ДОДАТОК А

ТЕСТОВІ ЗОБРАЖЕННЯ

Рисунок А.1 – Приклад зображення бренду Armani та його дескриптори

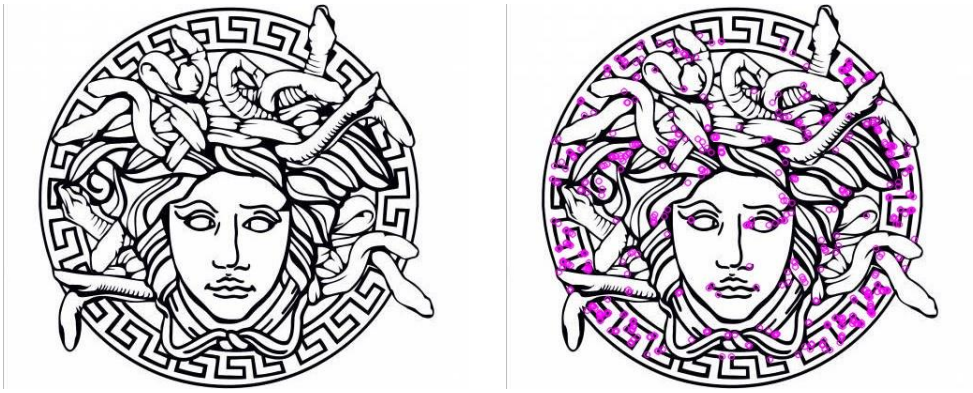

Рисунок А.2 – Приклад зображення бренду Versace та його дескриптори

Рисунок А.3 – Приклад зображення бренду Juicy Couture та його дескриптори

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМНОГО ЗАСТОСУНКУ

```
package main

import (
    "fmt"
    "io/ioutil"
    "math"
    "math/rand"
    "os"
    "strconv"
    "strings"
    "time"
)

const (
    distanceDelta = 1e-7
    iterationCup  = 100
    inputLimit    = 8
    delim         = " "
)

type (
    Input struct {
        args    []float64
        classID int
    }

    Center struct {
        children []Input
        info     string
        position []float64
    }

    SOM struct {
        inputs            []Input
        centers           []Center
        normalizationValue float64
    }
)
```

```
var (
   distFloor   = 0.0
   learningSet []string
   testingSet  []string
   numClasses  int
)

func InitInput(args []float64) Input {
   return Input{args, -1}
}

func InitCenter(pos []float64) Center {
   c := Center{make([]Input, 0), "", pos}
   c.info = "init as\n\n" + c.String() + "\n\n"
   return c
}

func (c *Center) RecalculatePosition() float64 {
   if len(c.children) > 0 {
      newCenter := make([]float64, len((*c).position))
      for _, v := range (*c).children {
         for i := 0; i < len(newCenter); i++ {
            newCenter[i] += v.args[i]
         }
      }
      for i := 0; i < len(newCenter); i++ {
         newCenter[i] /= float64(len((*c).children))
      }
      distance := c.DistanceEuclid(newCenter)
      (*c).position = newCenter
      (*c).info += fmt.Sprintf("->\n\tcenter: %s\n\tchildren: %v\n", c,
len(c.children))
      return distance
   }
   return 1
}

func (c Center) String() string {
   str := "[ "
   for _, v := range c.position {
      str += fmt.Sprintf("%.2f ", v)
   }
   str += " ]\n"
```

```
    return str
}

func (c Center) Save(id int) {
    os.MkdirAll("./info", os.ModePerm)
    err := ioutil.WriteFile(fmt.Sprintf("info/center_%v.txt", id+1), []byte(c.info),
0644)
    if err != nil {
        panic(fmt.Errorf("cannot create center log file for center_%v", id+1))
    }
}


func (c Center) DistanceEuclid(vector []float64) float64 {
    dst := 0.0
    for n := range vector {
        dst += (vector[n] - c.position[n]) * (vector[n] - c.position[n])
    }
    return math.Sqrt(dst)
}

func (c *Center) ClearChildren() {
    (*c).children = make([]Input, 0)
}

func (s *SOM) ClearClasses() float64 {
    maxDistance := 0.0
    for centerID := range (*s).centers {
        centerChangeDistance := (*s).centers[centerID].RecalculatePosition()
        if centerChangeDistance > maxDistance {
            maxDistance = centerChangeDistance
        }
        (*s).centers[centerID].ClearChildren()
    }
    return maxDistance
}

func (s *SOM) RecalculateClasses() {
    for inputID := range s.inputs {
        minDistance := 10000000000.0
        minDistanceID := 0
        for id := range (*s).centers {
            distance := (*s).centers[id].DistanceEuclid((*s).inputs[inputID].args)
            if distance < minDistance {
