IAVI Lab1 Report

第二组 焦笑然 温子奇 张雯琪

实验目的和要求

> •Study the relationship between camera parameters and captured images(pixels) via experiments:
> 1.Exposure
> 2.Gain
> •[Bonus 25%] Derive a quantitative noise model for captured pixels
> 1.What are related parameters?
> 2.What is the equation of the noise?
> 3.How well it explains captured pixels?
> •Download and compile the starter code that will allow you to directly operate on captured images
> •Due date: 09-25•Submit a report along with supporting data and code

实验任务和分析

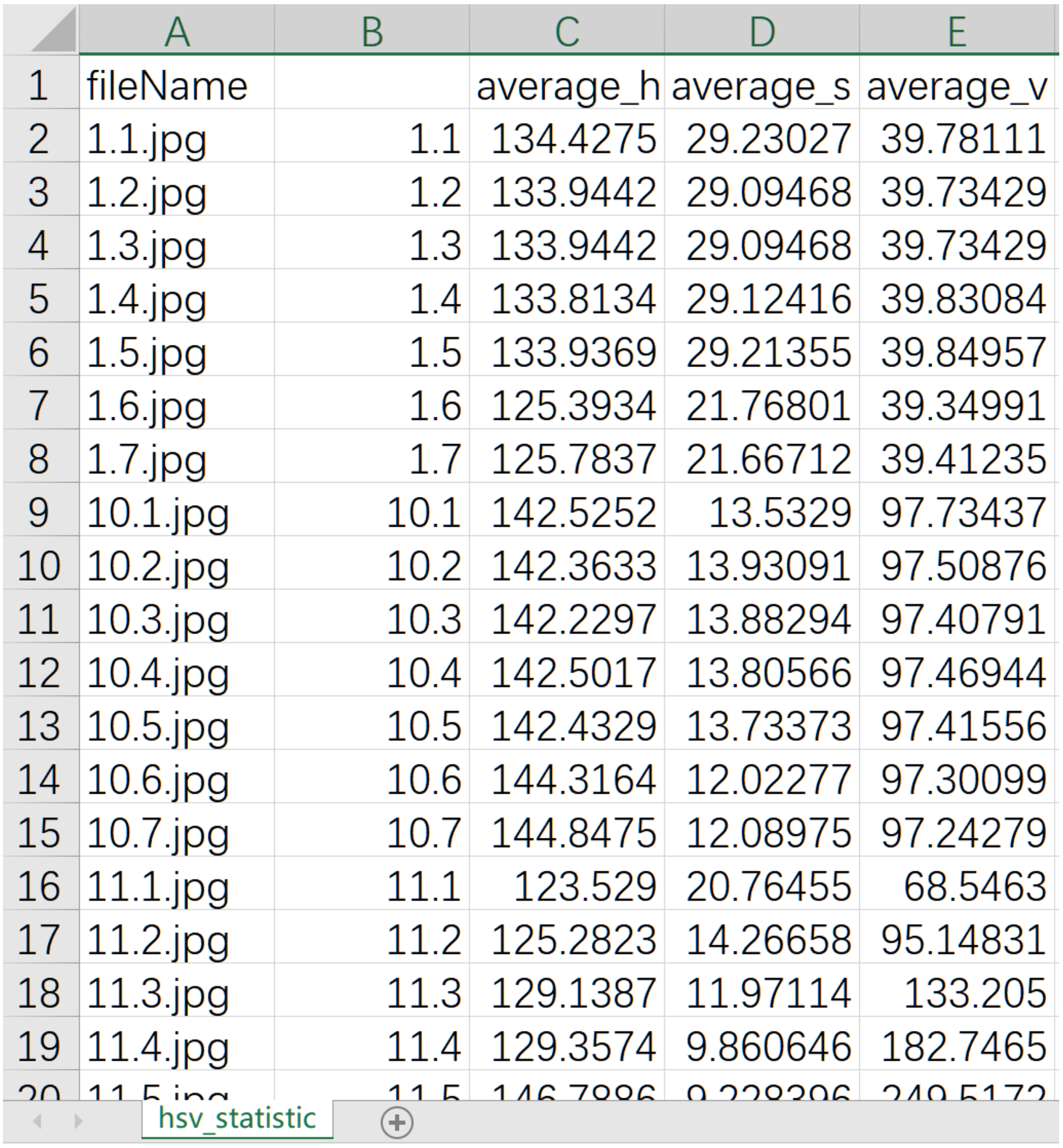
- 任务一 **探究(exposure,gain)参数对(pixels)的影响**
  - 采集数据
    - 尽可能多得拍不同exposure、gain下的照片
  - 数据处理
    - bmp转hsv格式
    - 取v(亮度)
  - 数据分析
    - 用sklearn进行多元线性回归
    - (exposure,gain)->(brightness)
    - 计算相关系数
- 任务二 **探究噪声模型**
  - 如何近似获取真值?
    - 假设：相同条件下各图片噪声均值为0，且噪声不受极短时间间隔影响
    - 蒙特卡洛法
      - 连续、相同条件拍多张图片

            - 求平均值来近似亮度真值
    - 计算亮度的真值、方差（标准差)
        - 以方差来衡量噪声的强度
    - 建模
        - 输入：亮度、exposure、gain...
        - 输出：噪声（方差）
        - 噪声
            - 信号相关噪声 （乘性）
                - 泊松分布
            - 信号不相关噪声（加性）
                - 高斯分布

