■ AI topic 중 "pose estimation" 관련 논문 scraping 및 hot topic 분석

</span>

Google Scholar 를 통한 논문 검색

- AI 는 최신 기술, 최신 연구 내용이 중요하고 계속 새로운 SOTA(state-of-the-art) 알고리즘이 나오고 있기 때문에 follow-up 해가며 update 해주는 것이 중요하다.
- 중복 연구를 피하면서 현재 연구에 도움이 되는 유의미한 논문을 찾아야 한다.
- 따라서 많은 수로 인용된 논문들을 선별하여 우수 논문을 찾고, 우수 논문들이 다루는 topic 에 대해 알아본다.
- 연도별로 논문을 분류해 pose estimation 관련하여 어떤 연도에 가장 연구가 활발히 이루어졌는지 알아본다.

- 영어 논문이기 때문에 nltk 의 Penn Treebank Tagset 을 사용하여 POS Tagging 을 진행한다.
- 논문은 네이버, 다음, 구글보다 google scholar 를 통해 더 질 좋은 검색을 할 수 있다.

In [2]:

```
## 기본
import numpy as np                     # numpy 패키지 가져오기
import pandas as pd                    # pandas 패키지 가져오기
import matplotlib.pyplot as plt        # 시각화 패키지 가져오기

## Text 데이터 처리
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize

from collections import Counter
from wordcloud import WordCloud

from bs4 import BeautifulSoup
import requests

import re

import warnings
warnings.filterwarnings('ignore')
```

(1) 논문 검색

■ 구글 scholar 검색창에서 'pose estimation' 입력

필요한 부분을 정규표현식을 사용하여 추출

In [3]:

```
url ='https://scholar.google.com/scholar?hl=ko&as_sdt=0%2C5&q=pose+estimation&btnG='
req = requests.get(url)    # 해당 페이지를 가져옴
page = BeautifulSoup(req.text , 'html.parser')
```

▶ 논문 제목 추출

In [4]:

```
titles = page.find_all(attrs='gs_rt')
```

검색된 **title**의 크기를 확인한다.

In [5]:

```
print(len(titles))
```

```
10
```

검색된 **title**을 각각 프린트 한다.

In [6]:

```
for title in titles :
    print(title.text)
    print()
```

```
Pose estimation from corresponding point data

Deeppose: Human pose estimation via deep neural networks

Stacked hourglass networks for human pose estimation

Head pose estimation in computer vision: A survey

Rmpe: Regional multi-person pose estimation

Fast pose estimation with parameter-sensitive hashing

Linear pose estimation from points or lines

Articulated pose estimation with flexible mixtures-of-parts

Realtime multi-person 2d pose estimation using part affinity fields

3d human pose estimation= 2d pose estimation+ matching
```

▶ 논문 인용횟수 추출

- 정규표현식을 이용해 논문 인용횟수를 추출한다.

In [7]:

```
cite = page.select('div.gs_fl')
num = re.compile("[0-9]+")
cite_final = []

for n in cite:
    if num.search(n.get_text()) :
        cite_final.append(num.search(n.get_text()).group())

print(cite_final)
```

```
['899', '2028', '2765', '1551', '622', '1009', '511', '1342', '3605', '318']
```

▶ 논문 저자 추출

- 정규표현식을 이용해 논문 저자를 추출한다.

In [8]:

```
author = page.select('div.gs_a')

first = re.compile(r"₩w*₩s*₩w+")
author_final = []

for a in author:
    if first.search(a.get_text()) :
        author_final.append(first.search(a.get_text()).group())

print(author_final)
```

```
['RM Haralick', 'A Toshev', 'A Newell', 'E Murphy', 'HS Fang', 'G Shakhnarovich', 'A Ansa
r', 'Y Yang', 'Z Cao', 'CH Chen']
```

▶ 논문 작성연도 추출

- 정규표현식을 이용해 논문 작성연도를 추출한다.

In [9]:

```
year = page.select('div.gs_a')

yr = re.compile('₩d₩d₩d₩d')
year_final = []

for y in year:
    if yr.search(y.get_text()) :
        year_final.append(yr.search(y.get_text()).group())

print(year_final)
```

```
['1989', '2014', '2016', '2008', '2017', '2003', '2003', '2011', '2017', '2017']
```

