stav lze zjednodušit jako soubor informací popisujících jednotlivé pohyblivé prvky robota. V rámci tohoto projektu budeme pracovat s dvojrozměrnou simulací čtyřnohého robota, jehož každá noha je ovládána dvěma klouby. Stav robota tak lze vystihnout jako osmirozměrný vektor $\vec{s}$. Hledáme pak ovladač robota, který pro každý stav malezne osmirozměrný vektor $\vec{\omega}$, ten každému kloubu určuje úhlovou rychlost.

## 3.1 Evoluční algoritmy

Evoluční algoritmy jsou metodou, která dokáže mnohnásobnou selekcí, mutací a kombinací různých řešení postupně vyvíjet postupně lepší řešení. K tomu potřebuje funkci, která dokáže každé řešení ohodnotit, tzv. *fitness funkci*. V našem případě hodnotíme maximální vzdálenost robota od počátku směrem do kladných hodnot $x$, a vzhledem k tomu, že cílem je chůze vzpřímená, tak u každého řešení, ve kterém některý článek robota klesne svou výškou těžiště pod experimentálně zjištěnou hodnotu $Y_{min}$, nastavíme výslednou *fitness* ovladače na *0*.

Podobně jako v přírodě, evoluční algoritmus si udržuje *populaci* řešení. V každém kroku ohodnotí výše zmíněnou funkcí všechny *živočichy*, kteří jsou pak podle své úspěšnosti rozmnoženi (pohlavně, či nepohlavně) a zmutováni. Tento proces je podrobněji popsán v následující sekci.

## 3.2 Neuroevoluce

Neuroevoluční algoritmy jsou specifickou kategorií, která se zabývá trénováním umělých neuronových sítí, anglicky *Artificial Neural Network* (ANN), které jsou abstraktním příblížením dějů v mozku, měly by být tedy schopné sloužit i jako ovladač projektovaného robota. Jejich struktura je popsána níže.

### 3.2.1 Neuronové sítě

Základní stavební jednotkou neuronové sítě je *neuron*, ten je výpočetní jednotkou, která přijímá data ve formě spojů (z biologického hlediska *dendrity*) s dalšími neurony. Pro každý z nich má nastavenou *váhu* (anglicky *weight*), která slouží jednoduše jako koeficient, který ovlivňuje „důležitost“ konkrétního spoje.

Výstupem, který pak mohou použít další neurony, je výsledek *aktivační funkce* skalárního součinu hodnot napojených neuronů a příslušných vah[4]:

$$y_k = \phi(\sum_{j=0}^{m} w_{kj} \cdot x_j),$$

kde $w$ je váha hodnoty neuronu $x$ a $\phi$ aktivační funkce.

Neurony pak lze tímto způsobem zapojovat do komplexnich sítí, kde hodnotu některých neuronů určujeme manuálně – považujeme je za *vstupy* sítě a u některých neuronů zjišťujeme hodnoty, které fungují jako *výstupy* sítě.

### 3.2.2 NEAT

*Neuroevolution of augmenting topologies*[5] (NEAT) je metodou, která definuje jeden ze způsobů, kterými lze vyvíjet neuronové sítě pomocí evolučního algoritmu. Revoluční ideou tohoto systému je způsob, kterým zapisuje neuronové sítě, jakožto *živočichy*.

Tento systém každou neuronovou síť popisuje jako ohodnocený graf pomocí dvou seznamů *genů* – seznamu vrcholů a seznamu hran. Každému vrcholu a hraně přisuzuje číselné identifikátory, pomocí kterých je dokáže přehledně propojovat a udržovat přehled o tom, kteří jedinci jsou si „geneticky“ podobní.

Stanley a Miikkulainen díky tomu zavádí i proces *speciace*, který ještě před rozmnožováním roztřídí jedince na různé *druhy* podle příbuznosti tak, aby se spolu křížily jen sítě s menšími rozdíly. Každý druh je pak ohodnocen svojí průměrnou hodnotou fitness, pomocí které se určí počet potomků v další generaci daného druhu.

Při procesu rozmnožování je z každého druhu vybrána *silnější* část, ze které se vytvoří požadovaný počet potomků, z nichž každý může vzniknout křížením – většinu genů zdědí po silnějším rodiči, ale část od slabšího, nebo bez křížení, kdy se jedinec pouze zkopíruje do další generace.