实验过程和结果

- 任务一
    - **思路**
      要探究exposure、gain等参数对pixel的影响，我们计划通过设定不同的exposure、gain的值来采集一定量的pixel数据进行处理。为简化模型，我们将图片转换为HSV格式，取亮度V来代表像素值。考虑到是研究关于一个因变量和多个自变量之间的相关关系，我们引入多元线性回归模型对数据进行拟合，计算相关系数，最终得到exposure、gain和pixel的关系模型。
    - **操作流程**
      实验中，我们选取一个淡黄色草稿本作为拍摄对象，设定10组不同的exposure和gain的值，每组拍摄7张照片，共得到70张图片。对得到的bmg格式图片进行处理，将得到的HSV值存放入csv格式文件。

代码见附录



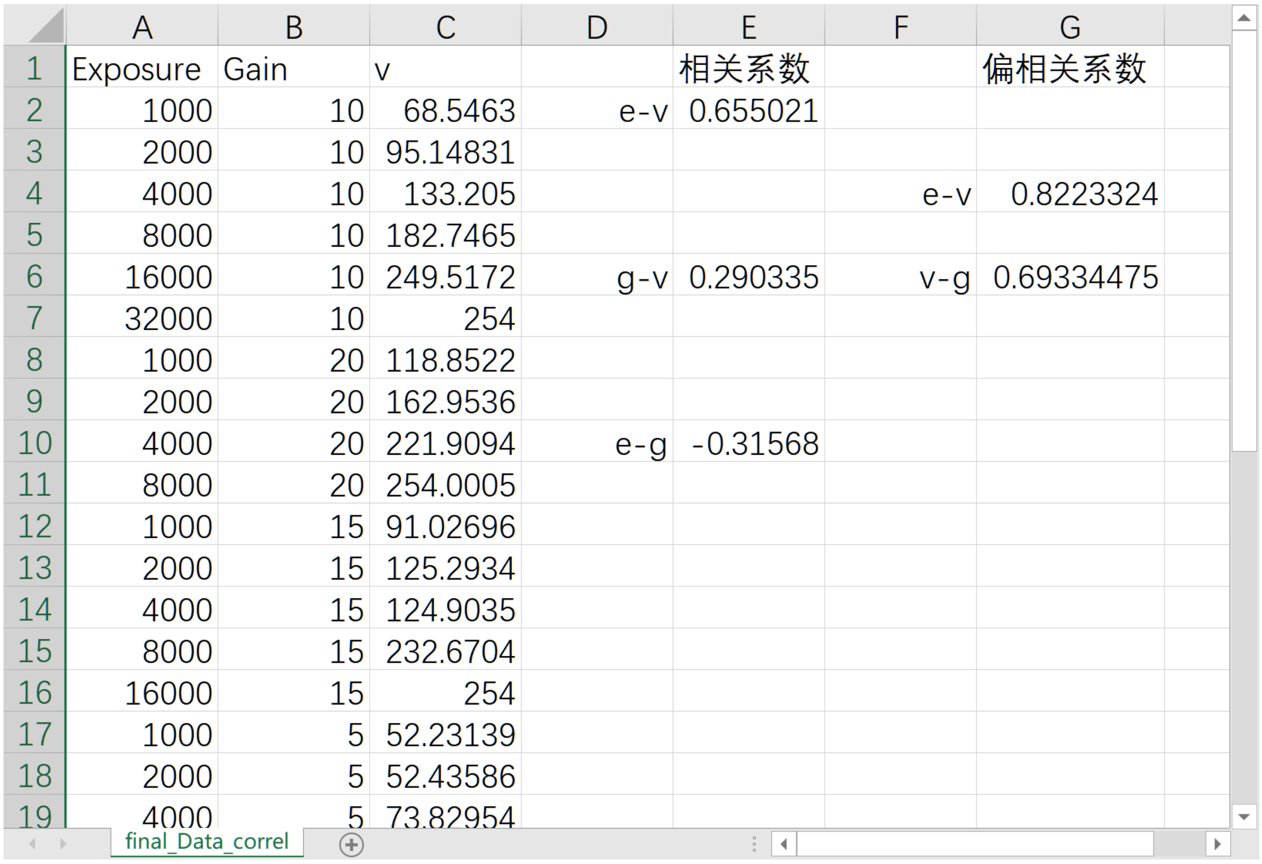
	A	B	C	D	E
1	fileName		average_h	average_s	average_v
2	1.1.jpg	1.1	134.4275	29.23027	39.78111
3	1.2.jpg	1.2	133.9442	29.09468	39.73429
4	1.3.jpg	1.3	133.9442	29.09468	39.73429
5	1.4.jpg	1.4	133.8134	29.12416	39.83084
6	1.5.jpg	1.5	133.9369	29.21355	39.84957
7	1.6.jpg	1.6	125.3934	21.76801	39.34991
8	1.7.jpg	1.7	125.7837	21.66712	39.41235
9	10.1.jpg	10.1	142.5252	13.5329	97.73437
10	10.2.jpg	10.2	142.3633	13.93091	97.50876
11	10.3.jpg	10.3	142.2297	13.88294	97.40791
12	10.4.jpg	10.4	142.5017	13.80566	97.46944
13	10.5.jpg	10.5	142.4329	13.73373	97.41556
14	10.6.jpg	10.6	144.3164	12.02277	97.30099
15	10.7.jpg	10.7	144.8475	12.08975	97.24279
16	11.1.jpg	11.1	123.529	20.76455	68.5463
17	11.2.jpg	11.2	125.2823	14.26658	95.14831
18	11.3.jpg	11.3	129.1387	11.97114	133.205
19	11.4.jpg	11.4	129.3574	9.860646	182.7465
20	11.5.jpg	11.5	146.7886	9.228396	249.5172