(2) 스크래핑

▶ 첫 페이지부터 원하는 페이지 까지 스크래핑하는 함수 정의

In [10]:

```
# 해드라인 검색 및 추출 함수를 정의
# 검색 키워드 & 스타트 페이지 & 검색 마지막 페이지를 parameter(매개변수) 로 받음
# 파이썬은 매개변수가 여러개 인 경우 뒤의 매개 변수부터 디폴트값을 가질 수 있음


title_list = []
cite_list = []
author_list = []
year_list = []

def web_scraping(keyword, end, start = 0):    # 시작 페이지의 default 값은 0(즉, 첫 페이
    while 1:

        if start > (end-1) * 10 :    # 스타트 페이지가 마지막 페이지보다 크면 while 문을
            break

        url ='https://scholar.google.com/scholar?start={}&q={}'.format(start,keyword)

        req = requests.get(url)    # 해당 페이지를 가져옴

        page = BeautifulSoup(req.text, 'html.parser')

        titles = page.find_all(attrs='gs_rt')     # 헤드라인 기사를 가져옴
        for one in titles:
            title_list.append(one.text)

        cite = page.select('div.gs_fl')
        num = re.compile("[0-9]+")
        for n in cite:
            if num.search(n.get_text()) :
                cite_list.append(num.search(n.get_text()).group())

        author = page.select('div.gs_a')
        first = re.compile(r"₩w*₩s*₩w+")
        for a in author:
            if first.search(a.get_text()) :
                author_list.append(first.search(a.get_text()).group())

        year = page.select('div.gs_a')
        yr = re.compile('₩d₩d₩d₩d')
        for y in year:
            if yr.search(y.get_text()) :
                year_list.append(yr.search(y.get_text()).group())

        start += 10

    print(title_list)
    print(cite_list)
    print(author_list)
    print(year_list)
```

▶ 위 함수를 호출

In [12]:

```
# 검색어를 입력 받음
keyword = input('검색어를 입력하세요 : ')
print()

end_page = int(input('스크래핑할 마지막 페이지를 입력하세요 : '))

web_scraping(keyword, end_page)      #  end 페이지 까지 스크래핑을 해온다.
```

```
검색어를 입력하세요 : pose estimation

스크래핑할 마지막 페이지를 입력하세요 : 5
['Pose estimation from corresponding point data', 'Deeppose: Human pose estimation via de
ep neural networks', 'Stacked hourglass networks for human pose estimation', 'Head pose e
stimation in computer vision: A survey', 'Rmpe: Regional multi-person pose estimation',
'Fast pose estimation with parameter-sensitive hashing', 'Linear pose estimation from poi
nts or lines', 'Articulated pose estimation with flexible mixtures-of-parts', 'Realtime m
ulti-person 2d pose estimation using part affinity fields', '3d human pose estimation= 2d
pose estimation+ matching', '2d human pose estimation: New benchmark and state of the art
analysis', 'Vision-based hand pose estimation: A review', 'A simple yet effective baselin
e for 3d human pose estimation', 'Multiposenet: Fast multi-person pose estimation using p
ose residual network', 'Robust pose estimation from a planar target', 'Deepercut: A deepe
r, stronger, and faster multi-person pose estimation model', 'Densepose: Dense human pose
estimation in the wild', 'Algorithms for plane-based pose estimation', 'Simple baselines
for human pose estimation and tracking', 'Recognizing human actions as the evolution of p
ose estimation maps', 'Face detection, pose estimation, and landmark localization in the
wild', '[PDF][PDF] Clustered Pose and Nonlinear Appearance Models for Human Pose Estimati
on.', 'Towards accurate multi-person pose estimation in the wild', 'Deep high-resolution
representation learning for human pose estimation', 'Recurrent human pose estimation', 'P
ifpaf: Composite fields for human pose estimation', 'Progressive search space reduction f
or human pose estimation', 'Posetrack: A benchmark for human pose estimation and trackin
g', 'Fast human pose estimation', '[PDF][PDF] Analysis and solutions of the three point p
erspective pose estimation problem.', 'Review and analysis of solutions of the three poin
t perspective pose estimation problem', 'Human pose estimation with iterative error feedb
ack', 'Flowing convnets for human pose estimation in videos', 'Monocular 3d pose estimati
on and tracking by detection', 'OpenPose: realtime multi-person 2D pose estimation using
Part Affinity Fields', 'Fast and globally convergent pose estimation from video images',
'Fast animal pose estimation using deep neural networks', 'Pose machines: Articulated pos
e estimation via inference machines', 'Learning feature pyramids for human pose estimatio
n', 'Efficient human pose estimation from single depth images', 'Real time hand pose esti
mation using depth sensors', 'Detect-and-track: Efficient pose estimation in videos', 'Le
arning to refine human pose estimation', 'Posetrack: Joint multi-person pose estimation a
nd tracking', 'Pose estimation for augmented reality: a hands-on survey', 'Learning pose
grammar to encode human body configuration for 3d pose estimation', 'Human pose estimatio
n via convolutional part heatmap regression', 'Structured feature learning for pose estim
ation', 'Worldwide pose estimation using 3d point clouds', "Does human action recognition
benefit from pose estimation?''"]
['899', '2028', '2765', '1551', '622', '1009', '511', '1342', '3605', '318', '1424', '99
3', '581', '120', '414', '754', '576', '175', '562', '124', '2507', '673', '462', '774',
'279', '109', '845', '176', '57', '469', '766', '560', '468', '583', '1313', '997', '17
5', '263', '304', '598', '419', '142', '59', '150', '321', '155', '438', '222', '391', '2
12']
['RM Haralick', 'A Toshev', 'A Newell', 'E Murphy', 'HS Fang', 'G Shakhnarovich', 'A Ansa
r', 'Y Yang', 'Z Cao', 'CH Chen', 'M Andriluka', 'A Erol', 'J Martinez', 'M Kocabas', 'G
Schweighofer', 'E Insafutdinov', 'RA Güler', 'P Sturm', 'B Xiao', 'M Liu', 'X Zhu', 'S Jo
hnson', 'G Papandreou', 'K Sun', 'V Belagiannis', 'S Kreiss', 'V Ferrari', 'M Andriluka',
'F Zhang', 'RM Haralick', 'BM Haralick', 'J Carreira', 'T Pfister', 'M Andriluka', 'Z Ca
o', 'CP Lu', 'TD Pereira', 'V Ramakrishna', 'W Yang', 'J Shotton', 'C Keskin', 'R Girdha
r', 'M Fieraru', 'U Iqbal', 'E Marchand', 'HS Fang', 'A Bulat', 'X Chu', 'Y Li', 'A Yao']
['1989', '2014', '2016', '2008', '2017', '2003', '2003', '2011', '2017', '2017', '2014',
'2007', '2017', '2018', '2006', '2016', '2018', '2000', '2018', '2018', '2012', '2010',
'2017', '2019', '2017', '2019', '2008', '2018', '2019', '1991', '1994', '2016', '2015',
'2010', '2019', '2000', '2019', '2014', '2017', '2012', '2013', '2018', '2018', '2017',
'2015', '2018', '2016', '2016', '2012', '2011']
```