Pak na všech potomcích proběhnou mutace, přičemž je náhodně rozhodnuto, které z druhů mutací na nich proběhnou, Stanley a Miikkulainen jich popisují hned několik:

- přidání nové hrany do sítě
- rozdělení hrany na dvě hrany s novým neuronem uprostřed
- změna všech vah o malou hodnotu, nebo na náhodnou hodnotu

V tomto projektu je použito ještě několik dalších mutačních operátorů:

- aktivace/deaktivace jedné hrany
- změna jedné váhy na náhodnou hodnotu
- změna jedné váhy o maximálně $\pm 5\%$

Díky této malé úpravě dokáže algoritmus ladit neuronovou síť v mírně detailnějším měřítku.

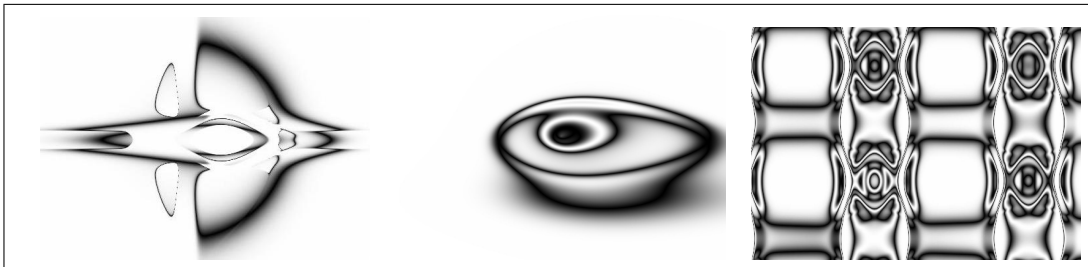
Po zmutování jsou všichni potomci prohlášeni za současnou generaci a algoritmus pokračuje znovu hodnocením jedinců.

### 3.3 CPPN-NEAT

Tím, že v NEAT algoritmu umožníme každému neuronu využívat jinou aktivační funkci, můžeme dosáhnout tvorby sítí, které jsou dobře uzpůsobené ke generování fyzické geometrie živočichů. Metoda CPPN-NEAT[6] využívá různých vlastností funkcí – symetrie, repetice apod. tak, že průchodem celou sítí jsou výsledná data nakombinována pomocí komplexní složené funkce, která si uchovává všechny tyto vlastnosti.

Pokud například vytvoříme CPPN síť s dvěma vstupy – souřadnicemi $x$ a $y$ s jedním výstupem, získáme dvojrozměrné útvary (viz obr. 1), které mají velmi blízko k fyzickému rozložení v reálných živočiších, např. dokáže nakombinovat repetici s variací a generovat útvary podobné prstům ruky nebo pomocí symetrie a variace útvar podobný lidskému oku (viz obr. 1b)

V samotném algoritmu stačí jen při tvoření nových neuronů určit náhodnou aktivační funkci a přidat mutační operátor, který změní funkci u náhodného neuronu.



(a) symetrie (b) nedokonalá symetrie (c) repetice s variací

Obrázek 1: tvary vygenerované pomocí CPPN-NEAT, převzato[7]

### 3.4 HyperNEAT

Technika HyperNEAT[7] využívá CPPN sítě ke generování čtyřrozměrného prostoru, který ve výsledku slouží jako definice vah v další neuronové síti.

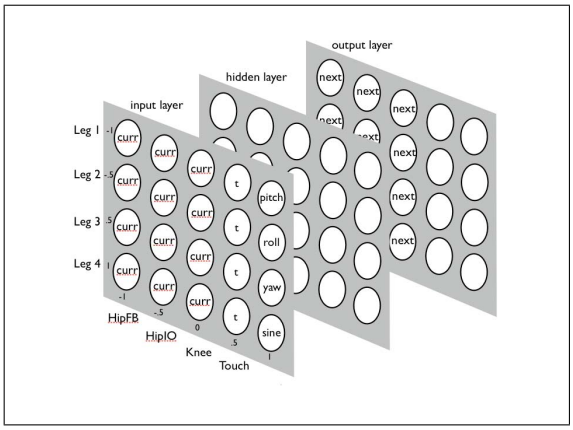
To znamená, že generujeme CPPN síť, která má čtyři vstupy – $x_1$, $y_1$, $x_2$, $y_2$. Výstup nám pak určuje váhu spoje z neuronu na *fyzických* souřadnicích $[x_1; y_1]$ do neuronu na souřadnicích $[x_2; y_2]$. Stačí nám pak navrhnout *fyzickou* strukturu sítě k využití této techniky.