```

```
                minDistance = distance
                minDistanceID = id
            }
        }
        (*s).centers[minDistanceID].children =
append(s.centers[minDistanceID].children, (*s).inputs[inputID])
        (*s).inputs[inputID].classID = minDistanceID
    }
}

func (s *SOM) AddInput(i Input) {
    (*s).inputs = append((*s).inputs, i)
}

func (s *SOM) AddCenter(c Center) {
    (*s).centers = append((*s).centers, c)
}

func (s SOM) PrintInputsWithClasses() {
    for id, input := range s.inputs {
        fmt.Printf("%v: %v\n", id, input.classID)
    }
}

func (s *SOM) Step() float64 {
    distance := (*s).ClearClasses()
    (*s).RecalculateClasses()
    //(*s).DisplayChange()
    return distance
}

func (s *SOM) StepFixedCenters() {
    for centerID := range (*s).centers {
        (*s).centers[centerID].ClearChildren()
    }
    (*s).RecalculateClasses()
}

func (s *SOM) Normalize() {
    for inputID := range (*s).inputs {
        for argID := range (*s).inputs[inputID].args {
            (*s).inputs[inputID].args[argID] /= (*s).normalizationValue
        }
    }
```

```
}

func (s *SOM) Table(files ...string) (string, []int, int, int) {
    res := fmt.Sprintf("  id\t")
    for n := range s.centers {
        res += fmt.Sprintf("|  %v\t", n+1)
    }
    res += fmt.Sprintf("|  E")
    dsts := make([][]int, len(files))
    dmap := make([]int, 0)
    for id, file := range files {
        (*s).SetInput(file)
        (*s).Normalize()
        (*s).StepFixedCenters()
        res += fmt.Sprintf("\n  %v\t", id+1)
        dsts[id] = make([]int, len((*s).centers))
        for cid, center := range (*s).centers {
            dsts[id][cid] = len(center.children)
            res += fmt.Sprintf("|  %v\t", len(center.children))
        }
        res += fmt.Sprintf("|  %.3f  | %s", s.CheckQuality(), strings.TrimSuffix(file,
".txt"))
    }
    res += "\n"
    sum := 0
    res += "\n      |"
    for i := 0; i < len(dsts); i++ {
        res += fmt.Sprintf(" %4d |", i+1)
    }
    res += "\n" + strings.Repeat("-", 7*(len(dsts)+1)-1) + "|"
    for i := 0; i < len(dsts)-1; i++ {
        res += fmt.Sprintf("\n %4d |%s", i+1, strings.Repeat("      |", i+1))
        for j := i + 1; j < len(dsts); j++ {
            rval := 0
            for c := 0; c < len(dsts[i]); c++ {
                rval += int(math.Abs(float64(dsts[i][c] - dsts[j][c])))
            }
            dmap = append(dmap, rval)
            sum += rval
            res += fmt.Sprintf(" %4d |", rval)
        }
    }
    dmed := 0
    for i := 0; i < len(dmap)-1; i++ {
```

```
        dmed += int(math.Abs(float64(dmap[i] - dmap[i+1])))
    }
    return res, dmap, sum, dmed
}

func (c Center) CheckQuality() float64 {
    result := 0.0
    for _, input := range c.children {
        result += c.DistanceEuclid(input.args) * c.DistanceEuclid(input.args)
    }
    return result
}

func (s SOM) CheckQuality() float64 {
    result := 0.0
    for _, center := range s.centers {
        result += center.CheckQuality()
    }
    result /= float64(len(s.inputs))
    return result
}

func (s SOM) DisplayChange() {
    for _, center := range s.centers {
        fmt.Println(center)
    }
}