문자열인 인용횟수를 정수형 리스트로 변환

In [13]:

```
cite_list2 = list(map(int, cite_list))
```

수집해 온 갯수 확인

In [14]:

```
print(len(title_list))
print(len(cite_list2))
print(len(author_list))
print(len(year_list))
```

```
50
50
50
50
```

(3) 논문을 dataframe 으로 정리하기

▶ **author, title, year, cite** 를 **column** 으로 **dataframe** 을 만들어 **'pose estimation'** 관련 논문을 보기 좋게 표로 정리한다.

▶ **rank** 를 이용해 인용횟수가 많은 순위대로 정렬한다.

In [15]:

```
data = pd.DataFrame(zip(author_list, title_list, year_list, cite_list2), columns = ['auth
data['rank'] = data['cite'].rank(method='min', ascending=False)
data_sorted = data.sort_values('rank')
```

In [16]:

```
data_sorted
```

Out[16]:

	author	title	year	cite	rank
8	Z Cao	Realtime multi-person 2d pose estimation using...	2017	3605	1.0
2	A Newell	Stacked hourglass networks for human pose esti...	2016	2765	2.0
20	X Zhu	Face detection, pose estimation, and landmark ...	2012	2507	3.0
1	A Toshev	Deeppose: Human pose estimation via deep neura...	2014	2028	4.0
3	E Murphy	Head pose estimation in computer vision: A survey	2008	1551	5.0
10	M Andriluka	2d human pose estimation: New benchmark and st...	2014	1424	6.0
7	Y Yang	Articulated pose estimation with flexible mixt...	2011	1342	7.0
34	Z Cao	OpenPose: realtime multi-person 2D pose estima...	2019	1313	8.0
5	G Shakhnarovich	Fast pose estimation with parameter-sensitive ...	2003	1009	9.0
35	CP Lu	Fast and globally convergent pose estimation f...	2000	997	10.0
11	A Erol	Vision-based hand pose estimation: A review	2007	993	11.0
0	RM Haralick	Pose estimation from corresponding point data	1989	899	12.0
26	V Ferrari	Progressive search space reduction for human p...	2008	845	13.0
23	K Sun	Deep high-resolution representation learning f...	2019	774	14.0
30	BM Haralick	Review and analysis of solutions of the three ...	1994	766	15.0
15	E Insafutdinov	Deepercut: A deeper, stronger, and faster mult...	2016	754	16.0
21	S Johnson	[PDF][PDF] Clustered Pose and Nonlinear Appear...	2010	673	17.0
4	HS Fang	Rmpe: Regional multi-person pose estimation	2017	622	18.0
39	J Shotton	Efficient human pose estimation from single de...	2012	598	19.0
33	M Andriluka	Monocular 3d pose estimation and tracking by d...	2010	583	20.0
12	J Martinez	A simple yet effective baseline for 3d human p...	2017	581	21.0
16	RA Güler	Densepose: Dense human pose estimation in the ...	2018	576	22.0
18	B Xiao	Simple baselines for human pose estimation and...	2018	562	23.0
31	J Carreira	Human pose estimation with iterative error fee...	2016	560	24.0
6	A Ansar	Linear pose estimation from points or lines	2003	511	25.0
29	RM Haralick	[PDF][PDF] Analysis and solutions of the three...	1991	469	26.0
32	T Pfister	Flowing convnets for human pose estimation in ...	2015	468	27.0
22	G Papandreou	Towards accurate multi-person pose estimation ...	2017	462	28.0
46	A Bulat	Human pose estimation via convolutional part h...	2016	438	29.0
40	C Keskin	Real time hand pose estimation using depth sen...	2013	419	30.0
14	G Schweighofer	Robust pose estimation from a planar target	2006	414	31.0
48	Y Li	Worldwide pose estimation using 3d point clouds	2012	391	32.0