V tomto projektu je použito rozložení zvané *„state-space sandwich"*[7] o třech vrstvách podobně jako [3], kde výsledná síť je rozvrstvená do trojrozměrného prostoru a z každého neuronu vede spoj do každého neuronu v další vrstvě. CPPN síť má pak dva výstupy, kde první určuje váhu mezi souřadnicemi $[x_1; y_1]$ z první vrstvy do $[x_2; y_2]$ v druhé a druhý výstup určuje stejným způsobem váhy mezi druhou a třetí vrstvou.

### 3.5 Rozložení sítě

Na rozdíl od klasických neuronových sítí jsou si sítě generované HyperNEATem *vědomé* geometrických souvislostí[7], proto je naprosto zásadní rozmístění vstupů a výstupů ve výsledné síti.

[3] umisťuje do každého řádku v *substrátu* jednu nohu a do posledního sloupce přidává neuspořádaně zbývající informace (viz obr. 2).



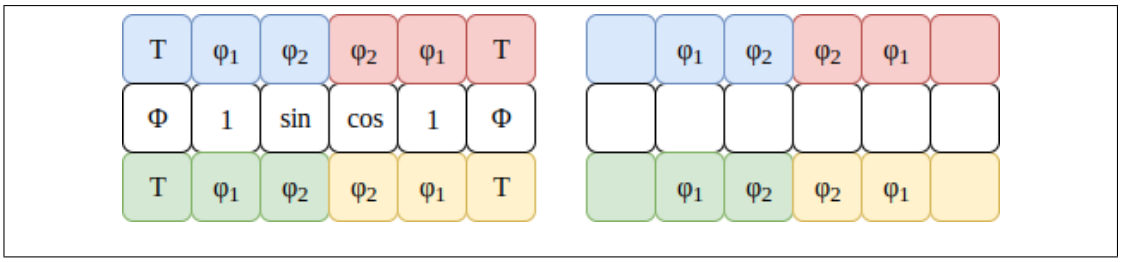

Obrázek 2: rozložení sítě v [3], převzato [3]

Tento postup sice dává geometricky najevo související buňky – podobné informace jsou ve stejných sloupcích, ale opomíjí rozložení nohou v prostoru, proto se v tomto projektu používá struktura, která za pomoci symetrie dává najevo souvislosti typu *přední/zadní pár nohou* a *levá/pravá noha*. Pro nejlepší výsledky je v rozložení nohou středová souměrnost a zbývající informace jsou v prostřední řadě mezi nohama (viz obr. 3). Každá noha je popsána pomocí dvou úhlů kloubů (označeno $\phi$) a dotykového *senzoru* (vstup označený T) na spodním článku nohy, jehož hodnota je nastavena na $-1$ bez dotyku a 1 s dotykem, pro plynulost pohybu mají tyto hodnoty lineární přechod o délce 25 snímků.

Jako přídavné proměnné jsou v tomto rozložení $sin(t/\tau)$, $cos(t/\tau)$, kde $\tau$ je experimentálně určená konstanta 40, dále úhel těla robota se zemí $\Phi$ a 1 jakožto lineární konstanta.

Z výstupní vrstvy jsou využívány pouze neurony v pozicích úhlů nohou, které jsou převedeny na úhlovou rychlost podobně jako v [3]:

$$\omega = 4,5 \cdot (\phi_{new} - \phi_{cur})$$

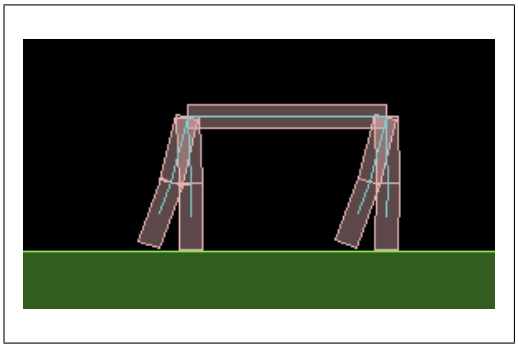



Obrázek 3: symetrické rozložení sítě, barvy označují jednotlivé nohy, vlevo vstupní vrstva, vpravo výstupní vrstva, prostřední skrytá vrstva nevyobrazena

### 3.6 simulace

K simulaci je použita knihovna JBox2D[8]. Robot se skládá z hlavního těla, na které jsou napojeny nohy (viz obr. 4) . Každá z nich je napojena pomocí rotujícího *RevoluteJointu* a stejným způsobem je napojen i spodní segment v koleni.

Měření vzdálenosti probíhá 3000 snímků při 60 snímcích za sekundu a dalších 1500 snímků se čeká, jestli robot po konci měření upadne na zem. Tím se zabraňuje strategiím, kdy místo snahy o chůzi robot pouze skočí směrem kupředu.



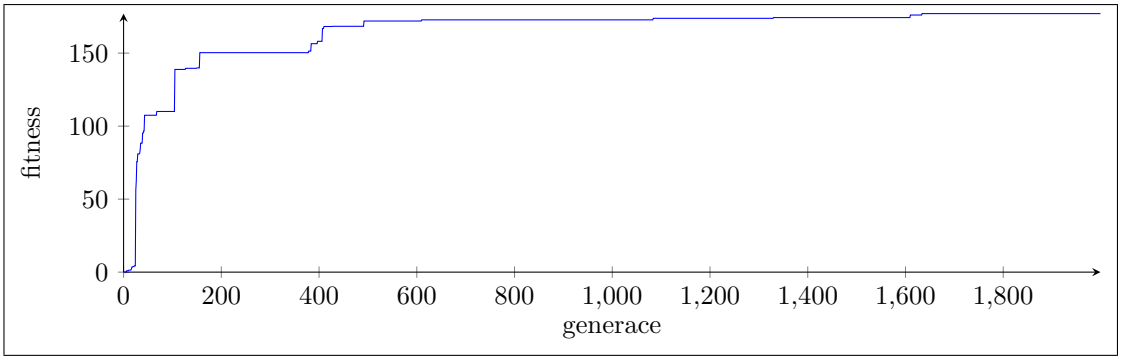
Obrázek 4: vizualizace simulace robota

## 4 Výsledky

Nejlepším výsledkem programu je ovladač robota, který je schopen během 3000 snímků ujít 177 délkových jednotek. Je schopen dosáhnout pravidelné chůze, která vypadá stabilně a přirozeně.[9]

K lidské, či zvířecí chůzi mu ale chybí jedna zásadní vlastnost – při chůzi nezvedá nohy. K tomu dochází zejména z toho důvodu, že při chůzi na absolutně hladkém povrchu k tomu ani není důvod a stálý kontakt se zemí mu dodává další stabilitu.

Jako aktivační funkce se ve výsledné neuronové síti používá sigmoida s posunutým nulovým bodem z 0,5 na 0, čímž dosahuje hodnot od -1 do 1 a je podobná často používané hyperbolické tangentě. Algoritmus zkonvergoval po přibližně 2000 generacích (viz obr. 5), což trvalo řádově 10 hodin na běžném laptopu (2 jádra, 2,4 GHz)




Obrázek 5: graf vývoje chůze

# 5 Uživatelská příručka

Výsledkem projektu jsou dva spustitelné programy – jeden, který provádí samotnou evoluci a druhý na zobrazovnání výsledků. Oba programy jsou napsány v jazyce Java a lze je zkompilovat pomocí prostředí IntelliJ IDEA, ve kterém byl projekt vyvíjen. Ke kompilaci je potřeba JDK s balíkem JavaFX (distribuce typu OpenJDK apod. nelze použít). Projekt byl vždy kompilován a spouštěn na systému Linux (Manjaro Linux s prostředím i3wm), případná kompatibilita s jinými systémy je pravděpodobná, avšak nezaručená.

## 5.1 Zobrazovač výsledků

Jedním z programů je aplikace na zobrazování simulací výsledných sítí (viz obr. 6). Její grafické prostředí je ovládané poměrně intuitivně, v levé části je na výběr postupně:

- běh programu (podle názvu složky)
- generace programu
- jedinec z generace

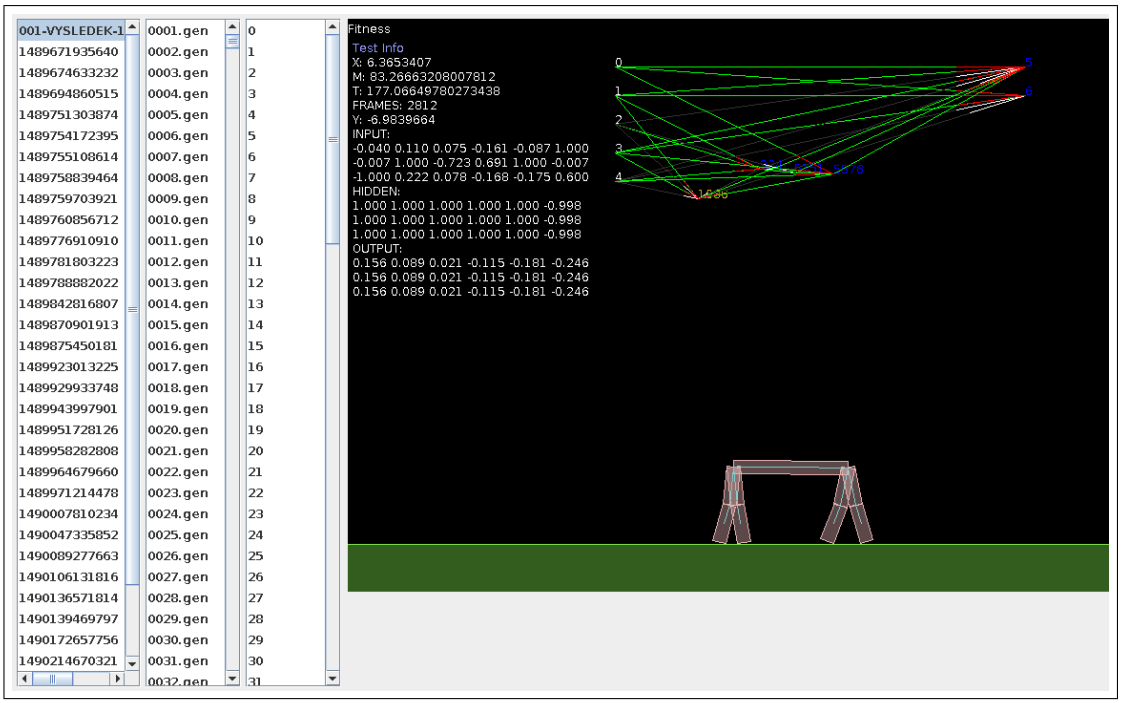
Na pravé straně se zobrazuje simulace společně s užitečnými daty umístěnými v levé části simulačního panelu:

- souřadnice $x$ těla robota
- maximální dosažená souřadnice $x$ těla robota v současné simulaci
- maximální dosažená souřadnice $x$ těla robota po 3000 snímků
- počet uběhlých snímků
- souřadnice $y$ těla robota
- současné hodnoty vstupní vrstvy neuronové sítě
- současné hodnoty skryté vrstvy neuronové sítě
- současné hodnoty výstupní vrstvy neuronové sítě

Napravo od užitečných dat je vyobrazena CPPN síť. Vstupní neurony jsou na jejím levém okraji, výstupní neurony na pravém okraji, mezi nimi jsou náhodně rozloženy skryté neurony, jejichž barva značí použitou aktivační funkci. Spoje v síti mají svůj cílový neuron označený kratším obarveným koncem. Neaktivní spoje jsou vyznačeny šedou barvou.

## 5.2 Evoluční program

Evoluční program je nutno spustit z konzole a vyžaduje soubor *config.cfg*, který obsahuje nastavení několika klíčových konstant, které lze měnit. Program ukládá výsledky do složky *results*, kam také ukládá záznam průběhu programu, který rovněž vypisuje do konzole. Vypisuje několik důležitých informací, jako nejdelší dosaženou vzdálenost, počet potomků, popis jednotlivých druhů apod.




Obrázek 6: screenshot zobrazovače výsledků

# 6 Technologie

Celý program je napsaný v jazyce Java, který byl zvolen zejména pro svou propracovanou objektovou strukturu, která je ideální pro větší projekty.

Celý program je pak rozstrukturovaný do několika balíků podle využití:

- *launcher*: balík pouze pro spouštěcí třídu evolučního algoritmu
- *neat*: balík obsahující celou evoluční část algoritmu, zejména třídu Evolution, která tvoří jádro algoritmu
- *simulation*: balík starající se o konstrukci sítě z genetického zápisu, následné simulace a měření fitness
- *worlbuilding*: pomocný balík využívaný pro přípravu JBox2D světa před simulací a stavění těla robota
- *testsettings*: balík obsahující kontejner konstant načítaných z konfiguračního souboru
- *iohandling*: balík starající se o zápis a čtení dat
- *results_viewer*: balík zabývající se vizualizací výsledků, obsahuje spustitelnou třídu GUI

Každá třída také obsahuje krátký popis svého obsahu.

Pro fyzikální simulace a jejich zobrazení je použita knihovna JBox2D[8], která byla zvolena pro svou jednoduchost užití a flexibilitu.

# 7 Závěr

Výsledky ukázaly, že pro robotickou chůzi je neuroevoluční algoritmus srovnatelný s konvenčními metodami. Program by měl být schopný generovat podobné

výsledky i pro mírně odlišné konfigurace robota jen s minimálními úpravami. I s výkonem běžně dostupného hardwaru se podařilo dosáhnout chůze podobné lidské či zvířecí.

Model však byl velmi přibližný a pro reálné využití tohoto postupu by bylo nutné simulovat v trojrozměrném prostoru a učit robota i chůzi terénem za pomoci dalších senzorů.

## Reference


[1] Introducing Spot. In: *Youtube* [online]. 9.2.2015 [cit. 2017-03-20]. Dostupné z: https://www.youtube.com/watch?v=M8YjvHYbZ9w. Kanál uživatele Boston Dynamics

[2] RAIBERT, Marc, Kevin BLANKESPOOR, Gabriel NELSON a Rob PLAYTER. *BigDog, the Rough-Terrain Quadruped Robot* [online]. [cit. 2017-03-20]. DOI: 10.3182/20080706-5-KR-1001.01833. ISBN 10.3182/20080706-5-KR-1001.01833. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1474667016407020

[3] CLUNE, Jeff, Benjamin E. BECKMANN, Charles OFRIA a Robert T. PENNOCK. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In: *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, s. 2764-2771. DOI: 10.1109/CEC.2009.4983289. ISBN 978-1-4244-2958-5. Dostupné také z: http://ieeexplore.ieee.org/document/4983289/

[4] Artificial neuron. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2017 [cit. 2017-03-21]. Dostupné z: https://en.wikipedia.org/wiki/Artificial_neuron

[5] STANLEY, Kenneth O. a Risto MIIKKULAINEN. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* [online]. 2002, **10**(2), 99-127 [cit. 2017-03-21]. DOI: 10.1162/106365602320169811. ISSN 1063-6560. Dostupné z: http://www.mitpressjournals.org/doi/10.1162/106365602320169811

[6] STANLEY, Kenneth O. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines* [online]. 2007-6-6, **8**(2), 131-162 [cit. 2017-03-21]. DOI: 10.1007/s10710-007-9028-8. ISSN 1389-2576. Dostupné z: http://link.springer.com/10.1007/s10710-007-9028-8

[7] STANLEY, Kenneth O., David B. D'AMBROSIO a Jason GAUCI. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life* [online]. 2009, **15**(2), 185-212 [cit. 2017-03-21]. DOI: 10.1162/artl.2009.15.2.15202. ISSN 1064-5462. Dostupné z: http://www.mitpressjournals.org/doi/10.1162/artl.2009.15.2.15202

[8] *JBox2D* [online]. Daniel Murphy, 2014 [cit. 2017-03-22]. Dostupné z: http://www.jbox2d.org/

[9] Crawler: HyperNEAT walking robot. In: *Youtube* [online]. 24.3.2015 [cit. 2017-03-24]. Dostupné z: https://www.youtube.com/watch?v=RKZqn1YrZOo. Kanál uživatele Jan Bouček