func (s *SOM) Iterate() {
    distance := 1.0
    for iter := 0; iter < iterationCup && distance > distanceDelta; iter++ {
        distance = (*s).Step()
    }
}

func InitSOM(classNum int, files ...string) SOM {
    som := SOM{make([]Input, 0), make([]Center, 0), 1.0}
    som.normalizationValue = som.SetInput(files...)
    som.Normalize()
    for centerID := 0; centerID < classNum; centerID++ {
        randomInputId := rand.Intn(len(som.inputs))
        center := InitCenter(som.inputs[randomInputId].args)
        som.AddCenter(center)
    }
```

```
    return som
}

func (s *SOM) SetInput(files ...string) float64 {
    (*s).inputs = make([]Input, 0)
    maxVal := 1.0
    for _, file := range files {
        data, err := ioutil.ReadFile(file)
        if err != nil {
            panic(err)
        }
        str := string(data)
        str = strings.ReplaceAll(str, "\r\n", "\n")

        vectors := strings.Split(str, "\n")
        for _, vector := range vectors {

            argsStrings := strings.Split(vector, delim)
            argsSlice := make([]float64, 0)
            for num, arg := range argsStrings {
                if num > inputLimit-1 {
                    break
                }
                if arg == "" {
                    continue
                }
                value, err := strconv.ParseFloat(arg, 64)
                if err != nil {
                    panic(fmt.Errorf("Unknown number format: %s (%s)\n", arg,
err.Error()))
                }
                if value > maxVal {
                    maxVal = value
                }
                argsSlice = append(argsSlice, value)
            }
            input := InitInput(argsSlice)
            (*s).AddInput(input)
        }
    }
    return maxVal
}

func (s SOM) Save() {
```

```
    err := os.RemoveAll("./info")
    if err != nil {
        panic(err)
    }
    for id := range s.centers {
        s.centers[id].Save(id)
    }
}

func SOMNoClass(maxdev int, dur time.Duration) {
    dev := 0
    start := time.Now()
    i := 0
    for time.Since(start) < dur && maxdev > dev {
        i++
        som := InitSOM(numClasses, learningSet...)
        som.Iterate()
        lrn, _, _, _ := som.Table(learningSet...)
        str, _, sum, _ := som.Table(testingSet...)
        if sum > dev {
            fmt.Printf("ITERATION %v\n\n%s\n\n%s\ndev = %v\n", i, lrn, str, sum)
            som.Save()
            dev = sum
        }
    }
}

func SOMClass(dur time.Duration) {
    dev := 0
    start := time.Now()
    i := 0
    for time.Since(start) < dur {
        i++
        som := InitSOM(numClasses, learningSet...)
        som.Iterate()
        lrn, _, _, _ := som.Table(learningSet...)
        str, _, sum, dv := som.Table(testingSet...)
        coef := sum * dv
        if coef > dev {
            fmt.Printf("ITERATION %v\n\n%s\n\n%s\n\n", i, lrn, str)
            som.Save()
            dev = coef
        }
    }
```

```
}
func main() {
    rand.Seed(time.Now().UnixNano())
    learningSet = []string{"versace_1.txt", "juicy_couture.txt", "burberry.txt"}
    testingSet = []string{"versace_2.txt", "juicy_2.txt", "armani.txt"}
    numClasses = 3
    SOMClass(time.Second * 2)
}
```

Номер докумен-та	*Познака*	*Найменування*	*Прим.*
		Текстові документи	
1	*ГЮІК.062022.013ПЗ*	*Пояснювальна записка*	*66 с.*
2		*Рецензія*	*1 с.*
3		*Науково-технічні публікації (апробація)*	*6 с.*
4		*Сертифікати (дипломи) про участь у науково-технічних*	*1 с.*
		Графічні документи	
5		*Презентаційний матеріал*	*10 с.*
		Електронні матеріали	
6		*2022_Б_ІНФ_ІТІНФ-18-1_Пронюк_О_Д.doc*	
7		*2022_Б_ІНФ_ІТІНФ-18-1_Пронюк_О_Д .pdf*	
8		*2022_Б_ІНФ_ІТІНФ-18-1_Пронюк_О_Д .pptx*	
9		*Каталог з програмою – program*	
10		*2022_Б_ІНФ_ІТІНФ-18-1_Пронюк_О_Д _readme.txt*	
11		*Оригінальність тексту _____ %*	
		Керівник кваліфікаційної роботи	
		проф. Гороховатський В.О._____	

Зм	*Арк*	*№ докум.*	*Підпис*	*Дата*	*ГЮІК.062022.013Д4*		
Розроб.		*Пронюк О. Д.*			*Моделювання методів класифікації зображень на підставі засобів навчання з учителем* *Відомість кваліфікаційної роботи бакалавра*	*Літ*	*Арк*	*Аркуш*
Перевір.		*Гороховатський В.О.*						1
Т.контр						*ХНУРЕ* *Кафедра Інформатики*		
Н.конт		*Бєлова Н.В.*						
Затв.		*Кобилін О.А.*						


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
МАТЕРІАЛИ
XXV МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ
РАДІОЕЛЕКТРОНІКА
ТА МОЛОДЬ
У XXI СТОЛІТТІ
Том 7/10
Харків 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ 25-го МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ»

20-22 квітня 2021 р.
том 7,10

КОНФЕРЕНЦІЯ «СУЧАСНІ МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ»

КОНФЕРЕНЦІЯ «АКТУАЛЬНІ ПРОБЛЕМИ ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ ТА ЕКОНОМІЧНОЇ БЕЗПЕКИ»

Харків 2021

АНАЛІЗ ВЛАСТИВОСТЕЙ МЕТОДУ BRISK

Пронюк О.Д.
Науковий керівник – д.т.н., проф. Гороховатський В. О.
Харківський національний університет радіоелектроніки
61166, Харків, просп. Науки, 14, каф. інформатики, тел. (057) 702-14-19,
e-mail: olena.proniuk@nure.ua

This work discusses the features of BRISK descriptor using at solving image recognition problems. A comparative characteristic of this method and other common detectors is given.

Перспективний сучасний підхід до розпізнавання зображень у системах комп'ютерного зору полягає в тому, що зображення подається як набір ключових точок, внесок яких буде значним у процесі розпізнавання [1, 2]. При цьому процес порівняння зображень розділяється на три етапи. І етап – знаходження множини ключових точок методами-детекторами, у результаті маємо множину координат ключових точок. На ІІ етапі відбувається побудова дескрипторів – векторів ознак для вихідного набору ключових точок. ІІІ етап полягає в порівнянні множин дескрипторів і пошуку точок, що співпадають на обох зображеннях.

Одним з поширених детекторів ключових точок є детектори кутів, в яких вимірюють зміну яскравості пікселя точки.

1. Обирається точка зображення p, для якої буде вирішуватися, чи є вона ключовою. Нехай I_p - яскравість точки. Розглядається окружність навколо обраної точки, наприклад з 16 пікселів (рис. 1).

2. Точка p вважається кутом, якщо серед усіх пікселів окружності існує n пікселів, кожен з яких яскравіший, ніж I_p+t, або темніший, ніж I_p-t. Значення порога t обирається, виходячи із прикладної задачі.

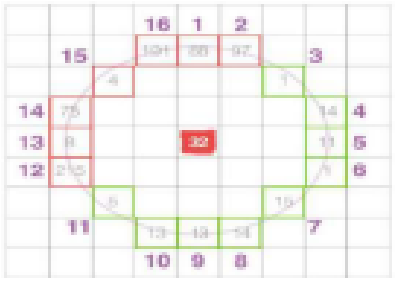

Рисунок 1 – Приклад обрання зони навколо ключової точки

3. Перевіряється інтенсивність точок 1, 5, 9 і 13 з окружності. Якщо хоча б для трьох з них виконується умова п. 2, тоді проводиться перевірка решти з 12-ти пікселів. Інакше обирається наступна точка і алгоритм повторюється. Саме такий підхід забезпечує високу швидкодію обробки даних.

Одним із найбільш ефективних сучасних методів формування дескрипторів є BRISK (Binary Robust Invariant Scalable Keypoints), у якому знаходження максимумів відбувається не тільки на оригінальному зображенні, але і в масштабованому просторі. Такий простір складається з n октав c_i і n внутрішніх октав d_i, i = {0,1, ..., n-1}). Зазвичай n дорівнює 4. Октави складаються шляхом зменшення масштабу вихідного зображення в 2 рази (рис. 2). Кожна внутрішня октава d_i розташовується між октавами c_i і c_{i+1}.

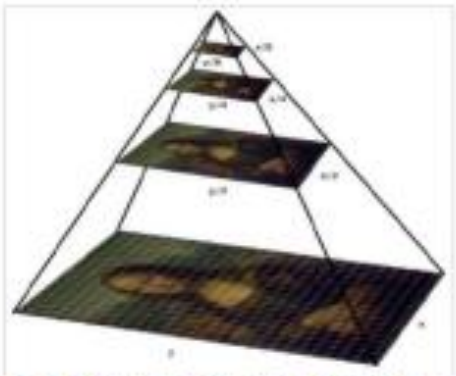

Рисунок 2 – Приклад піраміди зображення розбиттям на октави

Метод BRISK інваріантний до поворотів і частково до змін масштабу та яскравості. При цьому досягнута висока швидкодія реалізації, причому час оброблення не залежить від роздільної здатності зображення [1, 2].

Крім того, метод надає дескриптор у виді бінарного вектору ознак із розміром, кратним ступені двійки. Бінарне подання даних значно прискорює процес зіставлення множин дескрипторів із застосуванням двійкових операцій.

Загалом, BRISK має достатньо високу швидкодію оброблення на зображеннях з високою роздільною здатністю. При якості детектування, порівнянному з якістю детектора SURF і незалежністю від поворотів і зміни масштабу, він придатний для роботи з великими зображеннями, наприклад при пошуку збігів на фотографіях.

Список використаних джерел:

1. Гороховатський В. О. Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення / В.О. Гороховатський, Д.В. Пупченко, К. Г. Солодченко // Системи управління, навігації та зв'язку. 2018. №1 (47). С. 93-98.

2. Бубенчиков М. А. Сравнительный анализ методов нахождения особых точек на изображении, Санкт-Петербургский Университет, 2016. URL: https://nauchkor.ru/pubs/sravnitelnyy-analiz-metodov-nahozhdeniya-osobyh-tochek-na-izobrazhenii-587d363a5f1be77e40d589ec.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ДИПЛОМ

НАГОРОДЖУЄТЬСЯ

ПРОНЮК ОЛЕНА ДМИТРІВНА

(Харківський національний університет радіоелектроніки)

за **ІІІ місце**

на секції «**Математичні моделі і методи нормалізації та аналізу мультимедійних даних**»

конференції

«СУЧАСНІ МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ»

в рамках **25-го** Міжнародного молодіжного форуму

„РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ"

Голова форуму В.В.Семенець

21.04.2021 р.

МЕТОДИ НАВЧАННЯ У ЗАДАЧАХ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Пронюк О.Д.
Науковий керівник – д. т. н., проф. Гороховатський В. О.
Харківський національний університет радіоелектроніки
61166, Харків, просп. Науки, 14, каф. Інформатики,
тел.: +38 (057) 702-14-19
e-mail: olena.proniuk@nure.ua

The application of learning tools using neural networks is being studied in order to implement them in computer vision systems for the classification of visual objects.