	author	title	year	cite	rank
44	E Marchand	Pose estimation for augmented reality: a hands...	2015	321	33.0
9	CH Chen	3d human pose estimation= 2d pose estimation+ ...	2017	318	34.0
38	W Yang	Learning feature pyramids for human pose estim...	2017	304	35.0
24	V Belagiannis	Recurrent human pose estimation	2017	279	36.0
37	V Ramakrishna	Pose machines: Articulated pose estimation via...	2014	263	37.0
47	X Chu	Structured feature learning for pose estimation	2016	222	38.0
49	A Yao	Does human action recognition benefit from pos...	2011	212	39.0
27	M Andriluka	Posetrack: A benchmark for human pose estimati...	2018	176	40.0
36	TD Pereira	Fast animal pose estimation using deep neural ...	2019	175	41.0
17	P Sturm	Algorithms for plane-based pose estimation	2000	175	41.0
45	HS Fang	Learning pose grammar to encode human body con...	2018	155	43.0
43	U Iqbal	Posetrack: Joint multi-person pose estimation ...	2017	150	44.0
41	R Girdhar	Detect-and-track: Efficient pose estimation in...	2018	142	45.0
19	M Liu	Recognizing human actions as the evolution of ...	2018	124	46.0
13	M Kocabas	Multiposenet: Fast multi-person pose estimatio...	2018	120	47.0
25	S Kreiss	Pifpaf: Composite fields for human pose estima...	2019	109	48.0
42	M Fieraru	Learning to refine human pose estimation	2018	59	49.0
28	F Zhang	Fast human pose estimation	2019	57	50.0

(4) 데이터 분석 결과 wordcloud 및 그래프 생성

[1] 인용횟수가 많은 논문의 keyword 분석 - wordcloud 생성

In [17]:

```python
title_cite = pd.DataFrame(zip(title_list, cite_list2), columns = ['title', 'cite'])
```

pose estimation 검색 단어 자체와 **'PDF'** 라는 불용어는 자체 **stop words** 로 설정해 **keyword**에서 제외

In [18]:

```python
stopwords = ["PDF", "pose", "estimation", "Pose", "Estimation"]

for i in range(0, 50):
    word_tag = pos_tag(word_tokenize(title_cite['title'][i]))
    words = []
    for word, tag in word_tag:
        if tag in ['NN', 'NNS', 'NNP', 'NNPS'] and word not in stopwords:
            words.append(word)
            title_cite['title'][i] = words
        else :
            title_cite['title'][i] = "none"

# 리스트 형 문자열로 변환
```

```
for i in range(0,50) :
    t = title_cite['title'][i]
    if (t == 'none') :
        title_cite['title'][i] = 'none'
    else :
        strt = ", ".join(t)
        title_cite['title'][i] = strt
```

wordcloud 생성을 위한 기준 **dictionary** 생성 (해당 논문의 **keyword** : 인용횟수)

In [19]:

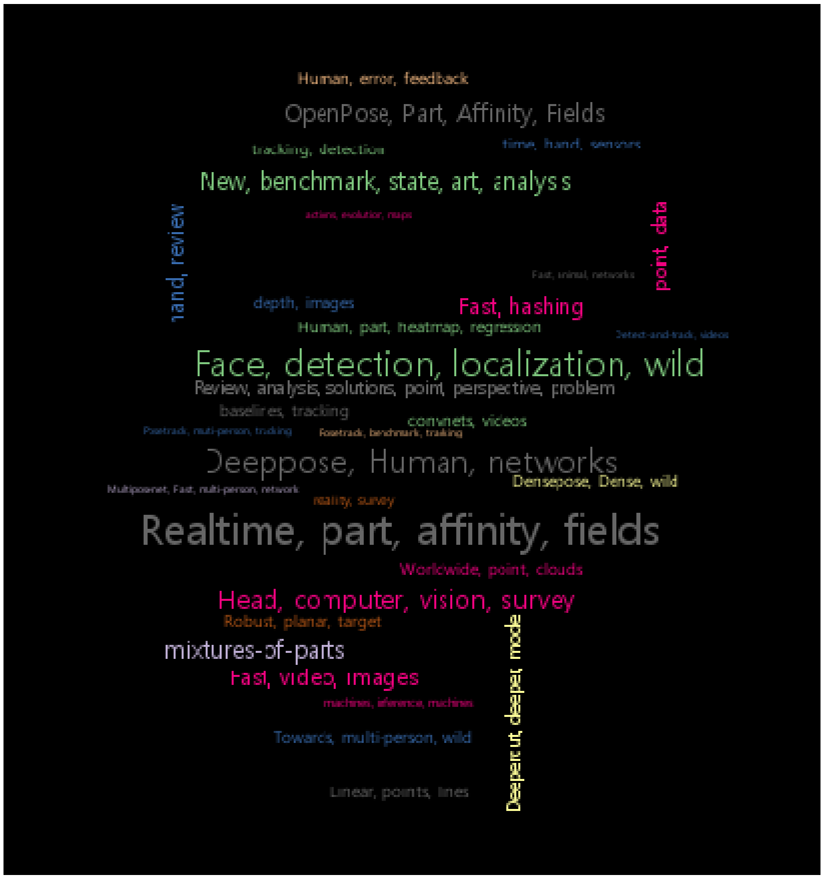
```
dict1 = []
dict2 = []
for i in range(0,50) :
    t = title_cite['title'][i]
    c = title_cite['cite'][i]
    if (t != 'none') :
        dict1.append(t)
        dict2.append(c)

cloud_list = dict(zip(dict1, dict2))
```