调用SKlearn库中的多元线性回归模型，拟合结果为：Value = 0.0036EXPOSURE + 5.56442GAIN + 59.97

代码见附录

最后计算e(Exposure)、g(Gain)、V(Value)的相关系数，考虑到在多元相关分析中，变量之间可能受到不止一个变量的影响，简单相关系数可能不能够真实的反映出变量之间的相

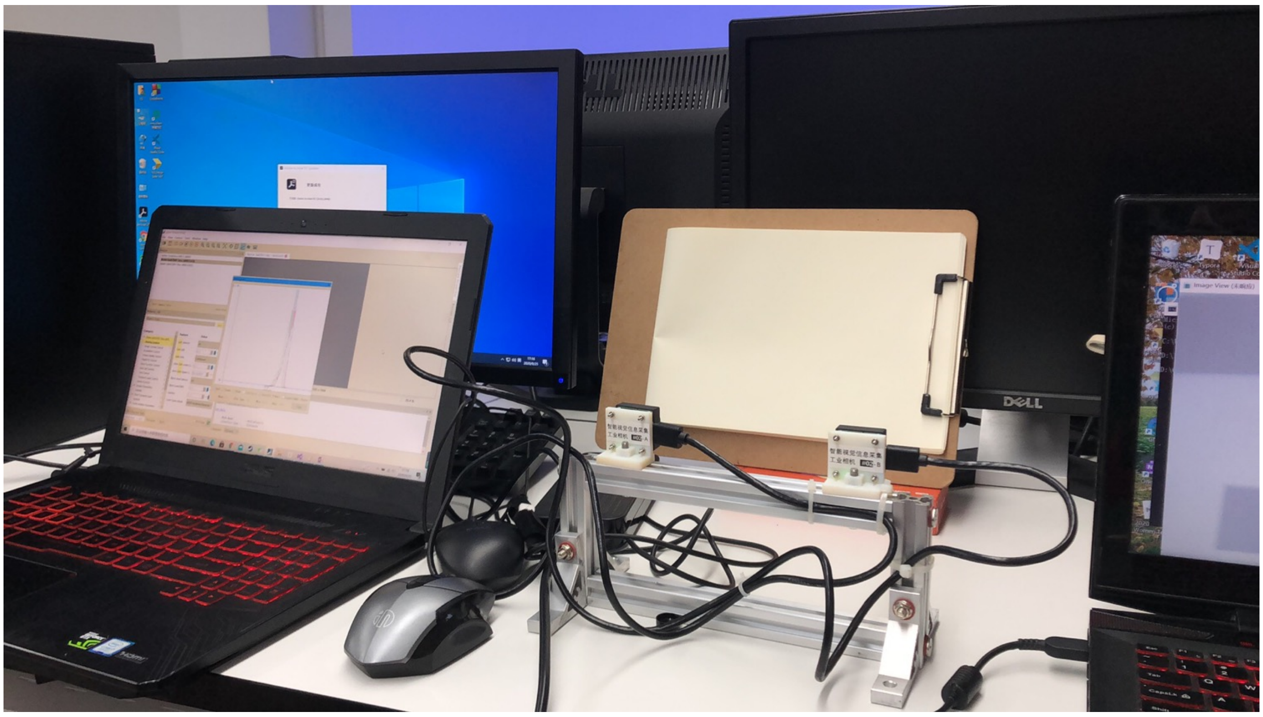
关性，故而我们通过相关系数来计算出偏相关系数。



	A	B	C	D	E	F	G
1	Exposure	Gain	v		相关系数		偏相关系数
2	1000	10	68.5463	e-v	0.655021		
3	2000	10	95.14831				
4	4000	10	133.205			e-v	0.8223324
5	8000	10	182.7465				
6	16000	10	249.5172	g-v	0.290335	v-g	0.69334475
7	32000	10	254				
8	1000	20	118.8522				
9	2000	20	162.9536				
10	4000	20	221.9094	e-g	-0.31568		
11	8000	20	254.0005				
12	1000	15	91.02696				
13	2000	15	125.2934				
14	4000	15	124.9035				
15	8000	15	232.6704				
16	16000	15	254				
17	1000	5	52.23139				
18	2000	5	52.43586				
19	4000	5	73.82954				

- **结论**
  本实验中我们只考虑了改变相机曝光时间来改变Exposure参数，在不过曝的前提下，增加曝光时间可以使图像更加清晰；gain通过改变相机的感官性能，增大gain在exposure较小时能有效提高图像清晰度。

- 任务二
  - **思路**
    我们对于图像噪声的简单理解是：相机拍摄所获得的像素值与实际物理上的像素真值之间的差异；这里为了方便起见，用HSV颜色模型中的明度V来代表一个像素的像素值。
    因此，我们选择一个物理上的像素真值一致的物理作为拍摄对象，本实验中采用一个淡黄色草稿本作为拍摄对象，并假设这一草稿纸上处处颜色完全一致，并在无遮挡的平行光环境下进行拍摄，这样，按照假设，我们相机所获取的图像的像素值在没有噪声的情况下，其各个像素点的明度V应当是一致的。



当存在噪点时，某些像素点的明度V会偏移实际的真值，因此我们用一张图片像素值均方误差作为一张图片的噪声的衡量指标。

在拍摄过程中，我们注意到，随着曝光时间exposure的增加，图像的噪点更加明显，因此我们做出第一条假设：

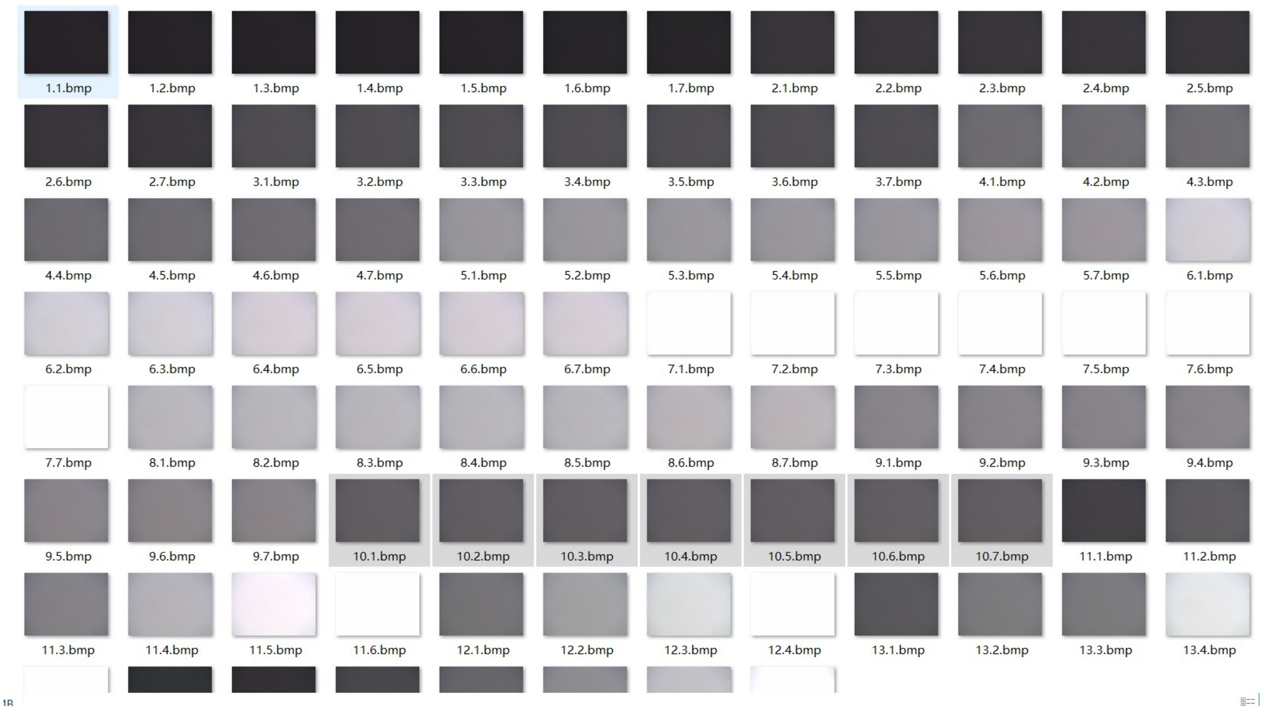
1. 存在一种噪声与信号的输入相关，根据定性观察的结果，暂时认为输入的信号值越大，噪声越大；

其实，根据常识中对于噪声的认识：一种随机产生的误差，因此我们作出第二条假设：

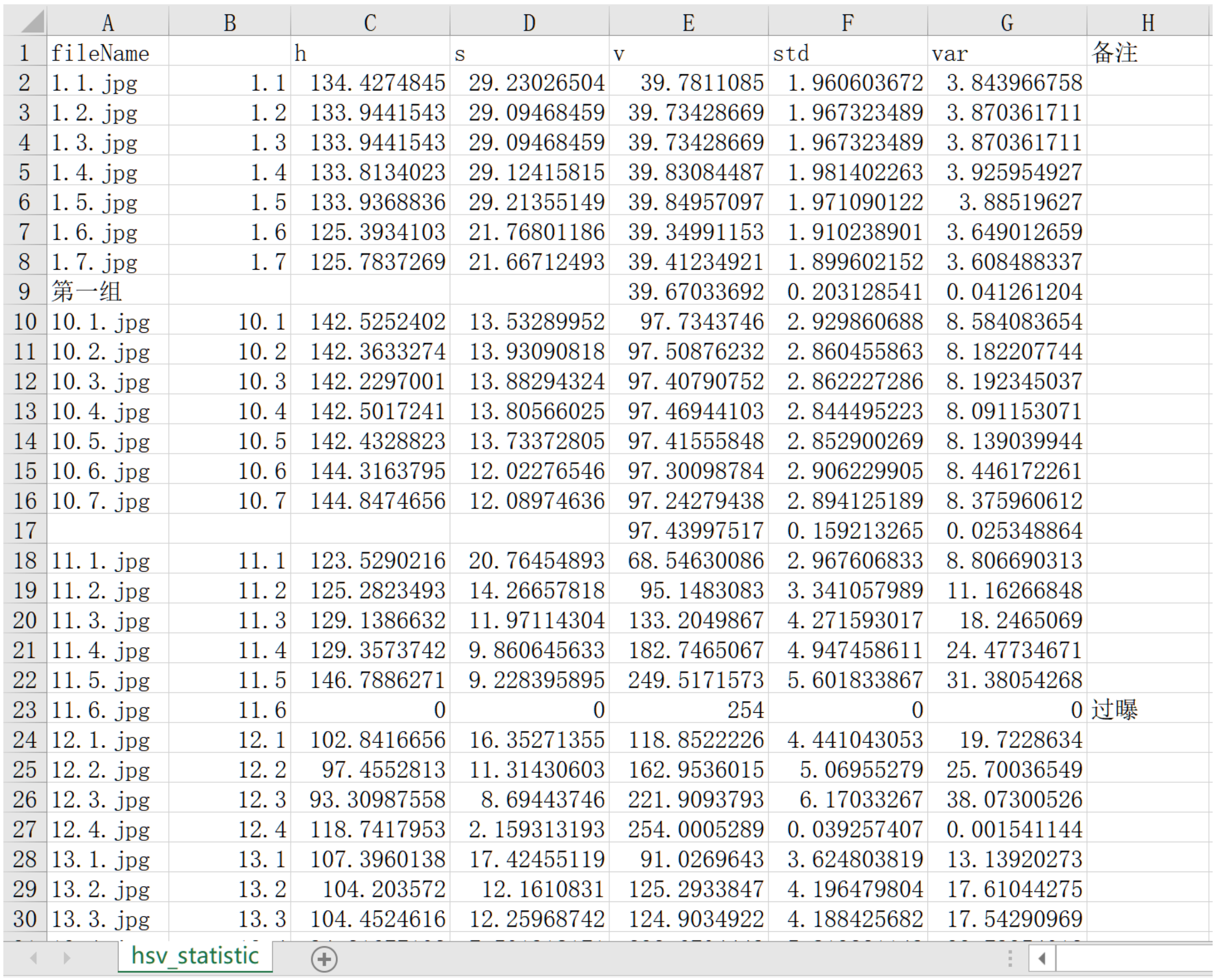
1. 存在一种无规律的噪声（与信号输入无关），在时间上随机分布，且其均值为0，意味着可以通过对同一物体的多次拍摄来降低这一噪声。

- 操作流程
  综上，根据以上两种假设，我们对于拍摄对象（淡黄色草稿纸），在不同的曝光时间下，进行多次拍照，一共获得10种曝光条件下的70张照片（每种曝光条件下重复拍摄7次）。并

计算出每一张照片的平均明度，以及其自身明度值的均方误差。

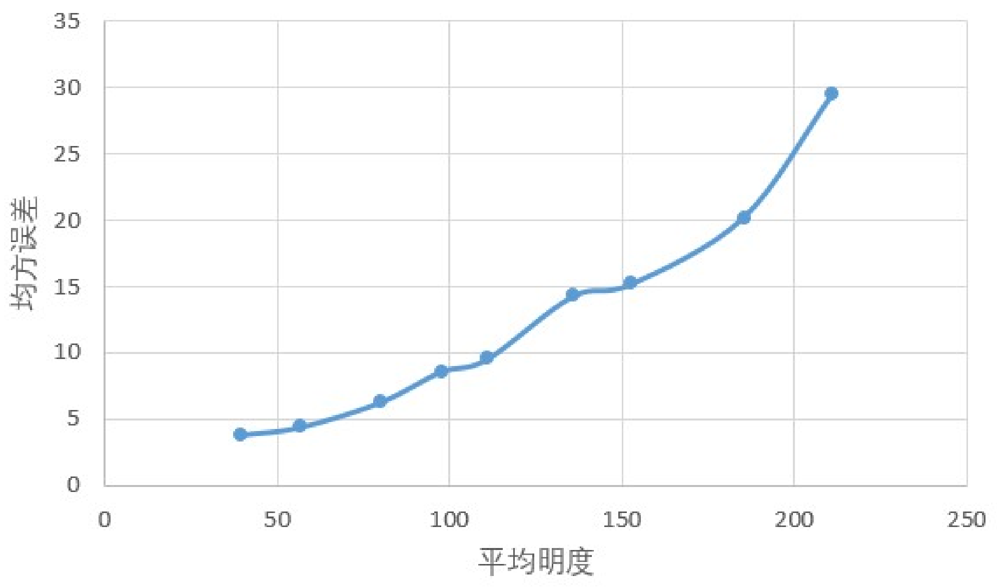


处理后的数据集包含每组的V均值、方差、标准差，以及每个图片自身亮度值的标准差和方差。



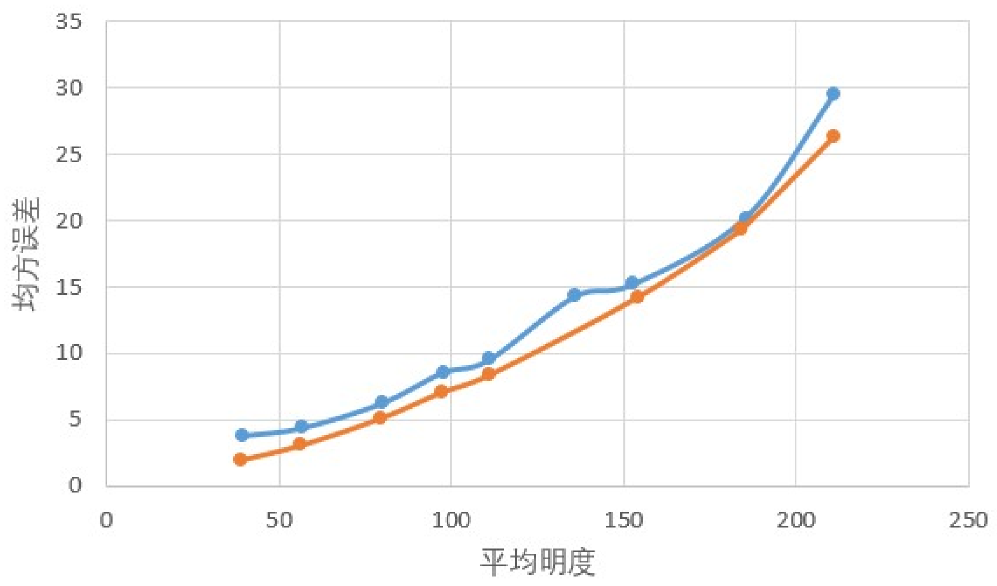
	A	B	C	D	E	F	G	H
1	fileName		h	s	v	std	var	备注
2	1. 1. jpg	1. 1	134. 4274845	29. 23026504	39. 7811085	1. 960603672	3. 843966758	
3	1. 2. jpg	1. 2	133. 9441543	29. 09468459	39. 73428669	1. 967323489	3. 870361711	
4	1. 3. jpg	1. 3	133. 9441543	29. 09468459	39. 73428669	1. 967323489	3. 870361711	
5	1. 4. jpg	1. 4	133. 8134023	29. 12415815	39. 83084487	1. 981402263	3. 925954927	
6	1. 5. jpg	1. 5	133. 9368836	29. 21355149	39. 84957097	1. 971090122	3. 88519627	
7	1. 6. jpg	1. 6	125. 3934103	21. 76801186	39. 34991153	1. 910238901	3. 649012659	
8	1. 7. jpg	1. 7	125. 7837269	21. 66712493	39. 41234921	1. 899602152	3. 608488337	
9	第一组				39. 67033692	0. 203128541	0. 041261204	
10	10. 1. jpg	10. 1	142. 5252402	13. 53289952	97. 7343746	2. 929860688	8. 584083654	
11	10. 2. jpg	10. 2	142. 3633274	13. 93090818	97. 50876232	2. 860455863	8. 182207744	
12	10. 3. jpg	10. 3	142. 2297001	13. 88294324	97. 40790752	2. 862227286	8. 192345037	
13	10. 4. jpg	10. 4	142. 5017241	13. 80566025	97. 46944103	2. 844495223	8. 091153071	
14	10. 5. jpg	10. 5	142. 4328823	13. 73372805	97. 41555848	2. 852900269	8. 139039944	
15	10. 6. jpg	10. 6	144. 3163795	12. 02276546	97. 30098784	2. 906229905	8. 446172261	
16	10. 7. jpg	10. 7	144. 8474656	12. 08974636	97. 24279438	2. 894125189	8. 375960612	
17					97. 43997517	0. 159213265	0. 025348864	
18	11. 1. jpg	11. 1	123. 5290216	20. 76454893	68. 54630086	2. 967606833	8. 806690313	
19	11. 2. jpg	11. 2	125. 2823493	14. 26657818	95. 1483083	3. 341057989	11. 16266848	
20	11. 3. jpg	11. 3	129. 1386632	11. 97114304	133. 2049867	4. 271593017	18. 2465069	
21	11. 4. jpg	11. 4	129. 3573742	9. 860645633	182. 7465067	4. 947458611	24. 47734671	
22	11. 5. jpg	11. 5	146. 7886271	9. 228395895	249. 5171573	5. 601833867	31. 38054268	
23	11. 6. jpg	11. 6	0	0	254	0	0	过曝
24	12. 1. jpg	12. 1	102. 8416656	16. 35271355	118. 8522226	4. 441043053	19. 7228634	
25	12. 2. jpg	12. 2	97. 4552813	11. 31430603	162. 9536015	5. 06955279	25. 70036549	
26	12. 3. jpg	12. 3	93. 30987558	8. 69443746	221. 9093793	6. 17033267	38. 07300526	
27	12. 4. jpg	12. 4	118. 7417953	2. 159313193	254. 0005289	0. 039257407	0. 001541144	
28	13. 1. jpg	13. 1	107. 3960138	17. 42455119	91. 0269643	3. 624803819	13. 13920273	
29	13. 2. jpg	13. 2	104. 203572	12. 1610831	125. 2933847	4. 196479804	17. 61044275	
30	13. 3. jpg	13. 3	104. 4524616	12. 25968742	124. 9034922	4. 188425682	17. 54290969	

为了验证假设1：取曝光时间逐渐增大的10张图片，横轴为平均明度，纵轴为明度的均方误差，绘制曲线：



可以看出，排除因为过曝而导致均方误差基本为0的图片，随着曝光时间增加（也就是平均明度增加），均方误差也在增加（即第一种噪声）。

为了验证假设2：将同一张物体重复拍摄的7张图片融合为1张（融合后的图片的每个像素值为7张图片对应像素值的平均值），最终获得10张新的图片，并绘制相应的曲线图，与未融合的图片的曲线图进行对比：(下图中蓝色曲线为拍摄的原数据，橙色曲线为7张图片经过融合后获得的图像数据)



可以发现，每处的均方误差均有下降，但未降低至最低，且依然符合假设1所提出的分布规律：这同时说明了两点：

存在与信号输入无关的噪声，可以通过取平均值来过滤；但依然存在与信号相关的噪声，无法通过平均值来过滤。

- 结论

  通过以上流程，我们总结出如下的噪声模型：

  至少存在两种噪声，一种与图像的输入信号有关，一种与图像的输入信号无关；

  与输入信号有关的噪声，其噪声幅度随输入信号幅度增加而增加；

与输入信号无关的噪声，其噪声在时间轴上随机分布，且其在时间轴上的均值为0，可以通过对同一物体进行多次拍摄的方式来过滤该噪声。

参考资料

[1]图解噪声与去噪之三：噪声建模与去噪 https://zhuanlan.zhihu.com/p/20947568
[2]图解噪声与去噪 https://zhuanlan.zhihu.com/p/20774510
[3]Opencv实现两幅图像融合 https://blog.csdn.net/dcrmg/article/details/52040561?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param

附录

- 将bmp格式图片转换为HSV值，存放入csv格式文件代码如下：

```
import numpy as np
import cv2 as cv
import os
from PIL import Image
from matplotlib import pyplot as plt
import csv

# bmp 转换为jpg
def bmpToJpg(file_path):
    for fileName in os.listdir(file_path):
        # print(fileName)
        newFileName = fileName[0:fileName.find(".")] + ".jpg"
        print(newFileName)
        im = Image.open(file_path + "\\" + fileName)
        im.save(file_path + "\\jpg\\" + newFileName)

def main():
    #bmpToJpg("D:\Courses\\2020\iavi\lab1\img\img_bmp")
    cal_hsv()


def cal_hsv():
    exposure = []
    gain = []
    average_v = []
    average_s = []
    average_h = []

    # 读入exposure & gain
    with open("D:\Courses\\2020\iavi\lab1\week1_raw_metadata.csv") as f:
        f_csv = csv.reader(f)
        for row in f_csv:
            exposure.append(row[0])
            gain.append(row[1])

    file_path = "D:\Courses\\2020\iavi\lab1\img\\img_bmp\\jpg\\"

    i = 0
    for fileName in os.listdir(file_path):
        img = cv.imread(fileName)
        hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
        H, S, V = cv.split(hsv)
        v = V.ravel()[np.flatnonzero(V)]   #亮度非零的值
        s = S.ravel()[np.flatnonzero(S)]
        h = H.ravel()[np.flatnonzero(H)]
        average_v.append(sum(v)/len(v))         #平均亮度
        average_s.append(sum(s)/len(s))
        average_h.append(sum(h)/len(h))
        # print(average_v)

    with open('D:\Courses\\2020\iavi\lab1\\hsv_statistic.csv','w')as f:
        f_csv = csv.writer(f)
        for i in range(0,17):
            row = [i+1,exposure[i],gain[i],average_h[i],average_s[i],average_v[i]]
            f_csv.writerow(row)


if __name__ == '__main__':
    main()
```

- 调用sklearn库进行多元线性回归拟合代码如下：

```
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

# some usable model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LinearRegression
```

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


def age(dataframe):
    dataframe.loc[dataframe['age'] <= 32, 'age'] = 1
    dataframe.loc[(dataframe['age'] > 32) & (dataframe['age'] <= 47), 'age'] = 2
    dataframe.loc[(dataframe['age'] > 47) & (dataframe['age'] <= 70), 'age'] = 3
    dataframe.loc[(dataframe['age'] > 70) & (dataframe['age'] <= 98), 'age'] = 4

    return dataframe

def duration(data):

    data.loc[data['duration'] <= 102, 'duration'] = 1
    data.loc[(data['duration'] > 102) & (data['duration'] <= 180)  , 'duration']    = 2
    data.loc[(data['duration'] > 180) & (data['duration'] <= 319)  , 'duration']   = 3
    data.loc[(data['duration'] > 319) & (data['duration'] <= 644.5), 'duration'] = 4
    data.loc[data['duration']  > 644.5, 'duration'] = 5

    return data

def data_preprocess(bank):

    y = pd.get_dummies(bank['y'], columns=['y'], prefix=['y'], drop_first=True)
    bank_client = bank.iloc[:, 0:7]
    bank_related = bank.iloc[:, 7:11]
    bank_se = bank.loc[:, ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m',
'nr.employed']]
    bank_o = bank.loc[:, ['campaign', 'pdays', 'previous', 'poutcome']]


    labelencoder_X = LabelEncoder()
    bank_client['job'] = labelencoder_X.fit_transform(bank_client['job'])
    bank_client['marital'] = labelencoder_X.fit_transform(bank_client['marital'])
    bank_client['education'] = labelencoder_X.fit_transform(bank_client['education'])
    bank_client['default'] = labelencoder_X.fit_transform(bank_client['default'])
    bank_client['housing'] = labelencoder_X.fit_transform(bank_client['housing'])
    bank_client['loan'] = labelencoder_X.fit_transform(bank_client['loan'])
    bank_client = age(bank_client)

    labelencoder_X = LabelEncoder()
    bank_related['contact'] = labelencoder_X.fit_transform(bank_related['contact'])
    bank_related['month'] = labelencoder_X.fit_transform(bank_related['month'])
    bank_related['day_of_week'] = labelencoder_X.fit_transform(bank_related['day_of_week'])
    bank_related = duration(bank_related)

    bank_o['poutcome'].replace(['nonexistent', 'failure', 'success'], [1, 2, 3], inplace=True)

    bank_final = pd.concat([bank_client, bank_related, bank_se, bank_o], axis=1)
    bank_final = bank_final[['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
                             'contact', 'month', 'day_of_week', 'duration', 'emp.var.rate',
'cons.price.idx',
                             'cons.conf.idx', 'euribor3m', 'nr.employed', 'campaign', 'pdays',
'previous', 'poutcome']]


    return bank_final, y

def LRpredict(X_train, X_test, y_train):

    # your code here begin
    # train your model on 'x_train' and 'x_test'
    # predict on 'y_train' and get 'y_pred'
    logmodel = LogisticRegression()
    logmodel.fit(X_train, y_train)
    y_pred = logmodel.predict(X_test)
    LOGCV = (cross_val_score(logmodel, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())
    # your code here end

    return y_pred,LOGCV

def KNNpredict(X_train, X_test, y_train):

    # your code here begin
    # train your model on 'x_train' and 'x_test'
    # predict on 'y_train' and get 'y_pred'
    knn = KNeighborsClassifier(n_neighbors=22)
    knn.fit(X_train, y_train)
    knnpred = knn.predict(X_test)
    KNNCV = (cross_val_score(knn, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())     #
your code here end

    return knnpred,KNNCV

def SVCpredict(X_train, X_test, y_train):

    svc = SVC(kernel='sigmoid')
    svc.fit(X_train, y_train)
    svcpred = svc.predict(X_test)

    SVCCV = (cross_val_score(svc, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())

    return svcpred , SVCCV

def DTreepredict(X_train, X_test, y_train):

    dtree = DecisionTreeClassifier(criterion='gini')  # criterion = entopy, gini
    dtree.fit(X_train, y_train)
    dtreepred = dtree.predict(X_test)

    DTREECV = (cross_val_score(dtree, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())

    return dtreepred, DTREECV

def RFCpredict(X_train, X_test, y_train):

    rfc = RandomForestClassifier(n_estimators=200)  # criterion = entopy,gini
    rfc.fit(X_train, y_train)
    rfcpred = rfc.predict(X_test)

    RFCCV = (cross_val_score(rfc, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())

    return rfcpred, RFCCV

def Gausspredict(X_train, X_test, y_train):

    gaussiannb = GaussianNB()
    gaussiannb.fit(X_train, y_train)
    gaussiannbpred = gaussiannb.predict(X_test)
    probs = gaussiannb.predict(X_test)

    GAUSIAN = (cross_val_score(gaussiannb, X_train, y_train, cv=k_fold, n_jobs=1,
scoring='accuracy').mean())
```

```
    return probs, GAUSIAN



def GBKpredict(X_train, X_test, y_train):

    gbk = GradientBoostingClassifier()
    gbk.fit(X_train, y_train)
    gbkpred = gbk.predict(X_test)
    print(confusion_matrix(y_test, gbkpred))
    print(round(accuracy_score(y_test, gbkpred), 2) * 100)
    GBKCV = (cross_val_score(gbk, X_train, y_train, cv=k_fold, n_jobs=1, scoring='accuracy').mean())

    return gbkpred, GBKCV



def split_data(data):
    y = data.y
    x = data.loc[:, data.columns != 'y']
    x = data_preprocess(x)
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
    return x_train, x_test, y_train, y_test

def print_result(y_test, y_pred):
    report = confusion_matrix(y_test, y_pred)
    precision = report[1][1] / (report[:, 1].sum())
    recall = report[1][1] / (report[1].sum())
    print('model precision:' + str(precision)[:4] + ' recall:' + str(recall)[:4])

if __name__ == '__main__':
    #bank = pd.read_csv('bank-additional-full.csv', sep=';')

    #bank_final, y = data_preprocess(bank)
    #k_fold = KFold(n_splits=10, shuffle=True, random_state=0)

    #X_train, X_test, y_train, y_test = train_test_split(bank_final, y, test_size=0.1942313295,
random_state=101)

    #sc_X = StandardScaler()
    #X_train = sc_X.fit_transform(X_train)
    #X_test = sc_X.transform(X_test)

    #y_pred,a = KNNpredict(X_train, X_test, y_train)

    #print('KNN Reports\n', classification_report(y_test, y_pred))
    #print('Accuracy:',accuracy_score(y_test, y_pred))
    #print_result(y_test, y_pred)

    data = pd.read_csv("final_Data.csv")
    list = data.values.tolist()
    list_ndarray = np.array(list)
    x_train = list_ndarray[:,0:2]
    x_train = x_train.tolist()
    y_train = list_ndarray[:,2:]
    y_train = y_train.tolist()

    list_t = [y_train[i][0] for i in range(len(y_train))]
    y_train = list_t
    model=LinearRegression()
    model.fit(x_train,y_train)
    w = model.coef_
    b = model.intercept_  # 得到bias值
    print(len(w))  # 输出参数数目
    print([round(i, 5) for i in w])  # 输出w列表，保留5位小数
    print(b)  # 输出bias
```

- 将7张图片融合成1张图片的代码如下：

```
#include<opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
using namespace cv;

int main(int argc, char* argv[]) {
    Mat imagesrc1, imagesrc2, imagesrc3, imagesrc4, imagesrc5, imagesrc6, imagesrc7;
    Mat image_1, image_2, image_3, image_4, image_5;
    imagesrc1 = cv::imread("E:/Code/C++/base/img_10/10.1.bmp");//读取图像1
    imagesrc2 = cv::imread("E:/Code/C++/base/img_10/10.2.bmp");//读取图像2
    imagesrc3 = cv::imread("E:/Code/C++/base/img_10/10.3.bmp");//读取图像1
    imagesrc4 = cv::imread("E:/Code/C++/base/img_10/10.4.bmp");//读取图像2
    imagesrc5 = cv::imread("E:/Code/C++/base/img_10/10.5.bmp");//读取图像1
    imagesrc6 = cv::imread("E:/Code/C++/base/img_10/10.6.bmp");//读取图像2
    imagesrc7 = cv::imread("E:/Code/C++/base/img_10/10.7.bmp");//读取图像1
    //判断读入是否成功
    if (!imagesrc1.data | !imagesrc2.data) {
        std::cout << "打开图片失败，请检查路径！" << std::endl;
        return 0;
    }
    //调整image2的大小与image1的大小一致，融合函数addWeighted()要求输入的两个图形尺寸相同
    //resize(imagesrc2, imagesrc2, Size(imagesrc1.cols, imagesrc1.rows));
    addWeighted(imagesrc1, 0.5, imagesrc2, 0.5, 0, image_1);
    addWeighted(imagesrc3, 0.5, imagesrc4, 0.5, 0, image_2);
    addWeighted(imagesrc5, 0.5, imagesrc6, 0.5, 0, image_3);
    addWeighted(image_1, 0.5, image_2, 0.5, 0, image_4);
    addWeighted(image_3, 0.667, image_4, 0.333, 0, image_5);
    addWeighted(image_3, 0.667, image_4, 0.333, 0, image_5);
    addWeighted(image_5, 0.857, image_3, 0.143, 0, image_1);
    imshow("效果图", image_1);
    //建立显示窗口
    namedWindow("效果图");
    imwrite("E:/Code/C++/base/img_10/out.bmp", image_1);
    waitKey();
    return 0;
}
```