Задача розпізнавання зображень є ключовим завданням у системах комп'ютерного зору [1-4]. У роботі пропонується використати мережу Кохонена для автоматичної кластеризації (класифікації без вчителя) наборів дескрипторів ключових точок як опису зображень.

Модель навчання Кохонена має вигляд:

$$\upsilon = \arg \min_{i=1,\dots,k} \rho(\mathrm{x},\mathrm{m}_i) , \qquad (1)$$

де v – номер кластеру, k – кількість центроїдів, $\mathrm{m_i}$ – вектор центроїда, $x \in W$ – вектор з навчальної вибірки, $i = 1,2,\dots,k$.

Етапи обчислень:

1) початковими центрами $\mathrm{M} = \{\mathrm{m}_i\}_{i=1}^{k}$, $\mathrm{m}_i \in W$ вибирають довільні k векторів із W;

2) для кожного $i = \overline{1,k}$ шляхом навчання (1) формують список $W_i \subseteq W$ елементів, для яких найближчим кодуючим вектором є m_i, тобто складають підмножини $W_i = \{\mathrm{x} \in W \mid \arg \min_{\upsilon} \rho(\mathrm{x},\mathrm{m}_\upsilon) = \mathrm{i}\}$; при цьому сукупність $\{W_i\}$ утворює розбиття W: $W = \cup W_i$, $W_i \cap W_j = \varnothing$; кожен вектор попадає виключно до одного із кластерів;

3) в якості чергового значення m_i обчислюють середнє серед елементів списку W_i, складеному в п. 2;

4) кроки 2-3 повторюють кілька разів до досягнення збіжності, коли список $\{\mathrm{m}_i\}$ перестане змінюватися, тобто зміни центрів стають незначними.

Алгоритм апроксимує функцію щільності розподілу множини вхідних зразків за критерієм мінімуму помилки – суми їх квадратів відхилень від центрів сформованих кластерів

$$E = \sum_{i=1}^{k} \sum_{\upsilon=1}^{s(\mathrm{i})} \rho^2(\mathrm{x}_\upsilon, m_i), \qquad (2)$$

де $s(\mathrm{i})$ – потужність i-го кластеру.

Для реалізації задачі були обрані тестові зображення (рис.1), створений опис кожного зображення у вигляді множини з 500 векторів-дескрипторів методом BRISK.

Рисунок 1 – Тестова вибірка зображень

За результатами роботи даної нейронної мережі отримали кластеризацію об’єктів, кількість ознак в кожному кластері представлені в таблиці 1.

Таблиця 1 – Результати програмного моделювання мережі Кохонена

Зображення брендів	Кластери								E
	1	2	3	4	5	6	7	8	
Burberry	62	57	49	73	57	75	76	51	0.393
Juici Couture	69	53	58	47	69	77	75	52	0.365
Versace	48	69	81	68	64	57	46	67	0.379

Навчання мережі Кохонена протестована на 100 ітераціях.

Самоорганізаційна мережа Кохонена перетворила об’ємні багатомірні пакети даних у більш просту структуру фіксованого розміру. Як можна побачити з табл. 1, значення критерію E не перевищує 0,393.

Ефективність класифікації залежить від взаємопов’язаних факторів: база візуальних даних, метод формування дескрипторів, вибір центрів та способів їх формування, метрика для порівняння дескрипторів.

Список використаних джерел

1. Гороховатський В.О., Пупченко Д.В., Солодченко К.Г. Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення. Системи управління, навігації та зв’язку. –2018. – №1 (47). – С. 93–98.

2. Кохонен Т. Самоорганизующиеся карты. М.: БИНОМ. Лаборатория знаний, 2013. – 655 с.

3. Гороховатський, В.О., Гадецька, С.В. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень (монографія), Харків, ФОП Панов А.Н., 128 с.

4. Gorokhovatskyi O., Gorokhovatskyi V., Peredrii O. (2018) Analysis of Application of Cluster Descriptions in Space of Characteristic Image Features. Data, 3(4), 52. – doi: 10.3390/data3040052. Available online: https://www.mdpi.com/2306-5729/3/4/52