▶ 결과 [인용횟수가 많은 논문의 최빈 keyword 값]

: DeepPose, Realtime, affinity, face detection, localization, networks

In [20]:

```
cloud_img = plt.imread('ai2.jpg')       # word cloud 이미지는 인공지능 이미지 사용
wordcloud2 = WordCloud(font_path='C:/Windows/Fonts/malgun.ttf',
                      colormap = 'Accent_r',
                      width=800,
                      height=600,
                      mask = cloud_img)
cloud = wordcloud2.generate_from_frequencies(cloud_list)
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(cloud)
plt.show()
```


Human, error, feedback
OpenPose, Part, Affinity, Fields
tracking, detection
time, hand, sensors
New, benchmark, state, art, analysis
hand, review
actions, evolution, maps
point, data
Fast, animal, networks
depth, images
Fast, hashing
Human, part, heatmap, regression
Detect-and-track, videos
Face, detection, localization, wild
Review, analysis, solutions, point, perspective, problem
baselines, tracking
convnets, videos
Posetrack, multi-person, tracking
Posetrack, benchmark, tracking
Deeppose, Human, networks
Densepose, Dense, wild
Multiposenet, Fast, multi-person, network
reality, survey
Realtime, part, affinity, fields
Worldwide, point, clouds
Head, computer, vision, survey
Robust, planar, target
mixtures-of-parts
Fast, video, images
machines, inference, machines
Towards, multi-person, wild
Linear, points, lines
Deepercut, deeper, model

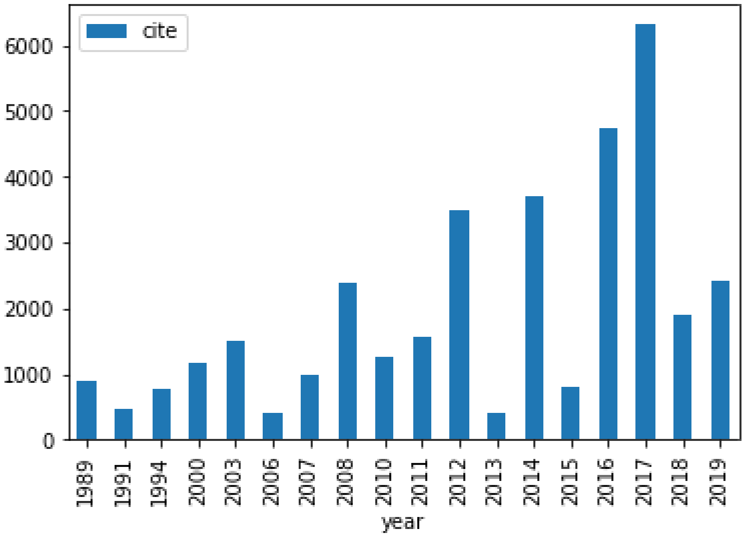
[2] 인용횟수 별 논문 작성 연도 plot

In [21]:

```
author_df = pd.DataFrame(zip(year_list, cite_list2), columns = ['year',  'cite'])
author_df2 = author_df.groupby('year').sum()
author_df2.plot.bar()
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22918627548>
```



▶ 결과 [인용횟수가 가장 많았던 연도]

: **2017**년도에 **pose estimation** 관련 가장 활발한 논문 인용이 있었다.

[3] 인용이 많이 된 상위 20개 논문을 뽑아 keyword 분석하기

In [22]:

```
top = data_sorted[:20]
top_title = []
for title in top['title'] :
    top_title.append(title)
```

형태소 분석

In [23]:

```
sentences_tag = []

for sentence in top_title:
    word_tag = pos_tag(word_tokenize(sentence))
    sentences_tag.append(word_tag)

print(sentences_tag)
```

```
[[('Realtime', 'NNP'), ('multi-person', 'JJ'), ('2d', 'CD'), ('pose', 'JJ'), ('estimatio
n', 'NN'), ('using', 'VBG'), ('part', 'NN'), ('affinity', 'NN'), ('fields', 'NNS')], [('S
tacked', 'JJ'), ('hourglass', 'NN'), ('networks', 'NNS'), ('for', 'IN'), ('human', 'JJ'),
('pose', 'JJ'), ('estimation', 'NN')], [('Face', 'NNP'), ('detection', 'NN'), (',', ','),
('pose', 'JJ'), ('estimation', 'NN'), (',', ','), ('and', 'CC'), ('landmark', 'JJS'), ('l
ocalization', 'NN'), ('in', 'IN'), ('the', 'DT'), ('wild', 'NN')], [('Deeppose', 'NN'),
(':', ':'), ('Human', 'NNP'), ('pose', 'VBP'), ('estimation', 'NN'), ('via', 'IN'), ('dee
p', 'JJ'), ('neural', 'JJ'), ('networks', 'NNS')], [('Head', 'NNP'), ('pose', 'JJ'), ('es
timation', 'NN'), ('in', 'IN'), ('computer', 'NN'), ('vision', 'NN'), (':', ':'), ('A',
'DT'), ('survey', 'NN')], [('2d', 'CD'), ('human', 'JJ'), ('pose', 'JJ'), ('estimation',
'NN'), (':', ':'), ('New', 'NNP'), ('benchmark', 'NN'), ('and', 'CC'), ('state', 'NN'),
('of', 'IN'), ('the', 'DT'), ('art', 'NN'), ('analysis', 'NN')], [('Articulated', 'VBN'),
('pose', 'JJ'), ('estimation', 'NN'), ('with', 'IN'), ('flexible', 'JJ'), ('mixtures-of-p
arts', 'NNS')], [('OpenPose', 'NN'), (':', ':'), ('realtime', 'JJ'), ('multi-person', 'J
J'), ('2D', 'CD'), ('pose', 'JJ'), ('estimation', 'NN'), ('using', 'VBG'), ('Part', 'NN
P'), ('Affinity', 'NNP'), ('Fields', 'NNP')], [('Fast', 'NNP'), ('pose', 'JJ'), ('estimat
ion', 'NN'), ('with', 'IN'), ('parameter-sensitive', 'JJ'), ('hashing', 'NN')], [('Fast',
'NNP'), ('and', 'CC'), ('globally', 'RB'), ('convergent', 'JJ'), ('pose', 'JJ'), ('estima
tion', 'NN'), ('from', 'IN'), ('video', 'NN'), ('images', 'NNS')], [('Vision-based', 'J
J'), ('hand', 'NN'), ('pose', 'JJ'), ('estimation', 'NN'), (':', ':'), ('A', 'DT'), ('rev
iew', 'NN')], [('Pose', 'JJ'), ('estimation', 'NN'), ('from', 'IN'), ('corresponding', 'V
BG'), ('point', 'NN'), ('data', 'NNS')], [('Progressive', 'NNP'), ('search', 'NN'), ('spa
ce', 'NN'), ('reduction', 'NN'), ('for', 'IN'), ('human', 'JJ'), ('pose', 'JJ'), ('estima
tion', 'NN')], [('Deep', 'JJ'), ('high-resolution', 'NN'), ('representation', 'NN'), ('le
arning', 'VBG'), ('for', 'IN'), ('human', 'JJ'), ('pose', 'JJ'), ('estimation', 'NN')],
[('Review', 'NNP'), ('and', 'CC'), ('analysis', 'NN'), ('of', 'IN'), ('solutions', 'NN
S'), ('of', 'IN'), ('the', 'DT'), ('three', 'CD'), ('point', 'NN'), ('perspective', 'N
N'), ('pose', 'JJ'), ('estimation', 'NN'), ('problem', 'NN')], [('Deepercut', 'NN'),
(':', ':'), ('A', 'DT'), ('deeper', 'NN'), (',', ','), ('stronger', 'JJR'), (',', ','),
('and', 'CC'), ('faster', 'RBR'), ('multi-person', 'JJ'), ('pose', 'JJ'), ('estimation',
'NN'), ('model', 'NN')], [('[', 'JJ'), ('PDF', 'NNP'), (']', 'NNP'), ('[', 'NNP'), ('PD
F', 'NNP'), (']', 'NNP'), ('Clustered', 'NNP'), ('Pose', 'NNP'), ('and', 'CC'), ('Nonline
ar', 'NNP'), ('Appearance', 'NNP'), ('Models', 'NNP'), ('for', 'IN'), ('Human', 'NNP'),
('Pose', 'NNP'), ('Estimation', 'NNP'), ('.', '.')], [('Rmpe', 'NN'), (':', ':'), ('Regio
nal', 'JJ'), ('multi-person', 'NN'), ('pose', 'NN'), ('estimation', 'NN')], [('Efficien
t', 'JJ'), ('human', 'JJ'), ('pose', 'JJ'), ('estimation', 'NN'), ('from', 'IN'), ('singl
e', 'JJ'), ('depth', 'NN'), ('images', 'NNS')], [('Monocular', 'JJ'), ('3d', 'CD'), ('pos
e', 'JJ'), ('estimation', 'NN'), ('and', 'CC'), ('tracking', 'NN'), ('by', 'IN'), ('detec
tion', 'NN')]]
```

명사 추출

In [24]:

```
# 형태소 분석 후 명사만 추출

noun_list = []

stopwords = ["PDF", "pose", "estimation", "Pose", "Estimation"]     # pose estimation 자

for sentence in sentences_tag:
    for word, tag in sentence:
        if tag in ['NN', 'NNS', 'NNP', 'NNPS'] and word not in stopwords:           #
            noun_list.append(word)

print(noun_list)
```

```
['Realtime', 'part', 'affinity', 'fields', 'hourglass', 'networks', 'Face', 'detection',
'localization', 'wild', 'Deeppose', 'Human', 'networks', 'Head', 'computer', 'vision', 's
urvey', 'New', 'benchmark', 'state', 'art', 'analysis', 'mixtures-of-parts', 'OpenPose',
'Part', 'Affinity', 'Fields', 'Fast', 'hashing', 'Fast', 'video', 'images', 'hand', 'revi
ew', 'point', 'data', 'Progressive', 'search', 'space', 'reduction', 'high-resolution',
'representation', 'Review', 'analysis', 'solutions', 'point', 'perspective', 'problem',
'Deepercut', 'deeper', 'model', ']', '[', ']', 'Clustered', 'Nonlinear', 'Appearance', 'M
odels', 'Human', 'Rmpe', 'multi-person', 'depth', 'images', 'tracking', 'detection']
```

두음절 이상 단어만 추출

In [25]:

```
print('▶ 전체 명사의 수 = ', len(noun_list))
print()
noun_list = [word for word in noun_list if len(word) > 1]    # 명사중에서 두음절 이상의

print('▶ 두음절 이상의 명사의 수 = ', len(noun_list))
print()
print(noun_list[:100])   # 처음부터 나오는 순서대로 100개 단어 출력
```

```
▶ 전체 명사의 수 =  65

▶ 두음절 이상의 명사의 수 =  62

['Realtime', 'part', 'affinity', 'fields', 'hourglass', 'networks', 'Face', 'detection',
'localization', 'wild', 'Deeppose', 'Human', 'networks', 'Head', 'computer', 'vision', 's
urvey', 'New', 'benchmark', 'state', 'art', 'analysis', 'mixtures-of-parts', 'OpenPose',
'Part', 'Affinity', 'Fields', 'Fast', 'hashing', 'Fast', 'video', 'images', 'hand', 'revi
ew', 'point', 'data', 'Progressive', 'search', 'space', 'reduction', 'high-resolution',
'representation', 'Review', 'analysis', 'solutions', 'point', 'perspective', 'problem',
'Deepercut', 'deeper', 'model', 'Clustered', 'Nonlinear', 'Appearance', 'Models', 'Huma
n', 'Rmpe', 'multi-person', 'depth', 'images', 'tracking', 'detection']
```

추출된 단어들의 출현 횟수를 확인

In [26]:

```
counts = Counter(noun_list)
words = counts.most_common(50)     # 가장 많이 출현한 횟수 순으로 50개 단어만 추출 한 후
print(words)
```

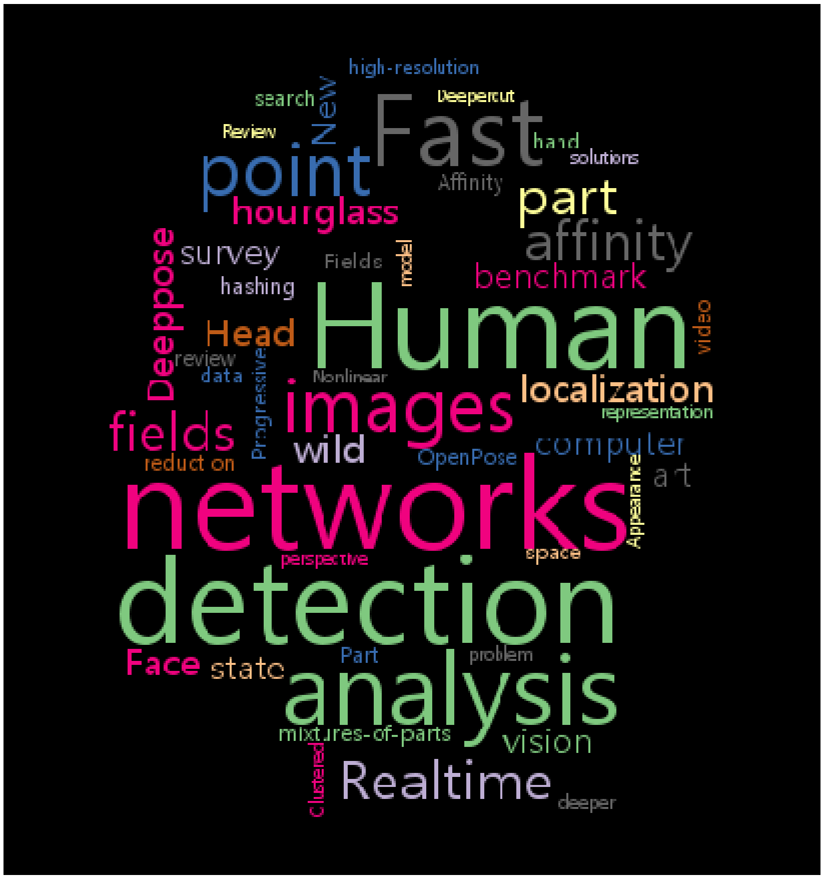
```
[('networks', 2), ('detection', 2), ('Human', 2), ('analysis', 2), ('Fast', 2), ('image
s', 2), ('point', 2), ('Realtime', 1), ('part', 1), ('affinity', 1), ('fields', 1), ('hou
rglass', 1), ('Face', 1), ('localization', 1), ('wild', 1), ('Deeppose', 1), ('Head', 1),
('computer', 1), ('vision', 1), ('survey', 1), ('New', 1), ('benchmark', 1), ('state',
1), ('art', 1), ('mixtures-of-parts', 1), ('OpenPose', 1), ('Part', 1), ('Affinity', 1),
('Fields', 1), ('hashing', 1), ('video', 1), ('hand', 1), ('review', 1), ('data', 1), ('P
rogressive', 1), ('search', 1), ('space', 1), ('reduction', 1), ('high-resolution', 1),
('representation', 1), ('Review', 1), ('solutions', 1), ('perspective', 1), ('problem',
1), ('Deepercut', 1), ('deeper', 1), ('model', 1), ('Clustered', 1), ('Nonlinear', 1),
('Appearance', 1)]
```

단어 출현 횟수에 근거하여 **word cloud** 를 생성

In [27]:

```
cloud_img = plt.imread('ai2.jpg')
wordcloud = WordCloud(font_path='C:/Windows/Fonts/malgun.ttf',
                      colormap = 'Accent_r',
                      width=800,
                      height=600,
                     mask=cloud_img)

print(dict(words))
cloud = wordcloud.generate_from_frequencies(dict(words))
plt.figure(figsize=(10, 8))
plt.axis('off')
plt.imshow(cloud)
plt.show()
```

```
{'networks': 2, 'detection': 2, 'Human': 2, 'analysis': 2, 'Fast': 2, 'images': 2, 'poin
t': 2, 'Realtime': 1, 'part': 1, 'affinity': 1, 'fields': 1, 'hourglass': 1, 'Face': 1,
'localization': 1, 'wild': 1, 'Deeppose': 1, 'Head': 1, 'computer': 1, 'vision': 1, 'surv
ey': 1, 'New': 1, 'benchmark': 1, 'state': 1, 'art': 1, 'mixtures-of-parts': 1, 'OpenPos
e': 1, 'Part': 1, 'Affinity': 1, 'Fields': 1, 'hashing': 1, 'video': 1, 'hand': 1, 'revie
w': 1, 'data': 1, 'Progressive': 1, 'search': 1, 'space': 1, 'reduction': 1, 'high-resolu
tion': 1, 'representation': 1, 'Review': 1, 'solutions': 1, 'perspective': 1, 'problem':
1, 'Deepercut': 1, 'deeper': 1, 'model': 1, 'Clustered': 1, 'Nonlinear': 1, 'Appearance':
1}
```



▶ 결과 [pose estimation 검색 결과 전반적인 keyword 값]

: Human, detection, networks, fast, analysis, localization face, point, hourglass

가장 인용이 많이 된 논문들의 **keyword <DeepPose, Realtime, affinity, face detection, localization, networks>** 와 비슷하지만 살짝 다른 양상

- 인용이 많이 된 논문들의 keyword 들은 전반적으로 최신 알고리즘이나 조금 더 trendy 한 keyword 를 가지고 있고, 전체 keyword 출현 횟수로 분석한 keyword 는 비교적 전문적인 용어보다 일반적인 단어들이 더 많이 나왔다.
- 최신의 trendy 한 keyword 를 분석하기 위해서는 전체적인 키워드 분석보다는 인용이 많이 된 논문들을 선별해 keyword 분석을 하는 것이 더 효과적이다.

In [ ]:

In [ ]: