Cattle Recognition in an Unconstrained Environment

submitted in partial fulfillment of the requirements

of

the degree of

Master Of Technology

by

Rohan Rawat
(202211012)

Supervisor :
Dr. Rishav Singh
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY DELHI
2022

Acknowledgement

I would like to express my sincere gratitude to my advisor Dr. Rishav Singh for his support and guidance throughout my research. His patience, motivation and immense knowledge helped me to carry out my thesis successfully. I could not have imagined having a better advisor and mentor for my M. Tech study.

I would also like to thank the faculty members at National Institute Of Technology Delhi for their insightful suggestions, and perceptive comments.

Finally, I would like to thank my family and friends who supported me throughout my work.

Rohan Rawat
202211012
Department of Computer Science and Engineering
National Institute Of Technology Delhi

Contents

List of Figures

List of Tables

Abstract

With the growth in the demand of food supply requirements, the size of the farms is increasing worldwide. This has led to the need for a better livestock identification system & recently it is receiving great research attention. Some of the classical approaches in this area include collar-id, tags on the ear, freeze branding, hot-iron marking on the body, inserting microchip inside the body, and sketch pattern-based marking. While these methods are popularly used around the world, there are some significant issues with these approaches, like ear-tags or pattern marking can fade easily, they can be eventually damaged, duplication fraud can be done easily, and it's also uncomfortable for cattle. Due to these reasons, many farm owners are seeking a smart cattle recognition system that can be used to track animals in the herd, given the poor farm environment.

In this study, work has been done on the cattle muzzle dataset which contains 395 classes of individual cattle's. The muzzle region of the cattle is unique for each individual cattle which can be used as a distinguishing feature. This thesis proposes an effective cattle identification method that can work in unconstrained environment in near-real time. The proposed methodology takes cattle image as input, localizes the muzzle pattern, extracts features from it and identifies the cattle. First localization of muzzle pattern is done using YOLOv3, then an indexing technique is used that takes this muzzle pattern to filter out top-K classes from the cattle database and then one-to-one verification is done by feature extraction and matching using Scale Invariant Feature Transform (SIFT) algorithm. Here, for 395 cattle classes we achieved an accuracy of around 79.8% for identification and implementation gives a near-real time performance of 4-5 seconds on average.

This proposed method can be used for identification in a dynamic environment where database may change over time. It can also be extended and used for other livestock and pets.

Chapter 1

Introduction

Animal biometrics is an emerging field seeking a lot of attention of researchers. As the size of the farms is increasing, they are seeking an automatic way to detect, register, identify, verify and track cattle on the fly. Plus a reliable system is required that can be used for tracking animals in herd, given poor farm environment.

Some of the classical approaches in this area includes collar-id, tags on ear, freeze branding, hot-iron marking on body, inserting microchip inside body, sketch pattern-based marking. While these methods are popularly used around the world, there are some major issues with these approaches like ear-tags or pattern marking can fade easily, they can be eventually damaged, duplication fraud can be done easily and its also uncomfortable for cattle.

Computer vision is the field which takes digital images as input and tries to extract meaningful information from it. The main purpose is to allow computer see and process images as we humans do. Machine learning and deep learning are mostly used for the implementing the tasks of computer vision. Animal biometrics is also one application of computer vision only, as it takes the pictures of livestock and tries to identify them. In biometrics, the phenotype presentations are the observable physical traits of an organism. These traits are unique for different individual livestock which can be used for identifying a particular livestock.

Animal identification strategies may be classified into the following classes based on published literature: There are three types of permanent recognition methods: (1) permanent recognition, (2) semi-permanent recognition, and (3) sketch pattern-based marking recognition. Making-based animal identification techniques are part of the permanent recognition approach. These marking strategies include ear tattoo-based identifying procedures, microchip-based intrusive marking, and hot-iron and freeze branding of animals. Animal marking systems based on ear tattoos are a manual way of identifying large herds of pastoral animals and other species. Solid color animal breeds are not identified using sketch patterning-based marker approaches (especially cattle breeds).

Several cattle breed associations, such as Brown Swiss, Red Poll, and Milking Shorthorn, can benefit from ear tattoo-based tagging systems for tracking and identification. Individual calves must be identified using different artificial marking techniques such as microchip implantation, ear tattooing, and hot-iron procedures, yet these treatments result in flaws in the cow.

On the other hand, collar-id and ear-tagging-based identifying procedures are examples of semi-permanent animal identification systems for individual animal identification. Individual cattle cannot be identified using ear-tagging-based label and verification procedures. Individual cattle may be readily tracked and monitored using integrated unique tag numbers. Animal verification and identification are difficult with ear-tagging-based procedures. The following are the significant issues with ear-tagging-based techniques: (1) integrated labels in ear-tags can quickly vanish; (2) distinct integrated labels can ultimately be destroyed; and (3) the ear can progressively be corrupted due to long-term use.

Sketch-based patterning techniques are used to create designs and patterns of various colors on the body of cattle. Individual drawing abilities are required for sketch patterning and design-based methods. The absence of standardization in the patterning process is a severe issue. The traditional animal identification procedures demand a more significant expense for cattle ear tagging due to the vast number of human resource requirements. Due to duplication, fraud, and forged embedded standard ear-tags, the venerability of ear-tagging-based systems decreases. Individual cattle are also identified and tracked using integrated RFID tags. However, the deployment and maintenance of RFID protocol, RFID chips, and scanners at various checkpoints have been recognized as one of the most challenging challenges for monitoring.

1.1 Problem Statement

As the number of farms is decreasing in India, but the numbers of cattle on each farm increasing every year, their monitoring is becoming more and more difficult. Classical monitoring frameworks which are currently used like ear-tagging and freeze branding provide manual procedures or methods which are also prone to human error.

An animal biometric based identification and verification smart system can be installed in farms providing a large scale monitoring system with reduced cost and better security. For such a system we need a reliable set of hardware and software that can work in an unconstrained farm environment. With the boom in the technology sector, the hardware can be easily arranged and can be set up. The problem is that we need a software framework which will take the image data, process it and provide accurate identification results in at least near-real time. Many authors have provided methods for the identification but none of them are good enough to be used in real life. Some of them are too slow to be used in real environment, some are too much constrained in terms of they need proper lighting, illumination or the amount of resources. Many of the authors are providing deep learning based methods in which there is no way we can add or remove a new class without training the whole model again which is not feasible.

1.2 Scope of work

In this study the authors are proposing an approach for the cattle identification. The cattle muzzle dataset used for the research cannot be provided publicly but the methodology of the work done is explained in detail and all the implementation hyperparameter details are also provided. The approach is compared with the state-of-the-art in the literature, and on the basis on the results achieved the superiority of the proposed approach is discussed.

1.3 Objective

Here the primary objective is to improve current methods used for identification of individual cattle. We want to propose a machine learning based approach where on the input cattle image is taken, and it will predict to which class it belongs or return the ID of he cattle. Our aim here is to build a cattle identification system which also support addition or removal of a cattle with minimal or no affect to the identification performance. It is also desired that the identification should give results in at least near-real time and the environment is not constrained in any way when the cattle image is being captured.

Chapter 2

Literature Survey

Here we will discuss what state-of-the-art methods and techniques are being used for the cattle identification task in the literature. We will also discuss literature which includes methods that we have used in our methodology.

Table 2.1: Comparative summary of the related literature

Paper	**Journal**	**Year of publication**	**Dataset**	**Model Used**	**Accuracy**
Automated Muzzle Detection and Biometric Identification [1]	Agronomy MDPI	2021	2900 cattle face images of 300 individual cattle (publicly-available)	YOLOv3-ResNet50	detection 99.13% Identification 99.11%

Deep Learning Framework for Recognition of Cattle using Muzzle Point Image Pattern [2]	International Measurement Confederation (IMEKO)	2018	395 cattle where each cattle has 10 images. Total 5000 muzzle point images	Deep Belief Network (DBN)	Identification 98.99%
Fingerprint Indexing and Matching: An Integrated Approach [3]	International Joint Conference on Biometrics (IJCB)	2017	2,000 fingerprints from 2,000 different fingers	ConvNet based fingerprint indexing algorithm	Identification 97.8%
Distinctive Image Features from Scale-Invariant Keypoints [4]	International journal of computer vision	2004	Two database's of 32 images	112 images	SIFT
Yolov3: An incremental improvement [6]	Computer Vision and Pattern Recognition (CVPR)	2018	MS COCO	YOLOV3	65.7% mAP at speed of 65 FPS

A Computer Vision Pipeline that Uses Thermal and RGB Images for the Recognition of Holstein Cattle [7]	Computer Analysis of Images and Patterns (CAIP)	2019	2257 images of 136 cattle	Alexnet	Identification 97.54%
Dog Identification Method Based on Muzzle Pattern Image [8]	Multidisciplinary Digital Publishing Institute (MDPI)	2020	55 muzzle images of 11 dogs	CLAHE + ORB + RANSAC + DMR	Equal Error Rate (EER) of 0.35%

Shojaeipour [1] has done multiple contribution in their paper. Firstly they are providing dataset which is available publicly. The dataset consists of 300 cattle, where face images is captured. Secondly, they are using YOLOv3 object localization model to first crop out the muzzle out of the face image. Then in the second step they used ResNet50 classification model that extracts the features using convolution layers and does classification. These two models combined together gives a biometric identification system for cattle. These models are trained using the transfer learning concept using few-shot learning where they are using 5 images only per cattle. For the muzzle detection they achieved 99.13% testing accuracy, and for identification task they achieved 99.11% accuracy. Their main contribution is that they are providing a large cattle muzzle dataset publicly. Many author's have worked on it, but this is not only the largest one but the only dataset we found which is on public access.

Santosh Kumar [2] are also using deep learning approaches for the identification of cattle using features extracted from muzzle (nose) of each cattle. They worked on the largest cattle muzzle dataset of 500 cattle as far as we know. But the dataset is not publicly available. They first tested the dataset using traditional feature extraction techniques like Local Binary Pattern

(LBP), Circular-LBP, Linear Discriminant Analysis (LDA) with One-Shot Similarity (OSS) technique and many others which didn't achieved noticeable results. Then they applied convolutional based techniques - deep belief neural network for feature extraction and classification, stacked denoising auto-encoder for encoding the extracted features. As expected these models gave the best result - identification accuracy of 98.99% was achieved.

Kai Cao [3] proposes a fingerprint indexing algorithm for human fingerprints. Actually, the issue with the convolutional model classification approach is that we cannot add or remove an existing class without retraining whole model. So, these indexing algorithms can be used to overcome this issue. They are using NIST SD4 and NIST SD14 fingerprinting dataset for the training and testing. Then they merged their proposed indexing algorithm with the popular COTS SDK algorithm to achieve identification accuracy of 97.8% at the penetration rate of 1%. This result means that it gives a 100-times reduction the search space. Our motive is also to use such an fingerprint indexing approach and apply it for muzzle-print indexing. This can significantly reduce the the search time over the dataset.

David G. Lowe [4] proposes the algorithm SIFT where they are extracting unique invariant features from images, and these extracted features can be used to represent object characteristics. These features are invariant to the scale that is the object size change will not change the features and similarly these features are also invariant to rotation and illumination. These features can be reliably used for different purposes like object detection in different photos or different frames of a video, images taken of an object from different angles or in different lighting, images with some noise in them etc. SIFT is one of the well-known algorithm used for verification purpose. Our motive is to try this algorithm for the one-to-one verification of two muzzle images.

For many of the complex problems, nearest neighbor matching algorithm is used. The algorithm takes quadratic time complexity for running, which makes it hard to be used for applications like matching in high-dimensional spaces. Marius Muja [5] proposed approximate algorithms for nearest-neighbor search. These algorithms at the loss of a little accuracy, gives huge speedup as fast as making it run in around linear run-time complexity. They are actually providing a library publicly which analyses the input and accordingly chooses the fastest algorithm. The parameters for the selected algorithm are also automatically decided. According to them, there are two main algorithms that works best for most of the input. One is using multiple randomized k-d trees and the other algorithm applies priority search on hierarchical k-means trees. This library is popularly known as FLANN which stands for 'fast library for approximate nearest neighbour'. Here, we are interested in using this algorithm for matching the features extracted from muzzle images.

Joseph Redmon [6] is one of the well-known machine learning practitioner. In this paper he is proposing YOLOv3 model which introduces many improvements over YOLOv2 model. There are many other models like R-CNN family which gives better accuracy than YOLov3 but in terms of speed YOLOv3 is just out of the league as it gives fastest results with competitive accuracy. The YOLOv3 model consists of two parts - first is the classification part darknet53, which is trained on Imagenet dataset. After that the model architecture is modified and again trained on COCO and Pascal VOC object detection dataset so that it can learn how to localize the bounding box around the object. Final evaluation is done on COCO dataset where it achieves 28.2 mAP to 57.9 mAP running in 22 ms to 51 ms at the resolution of 320×320 and 608×608 respectively. Here, we want to use this model and using transfer learning we want to re-train the model for muzzle localization purpose.

Amey Bhole in their paper [7] are providing holstein cattle dataset of 136 cattle publicly. The dataset contains RGB image and thermal image of side view of the cattle, but the images are taken from outside the milk station, so some horizontal and vertical bars are also getting captured along with the cattle. To overcome this, they are using the thermal image to segment out these bars region first. This segmentation creates empty blank gab for which they are using an inpainting algorithm which approximates the region that was supposed to be behind the bars. After this they are training Alexnet model using transfer learning. They achieved 97.54% identification accuracy with five-fold cross-validation. Instead of muzzle they are using the pattern over their body for the cattle identification. This pattern is also one of the distinguishing feature but for only some of the breed like holstein. This feature cannot be used as a reliable distinguishing feature.

Dong-Hwa Jang [8] proposed a dog identification system where they are using the muzzle region of the nose of the dog as the distinguishing feature for identification. For this they have self captured a dog muzzle dataset of 11 dogs. First as preprocessing they are using CLAHE algorithm (Contrast Limited Adaptive Histogram Equalization) which enhances the beads and ridges contrast and improves the image quality. After this for feature extraction they are using SIFT, SURF, BRISK and ORB algorithm for feature extraction. After feature extraction, FLANN and hamming distance methods are used for matching features of multiple images. They achieved the best EER of 0.35% where according to them ORB algorithm gave the best results.

Chapter 3

The dataset

The dataset is provided from Indian Veterinary Research Institute (IVRI) located in Izatnagar, Bareilly in Uttar Pradesh state of India. It consists of cattle images of different breeds including Holstein Friesian cattle, Jersey cattle, Red Sindhi cattle, Gyr cattle, Punganur cattle. This dataset is collected from different villages and farms of Karnataka, Tamil nadu, Telangana, Andra Pradesh states of India. The images in the dataset consists of different resolutions, different aspect ratio's with different illumination conditions. That's why these settings are being referred to as unconstrained environment, which means the dataset represents the actual real-world conditions. The only required constraint is that the image should be upright and the cattle features should be properly identifiable.

The dataset contains 395 classes of individual cattle's where each class contains 6 image views - front, full-left, full-right, left, right & muzzle as shown in the fig 1, 2, 3, 4, 5 and 6 respectively.

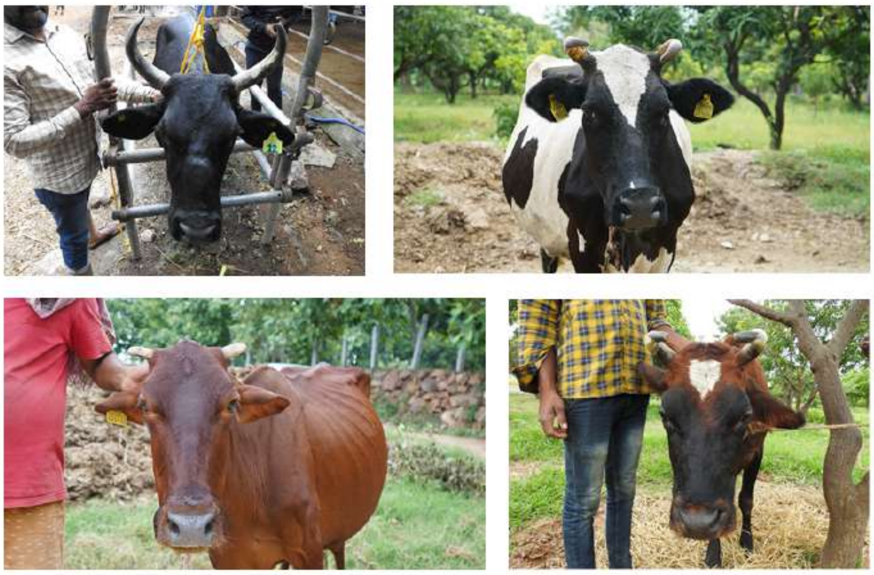


Figure 3.1: Front type images of dataset

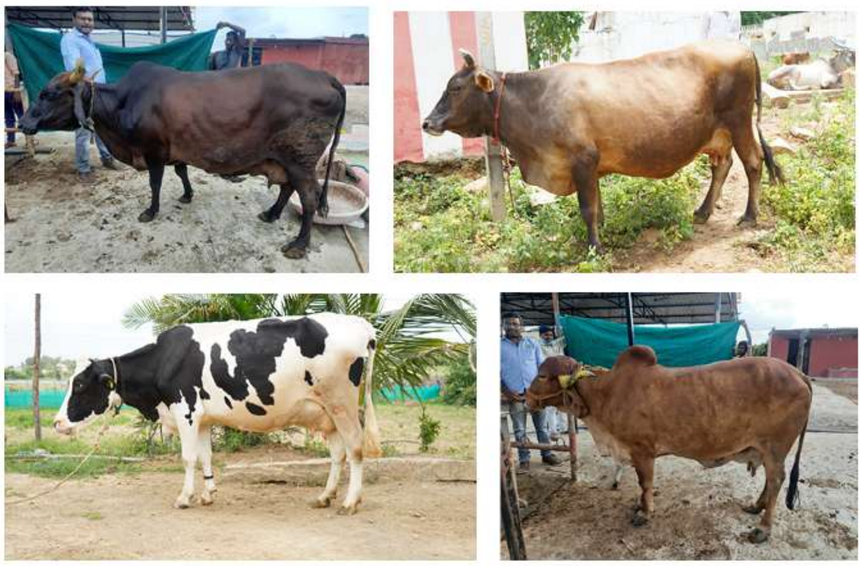


Figure 3.2: Full-left type images of dataset

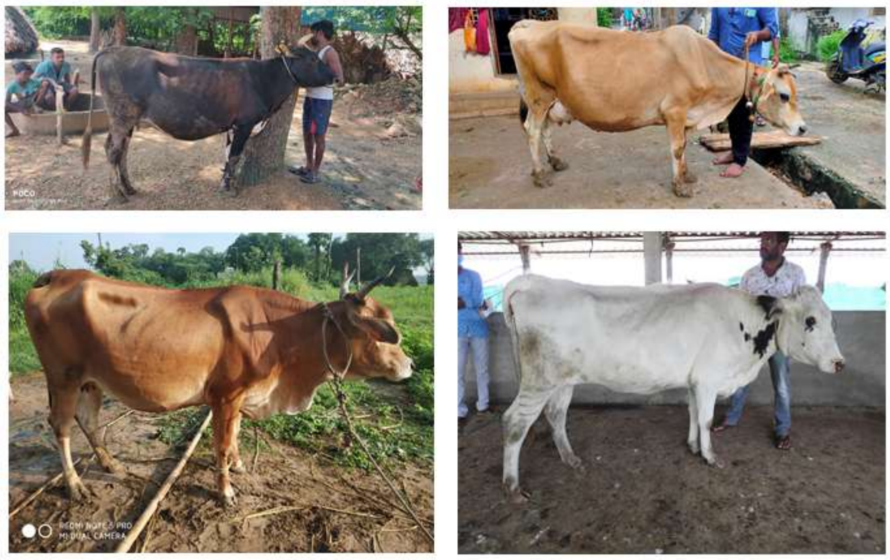


Figure 3.3: Full-right type images of dataset

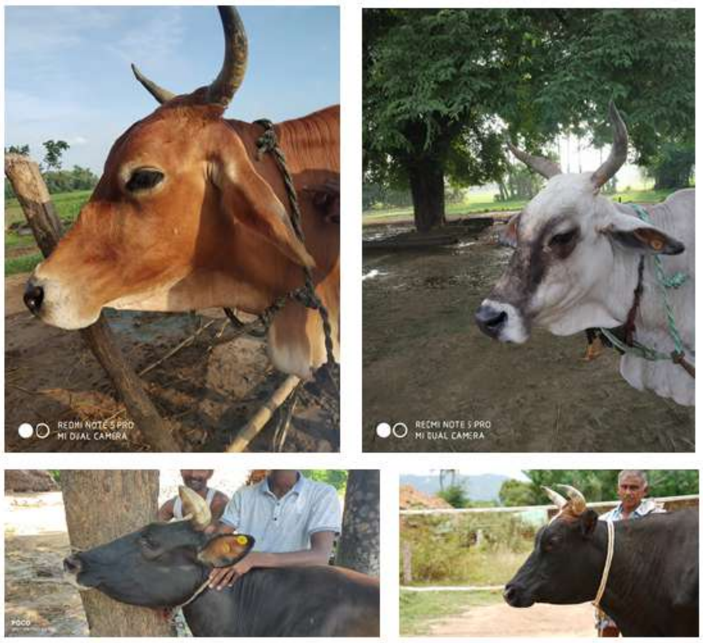


Figure 3.4: Left type images of dataset

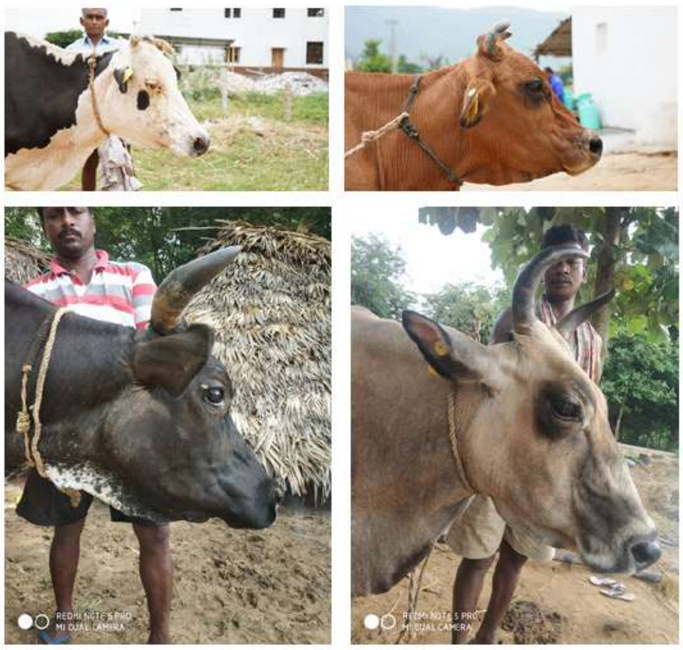


Figure 3.5: Right type images of dataset

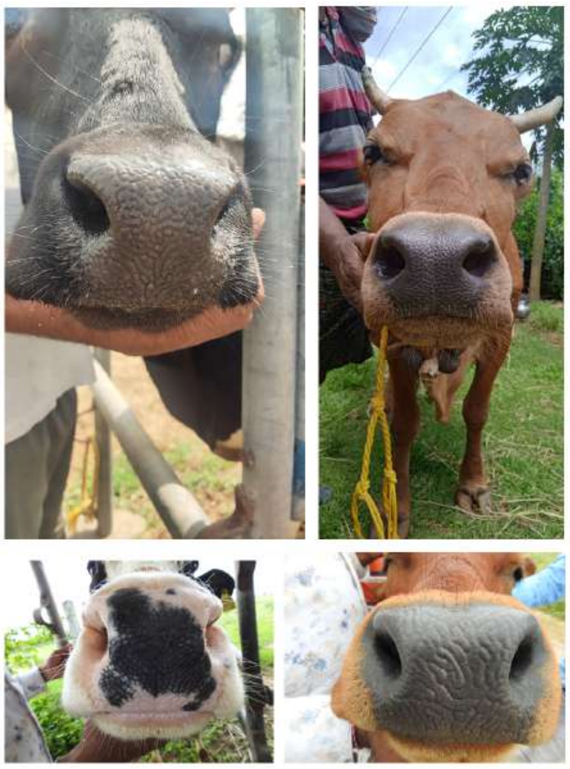


Figure 3.6: Muzzle type images of dataset

Chapter 4

Proposed methodology

In this section we will explain our proposed methodology in detail. The flowchart showing the steps for the identification approach is shown in the figure 4.1. In short, we take cattle image as input, localize the muzzle pattern, extract features from it and identify the cattle. First localization of muzzle pattern is done using YOLOv3, then an indexing technique is used that takes this muzzle pattern to filter out top-K classes from the cattle database and then one-to-one verification is done by feature extraction and matching using Scale Invariant Feature Transform (SIFT) algorithm.

This methodology can be used for both identification and registration of new cattle. That is, cattle database can also be updated in online mode, while maintaining the identification performance at the same time.

4.1 Localization

Object localization in an image refers to the task of predicting the rectangular bounding box around the object. There is very subtle difference between localization and detection, but detection mostly refers to the task of both localization and classification. Here, we are interested in localizing the muzzle pattern of the cattle and since there is only one class – muzzle, we are calling it muzzle localization. An example showing the localization is shown in the figure 4.2.

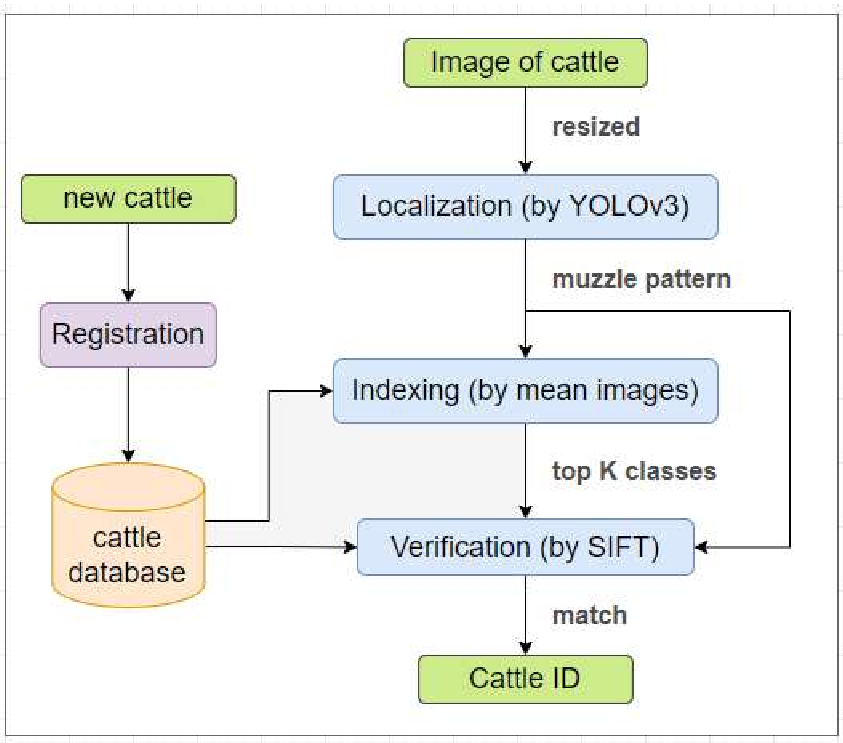


Figure 4.1: Flowchart for identification methodology

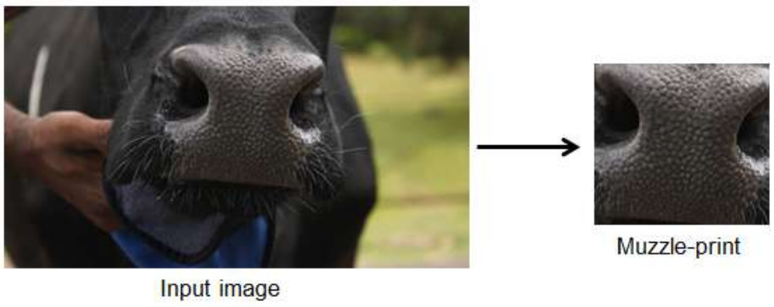
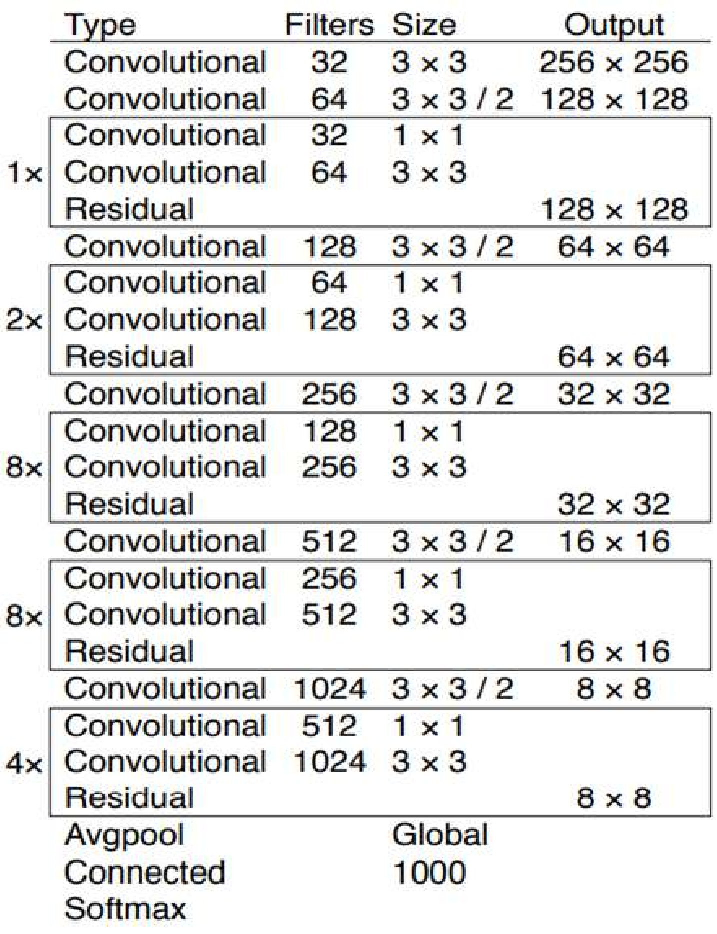


Figure 4.2: Localization of muzzle-print

4.2 YOLOv3 architecture & working

We picked YOLOv3 as it is the state-of-the-art object detector that gives best accuracy-complexity trade-off for our task. It detects and localizes the object in a single pass and that's why the name "You Only Look Once". We will now dive into the YOLOv3 architecture and working.

For feature extraction, they first trained Darknet-53 classification model having 53-convolutional layers. The model architecture for darknet is shown in the figure 4.3 which is taken from [6]. It contains successive 3 × 3 and 1 × 1 convolutional layers and residual shortcut connections, a technique they picked up from Inception and ResNet. They trained it on ImageNet 1000-class classification dataset and testing at 256×256 resolution achieved top-1 accuracy of 77.2% & top-5 accuracy of 93.8%.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4.3: Model architecture of darknet [6]

For object detection task, they modified darknet-53 by adding several convolutional layers including 3 yolo detection layers at different depths. This new fully convolutional model is again trained on COCO and Pascal VOC object detection dataset and is responsible for the bounding box prediction at different scales. These yolo-layers contain anchor boxes which are actually pre-defined bounding boxes that best describe scales and aspect-ratios of all objects of the dataset and then while training YOLOv3 model learned to make minor adjustments to these anchor boxes to localize the object perfectly. These anchor boxes are determined using K-means clustering algorithm. Different YOLO versions have different number of anchor boxes, YOLOv3 use 9 anchor boxes in total which are (373 × 326, 156 × 198, 116 × 90) for 1st yolo-layer, (59×119, 62×45, 30×61) for 2nd yolo-layer & (33×23, 16×30, 10×13) for last yolo-layer. As you can see from 1st to last, these yolo-layers learns to detect large, medium & small scale objects respectively.

Say, the dimension of the yolo layer is N×N, it is said to be divided into N×N grid cells. For 3 anchor boxes it predicts (N×N×3) boxes where each box is a (4+1+C) length vector, for the 4 bounding box offsets (x,y,w,h), 1 objectness prediction, and C class predictions. (x,y) adjusts the center and (w,h) adjusts the width and height of the anchor box to fit the object perfectly. Objectness score is the probability whether there is an object or not of any class.

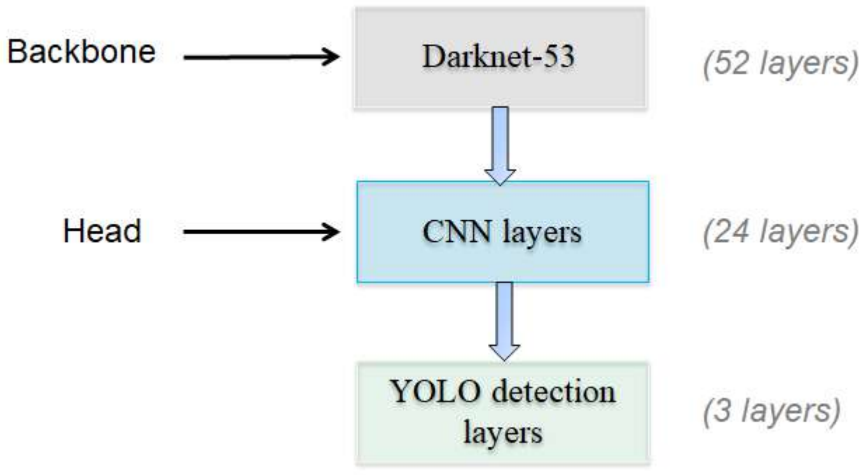


Figure 4.4: Architecture breakdown of YOLOv3

For the evaluation, YOLOv3 model is tested on COLO dataset. At 320 × 320 image resolution, YOLOv3 runs in 22 ms at 28.2 mAP. At 608 ×

608 image resolution, it achieves 57.9 mAP in 51 ms. YOLOv3 has been trained using multi-scale training, so on input it can take image of different resolutions, giving better results with better resolution, but also with increase in the time taken to produce result. There are some models like RetinaNet which are giving better mAP result as compared to YOLOv3 model but in terms of speed, YOLOv3 model is at least 3 times faster to every other object detection models. That's why this model is best suited to be used for real-time object detection and is already being used in many applications.

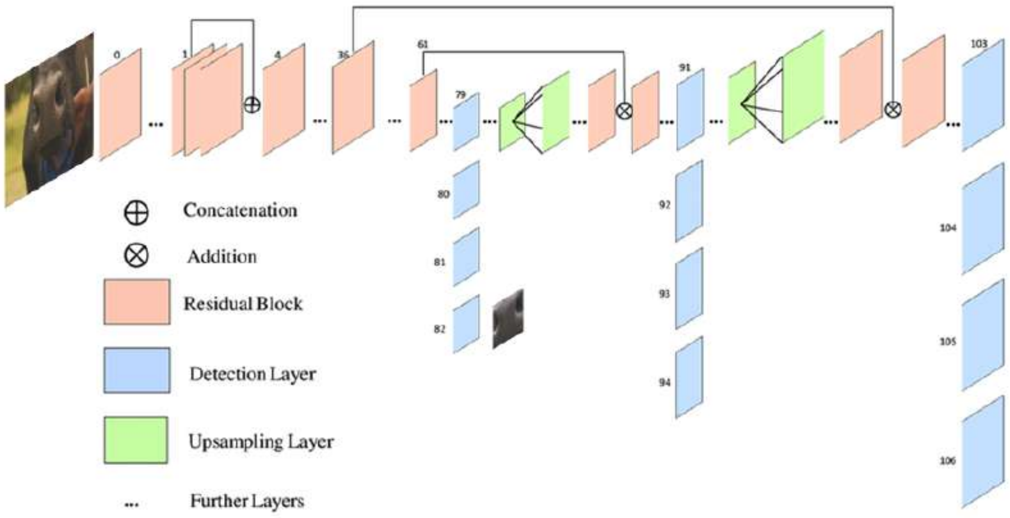


Figure 4.5: YOLOv3 architecture [6]

4.3 YOLOv3 fine-tuning

In this subsection we will see the training procedure for YOLOv3 based muzzle localizer. Firstly the architecture was modified; we removed all the layers after first yolo-layer, as we are not interested in detecting small scale objects. We need muzzle pattern detected to be of good enough size for identification purpose. We changed the number of classes in yolo-layer to 1 and the number of filters in the last convolutional layer just before yolo-layer to 18. Input image resolution was kept (416 $\times$ 416 $\times$ 3). Rest of the hyper-parameter are kept same as default which includes learning rate of 0.001, momentum of 0.9, decay of 0.0005 and batch-size of 64. Saturation, exposure & hue image augmentation is used. For ground truth bounding-box labeling of training dataset for YOLOv3 fine tuning we used LabelImg[1] tool. We manually labeled 340 images and trained on 300 images while 40 images were kept for validation. We trained for 1300 iterations till we achieved 100% mAP@.5 result on both training and validation set.

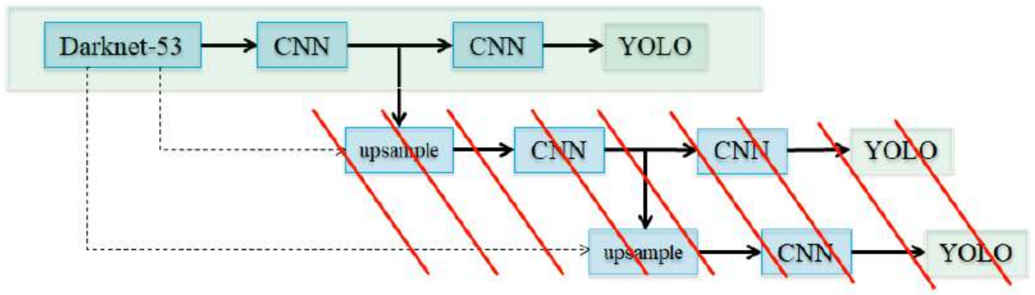


Figure 4.6: Modification in the architecture of YOLOv3

4.4 Indexing

Image indexing or image retrieval in computer vision refers to the problem of searching for a digital image in large image database and quickly filters out a small subset of best matching images. Indexing techniques are not supposed to find the exact match but to reduce the search space only. These approaches are mostly used for search and retrieval in large & dynamic databases where image data can be inserted, deleted or updated over time. Here, after localization of the muzzle pattern we added an indexing technique to filter out top-K classes to decrease our identification time. For each class in our database we keep an updated mean image of muzzle pattern and then at query time we compute distance of query muzzle-pattern image from each of these mean images and select K classes with minimum distance. For calculating mean image for a class, we first extract the muzzle pattern of each image of that class using our YOLOv3 muzzle localizer, resize these images to same width and height (say w $\times$ w), convert them from RGB format to grayscale format and then simply compute the pixel-wise mean of these images. In result we get one mean grayscale image of same width and height (w $\times$ w). These mean images for each class are also kept in the database. In case of any insertion of new image data, these mean image can also be easily updated. Now at query time, the cattle image is first passed to YOLOv3 muzzle localizer to extract its muzzle pattern. Then distance between this

[1]TzuTa, "LabelImg": https://github.com/tzutalin/labelImg, 2017

muzzle pattern and mean images of each class is computed. The distance between two grayscale images img1 and img2 of resolution (N×N pixel) is given by the following equation –

$$distance = \left[\sum_{i=1}^{N} \left(\sum_{j=1}^{N} (img1[i,j] - img2[i,j])^2 \right) \right] \tag{4.1}$$

4.5 Verification

Verification process takes two images as input, does one-to-one matching and tells whether they belong to same class or not. After indexing, top-k candidate matches are returned and by verification we want to find out to which class exactly the query cattle image belongs. For this task we are extracting feature points from both the images and then matching these points (called keypoints). For feature extraction we used SIFT algorithm. For matching we compare three algorithms, 1st - brute force matcher, 2nd - FLANN based matcher and 3rd - proposed rotation-variant matcher - AngMatcher.

4.6 SIFT algorithm

Scale Invariant Feature Transform (SIFT) is an algorithm that extracts features of an image and gives these features a coded representation that can be used for further tasks. Image matching is one of the application where SIFT can be employed. For matching two images, we first detect salient, stable points in both images. These points are called keypoints which represents features of an image. These features must not change with object position & pose, scale, illumination or minor noise & blur. Then we compute keypoint descriptors that uniquely characterize these keypoints. Then at last we determine correspondences between these keypoints, by matching these keypoint descriptors.

SIFT is an algorithm for detecting and describing feature points in an image. It extracts keypoints from the input image and compute its descriptors. This algorithm can be used in many different applications such as –

- Image matching
- Estimation of affine transformation/homography between images
- Structure from motion, tracking, motion segmentation

All these applications need to -

- detect salient, stable points in two or more images
- we need some features characterizing a salient point
- determine correspondences between them

These features must not change with

- Object position/pose
- Scale
- Illumination
- Minor image artifacts/noise/blur

Let's understand how SIFT algorithm woks in brief. There are mainly four steps involved in the SIFT algorithm –

1. **Keypoint detection**: *finding keypoints*

   By repeated down-scaling & Gaussian blurring multiple images are generated, then difference of each consecutive images is taken. Now in each of these difference images extrema points are searched and marked as keypoints.

   The "Scale space" for a sample muzzle-print image is shown below in figure 4.7. The picture is subsequently blurred using a Gaussian convolution as shown by yellow arrow. Gaussian convolution of the left neighbor is done with increasing standard deviation as we move towards right as shown by green arrow. The third-last picture of each row is downsampled as shown by blue arrow.

   After this the "scale space" is then normalized as shown in the figure 4.8.

   Then for each pair of horizontally adjacent pictures, we compute the differences of the individual pixels, a technique called difference of Gaussians (DoG). The discrete extrema of these difference images are calculated. The extrema we've found are marked in the figure 8 as shown below.

2. **Refining keypoints**: *removing low/high contrast keypoints*

   Low contrast and high contrast keypoints are removed. Low contrast means the extrema is too small and these mostly occur because of noise whereas high extrema keypoints mostly occurs at edges which do

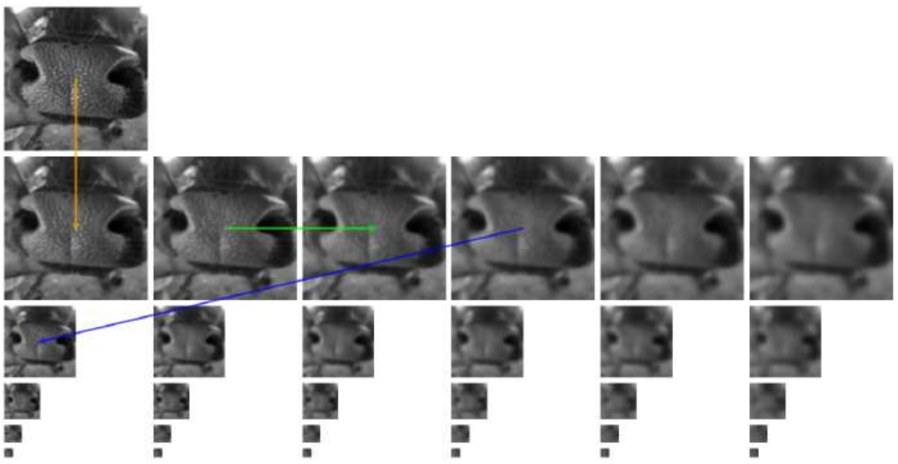


Figure 4.7: Scale space

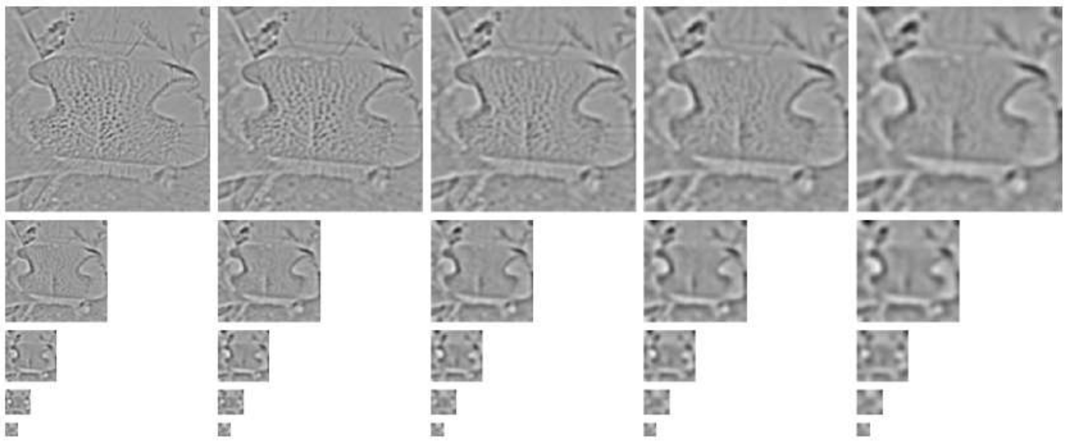


Figure 4.8: Normalized scale space

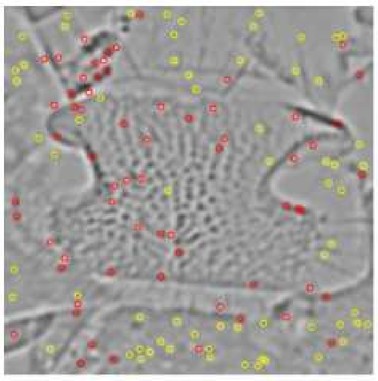


Figure 4.9: Discrete extrema points (initial keypoints)

not represent salient feature. We can interpolate for greater accuracy because the keypoint location and scale are discrete. A second-order Taylor-series is used to represent the DoG function in a tiny 3D neighbourhood around a keypoint. This allows for more precise extrema location, and if the intensity at these extrema falls below a threshold value (0.03 in the paper [5]), it is rejected. Low-contrast and high-contrast keypoints are removed.

Edges have a stronger response in DoG, so they must be removed as well. The primary curvature is computed using a 2x2 Hessian matrix (H). If the ratio exceeds a certain threshold (10 in the paper [5]), the keypoint is eliminated. This removes high-contrast edge keypoints. As a result, low-contrast keypoints and edge keypoints are removed, leaving just strong interest points. Two examples are shown in the figure 4.10 & 4.11 below.

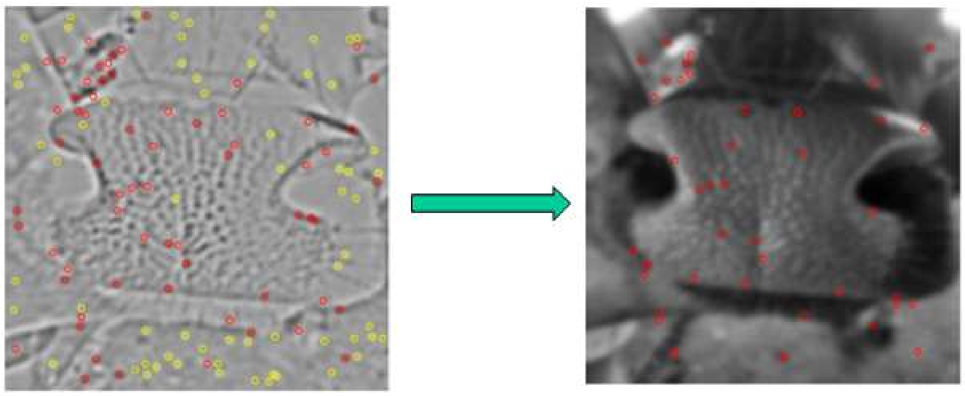


Figure 4.10: Refining keypoint location in muzzle-print

3. **Orientation assignment**: *computing orientation angle of each keypoint*

   For each keypoint, an orientation angle is computed by observing local gradient directions in the neighboring region of that keypoint.

   To achieve image rotation invariance, each keypoint now has an orientation assigned to it. Depending on the scale, a neighbourhood is drawn around the keypoint location, and the gradient magnitude and direction are determined in that area. An orientation histogram with 36 bins covering 360 degrees is constructed (it is weighted by gradient magnitude and has a gaussian-weighted circular window with a scale of 1.5 times the keypoint scale). To compute the orientation, the highest

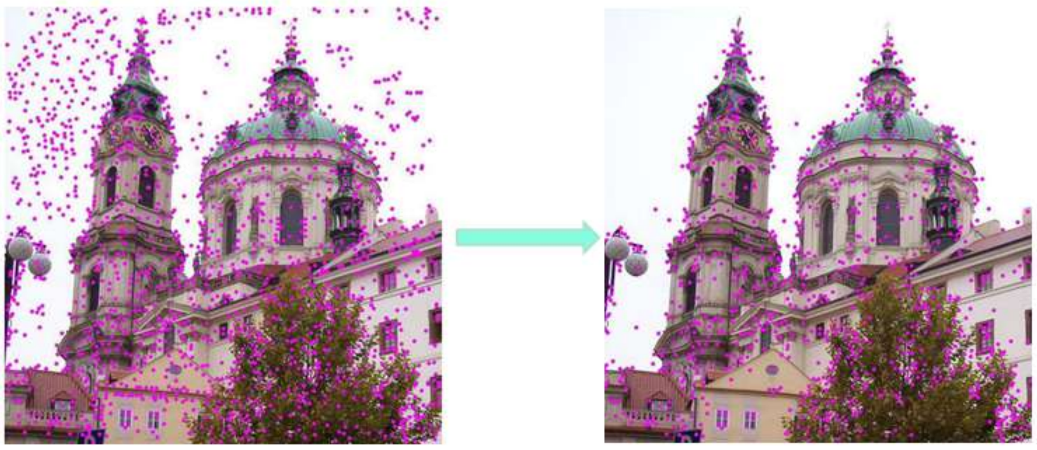


Figure 4.11: Refining keypoint location example

peak in the histogram is used, as well as any peak above 80% of it. It creates keypoints that are the same size and location, but face in different directions. It helps to keep the matching stable.

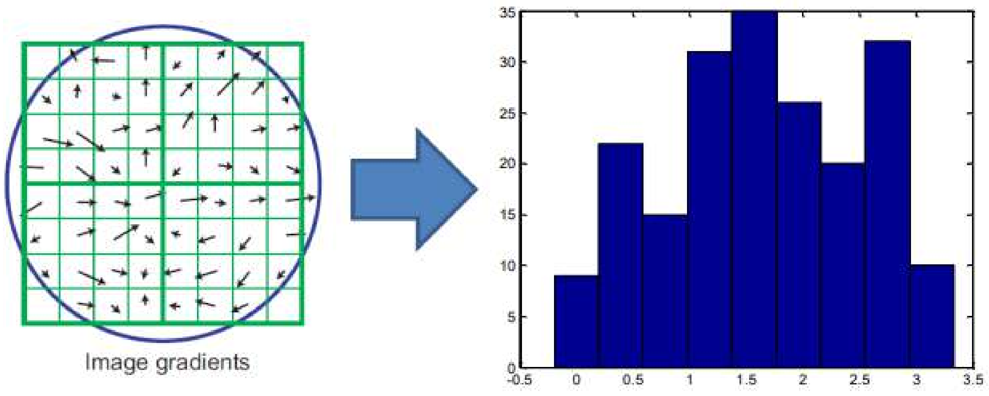


Figure 4.12: Assigning orientations

4. **Keypoint descriptor**: *computing descriptor of each keypoint*

   For each keypoint, first the coordinate system is rotated by its orientation angle (to make it scale invariant), then its surrounding region is subdivided into 4×4 grid, where for each cell 8-bin orientation histogram is made. This gives 4×4×8=128 length array which is resultant keypoint descriptor.

   A keypoint descriptor has now been generated. A 16x16 neighbourhood is drawn around the keypoint. It's subdivided into 16 4x4 sub-blocks.

An 8-bin orientation histogram is constructed for each sub-block. As a result, there are a total of 128 bin values available. To form a keypoint descriptor, it is represented as a vector. In addition, many methods are used to ensure robustness against variations in illumination, rotation, and other factors.

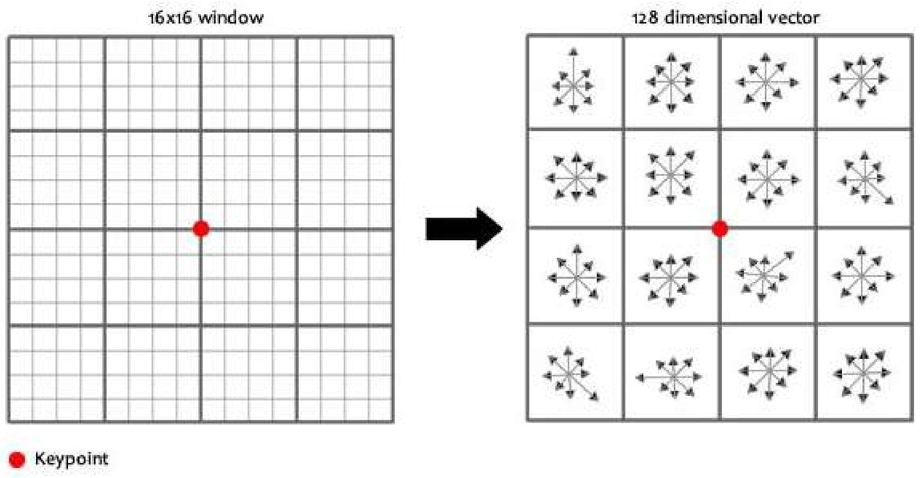


Figure 4.13: Keypoint Descriptor

4.7 Keypoint matching

SIFT algorithm takes one image as input and returns keypoints and keypoint-descripters which represent salient features of the image. We apply SIFT algorithm on both images, then we match keypoints of one image to another. For this matching we use our proposed matching algorithm and we also compare it with two other algorithms - brute force matcher & FLANN based matcher.

- Brute force matcher (BFmatcher)

  It is most basic type of matching. From each keypoint of first image, we calculate euclidean distance to every keypoint of second image and we say that two keypoints are same if distance between them is smaller than a threshold value. If the number of keypoint detected in both the images by SIFT is n & m respectively then BFmatcher takes O(n*m) time complexity.

- FLANN based matcher (FLANNmatcher)

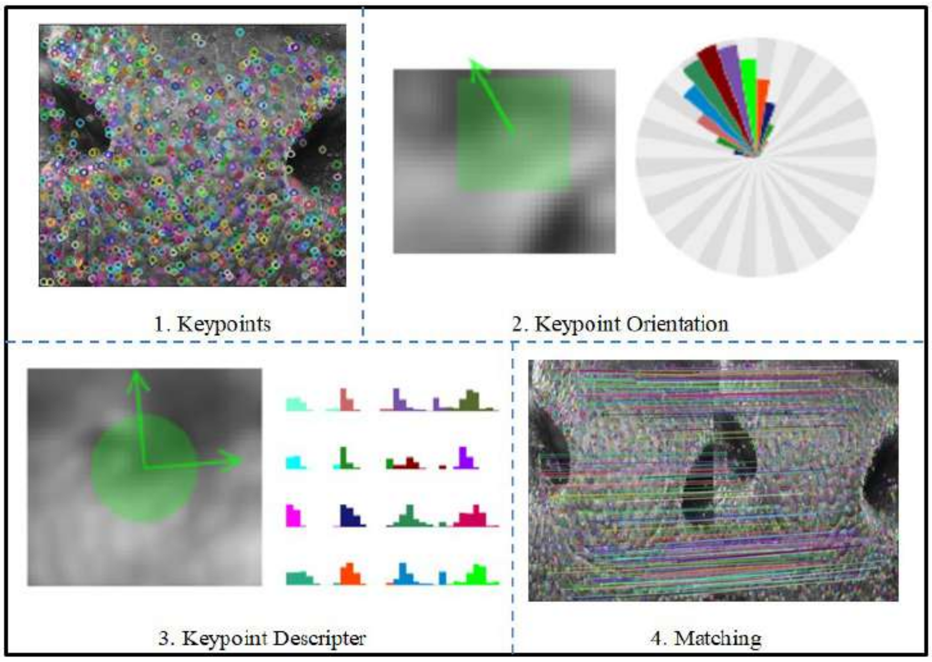


Figure 4.14: Verification using SIFT algorithm

FLANN stands for “Fast Library for Approximate Nearest Neighbors”. It automatically chooses the best algorithm and optimum parameters by analyzing its input. For each keypoint we find two nearest neighbors (i.e. with minimum distances) and use Lowe’s ratio test to filter good matches. Lowe’s ratio test checks that the two distances are sufficiently different, that is, if the ratio of the distance of the two nearest neighbors is smaller than a threshold value. FLANNmatcher gives faster and more robust keypoints as compared to BFmatcher in most of the applications.

- Proposed matcher (AngMatcher)

  We will be calling our proposed keypoint matcher as “AngMatcher” for simplicity. Feature detector & descriptors like SIFT gives rotation-invariant keypoints by aligning them with their orientation angle before computing descriptors. AngMatcher gives only $\pm A^{\circ}$ rotation-invariance reducing computations and time complexity. This is a sliding window based algorithm. Suppose we are matching keypoints of image1 and image2, we take keypoints in A° window from image1 and perform BFmatcher with keypoints in $3A^{\circ}$ window of image2.

Chapter 5

Workflow

In previous sections we explained and discussed our methodology like how our approach to the cattle identification is going to work as a whole. Here, we will discuss how an individual cattle will be processed by our pipeline. We will explain with an example like how a cattle is registered with our system and later when the user gives a cattle image in the input, how it is getting identified by our system.

Our model consists of two independent steps - firstly the cattle registration, where images of the new cattle is taken on the input and added to the cattle database making it ready for the identification purpose. The second part of the model is the online cattle identification where an image is given on input and it tries to identify which cattle class it belongs to. For the registration part we are calling it as 'registration pipeline' and the identification part, we are calling it as 'identification pipeline'. Let's look into these two parts in detail.

5.1 Cattle registration

Cattle registration is the first step where when a new cattle needs to be added its images are processed and added to the database. The registration procedure is described in the figure 5.1.

Registration of a new cattle is performed as follows -

- **Localize muzzle-prints**
  Firstly, atleast 8 images of the cattle face are taken as input in which the muzzle region is clear enough. These images are passes to our YOLOv3 localization model and the muzzle region is extracted. We expect the muzzle region extracted to be at least 500 pixel resolution. These extracted muzzle-prints are stored in the database, which are

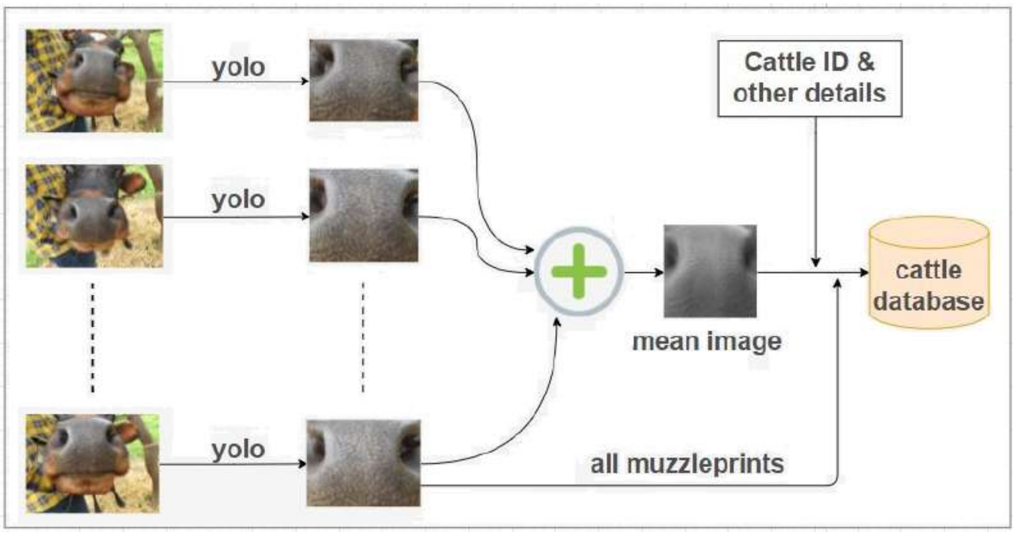


Figure 5.1: Cattle registration

required for the SIFT verification step. Please note the original images of full face taken are not being stored in the database, only the cropped muzzle part is being stored.

- **Mean image generation**
  To compute the mean picture for a class, we use our YOLOv3 muzzle localizer to extract the muzzle pattern from each face image of the cattle as explained in previous point. We resize these muzzle-prints to the same width and height (say w $\times$ w), convert them from RGB format to grayscale format, and then compute the pixel-wise mean of these muzzle-prints. As a consequence, we have one grayscale picture with the same width and height (w $\times$ w). Here in our study we have chosen the value of w to be 512 pixel. Below 500 pixel dimension the performance of the indexing was decreasing so the value of 512 is chosen. This mean image is also stored in the database for our new class. This mean-image is required for the indexing step of our identification pipeline.

5.2 Cattle identification

Cattle identification is the step where an image of the cattle face is provided in the input. This image is processed by the identification pipeline as shown by the figure 5.2 where it shows how the input probe image is processed by

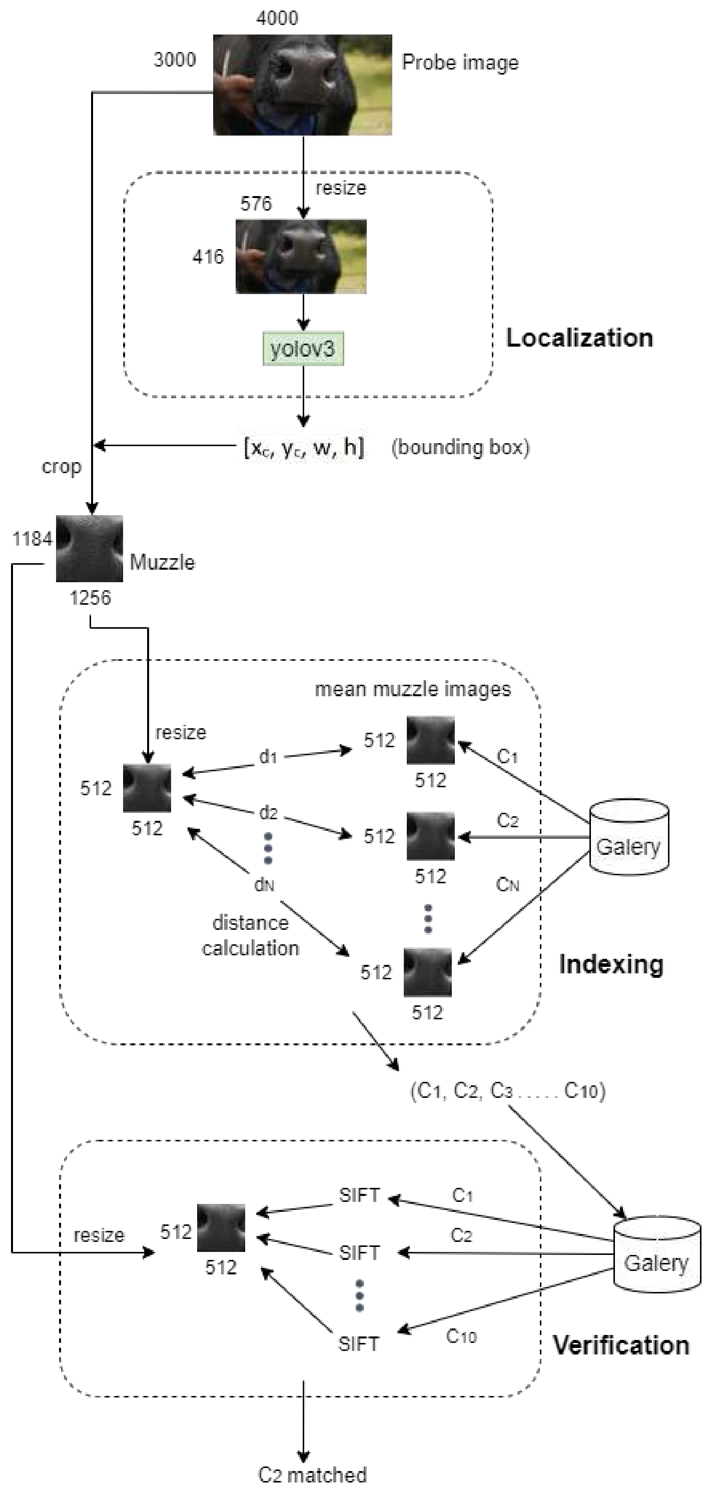


Figure 5.2: Flowchart for the identification of an image

the pipeline.

Steps of the identification pipeline are described as follows with the help of an example -

- **Localization of muzzle**
  As shown in the figure 5.2, input probe image of 3000×4000 is taken. Here, the input image for the identification pipeline is being referred to as probe image.

  The localization of the muzzle from the probe image is done by our re-trained YOLOv3 localizer. For YOLOv3, the probe image is resized to keep the smaller side to be of dimension 416 while maintaining the aspect ratio of the image. This resized image is passed to YOLOv3, which outputs the bounding box values for the muzzle region in that image. In case the YOLOv3 is not able to localize the muzzle then the pipeline ends here and 'not found' result is shown as result. The bounding box includes x and y coordinate of the center and the width & height of muzzle, and these values are in ratio with the original width & height so they range between 0 & 1.

  Using the bounding box values the muzzle region is cropped out from the original image only. So, as in example, muzzle-print of (1184×1256) dimension is cropped out of original probe image of (3000×4000) dimension. This muzzle-print is is then goes on to the next step for indexing.

- **Mean muzzle indexing**
  Now, the mean muzzle image of each individual cattle is fetched from the gallery and the distance of our muzzle-print is calculated with each of these mean muzzle image. Then we select K classes with minimum distance where the value of K is taken as 10 here. The working of indexing is explained in detail in the methodology section. The actual label is expected to be in one of these K classes. These K class labels are then passed to the next step verification.

- **Verification**
  After indexing K class labels filtered and it is expected the desired cattle is among one them. All the muzzle print images of these K classes is fetched from the gallery and one-to-one verification is performed with the muzzle-print of probe image.

  The verification is done by first extracting SIFT features of two images and then performing keypoint matching using our proposed AngMatcher. AngMatcher algorithm returns the number of keypoint matched and using a constant threshold it is decided whether it is a match or not. From each individual class 7 muzzle-prints are fetched and matched with the muzzle of probe image. If for a class 3 or more images are matched then the final decision is made that the probe image belong to this class. If from no class 3 or more images are matched then 'no match' result is shown.

Chapter 6

Experimental results and analysis

In this section, the evaluation metrics are discussed first which are used for the performance evaluation. The performance results are discussed for each step of the identification pipeline. Then the evaluation of the whole pipeline is discussed. At the end the runtime analysis is also performed for the proposed approach in this study.

6.1 Defining metrics

Following some of the evaluation metrics are explained which are used for reporting results in this study.

- **Confusion matrix**
  Confusion matrix is one of the most popular metric used for the easy visualization and analysis of the performance of a binary classification. Confusion matrix is just a 2×2 matrix showing the count of whether a positive class or a negative class is correctly predicted or not.

  The terminologies related to confusion matrix are as follows -

  1. **true positive (TP)**
     The number of positive class instances correctly predicted.
  2. **true negative (TN)**
     The number of negative class instances correctly predicted.
  3. **false positive (FP)**
     The number of negative class instances wrongly predicted.

	positive (0)	**negative (1)**
positive (0)	TP	FP
negative (1)	FN	TN

Table 6.1: Confusion matrix

4. **false negative (FN)**
   The number of positive class instances wrongly predicted.

There are many metrics that are derived using the confusion metrics like accuracy, recall, precision etc.

- **Accuracy**
  Accuracy is the most common and popular metric used. It answers the question - "Overall, how often is the classifier correct?". It tells how many instances are correctly predicted.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

- **Recall**
  Recall answers to the question - "When it's actually yes, how often does it predict yes?". It tells out of all the positives how many are correctly predicted. It is given by the following formula -

$$recall = \frac{TP}{TP + FN} \tag{6.2}$$

6.2 Performance evaluation

In this section the the experimental results are discussed. Testing of each operation is done on the original dataset only, not on the output of previous step in the methodology pipeline. Thus it gives an idea of how accurate the algorithm is, not how it's affecting the whole pipeline. The performance

	Accuracy
YOLOV3 muzzle localization	99.8% (@.5IOU)
Mean muzzle indexing (k = 10)	85.4%
SIFT verification	82.0%
Combined identification pipeline	79.8%

Table 6.2: Experimental results for different steps of identification method

evaluation is shown in the table 6.3, where the accuracy metric is used for reporting results.

Lets look into the details of how the train and test is performed. Firstly for localization all the muzzle-type images of the dataset are passed to YOLOv3 model and percentage of bounding box for muzzle-print correctly predicted are being reported as the accuracy. Here, all the muzzle-prints are also saved as for indexing and verification these localized cropped muzzle-prints are used directly. For the indexing, first 8 images are used for the mean muzzle image generation and the 9th image is used for the testing. One image from each of the 395 classes is used for testing. If the top-K classes returned from indexing includes the actual class label, then it is masked as the true prediction, else its taken as incorrect prediction. Accordingly accuracy is determined.

For verification using SIFT, again the localized muzzle-prints using the YOLOv3 model is considered. Similar to indexing the 9th muzzle-print of each class is taken and matched with the first 8 muzzle-prints of each of the 395 classes. Out of 8, if 3 or more matches are found then it is predicted that the probe image belong to this class. But it is possible that multiple classes are matched with it, in this case we are marking it as incorrect prediction. Only if there is a single match which should also be the actual class then it is marked for true prediction. Accordingly we calculate accuracy metric from it.

For the evaluation of combined pipeline, one image from each the 395 classes is taken, which is not used for generating the mean muzzle image. Each of these test image is passed through YOLOv3 localizer followed by indexing and then followed by SIFT verification. Basically each test image is passed through whole pipeline and if the result predicted is actual label then its marked as true prediction and like this the accuracy is computed.

6.3 Runtime analysis

We covered the experimental results showing the performance based on accurate the approach is, but for the system to work in real life, run-time of the algorithm is also a deciding factor. That's why we are also reporting the time taken by the algorithms.

	Approx. runtime (second)
YOLOv3 muzzle localization	2 s
Mean muzzle indexing (k = 10)	0.3 s
SIFT verification	0.4 s
Combined identification pipeline	5 s

Table 6.3: Approximate runtime for different steps of identification system

The results are being tested on Intel core i3-5005U @ 2.00GHz processor with 8 GB RAM. The first step of localization takes around 2 seconds, mean muzzle indexing takes around 0.3 seconds and verification of probe muzzle with 1 image takes around 0.5 seconds only. For verification step, the probe muzzle is matched with all the 8 images of top-k classes returned from indexing, where K is kept 10 here.

For the calculation of average runtime of the combined identification pipeline, the probe image passes through YOLOv3 localizer followed by indexing which returns top K cattle labels. In most of the cases the top class label returned is the desired class only so if the first 3 SIFT matching are successful, the identification process ends the matching class is returned. Following is the best case runtime, for which the total time taken is calculated as follows -

$$2 + 0.3 + (0.4 * 1 * 3) = 3.5s \tag{6.3}$$

whereas if for some reason the there is no match, then the pipeline gives worst case time that can reach upto 34 seconds. It is calculated as follows -

$$2 + 0.3 + (0.4 * 8 * 10) = 34.3s \tag{6.4}$$

When the cattle identification system is actually used in the any real application in some farm, mostly it will be used for the identification of the cattle which is already in the database. That's why to get a better idea of the processing speed we also tested our method for 395 images, one unseen image from each class where each class is already registered in the system. Then for all these 395 identifications whether the result is correct or not, the mean of their runtime is taken. Unlike accuracy, the runtime depends on many factors like the system resources, the background processes, memory state so the runtime value actually differs each time. The average processing speed of a single identification is around 5 seconds.

Chapter 7

Conclusion

This study proposes a new approach for the cattle identification. It consists of mainly 3 steps - localization, indexing and verification. Here, we achieved an accuracy of 79.8% which is as compared to the literature a little less, but we are also providing analysis of the runtime. The runtime performance of our approach is really good. For 395 cattle classes our model gives a near-real time performance of approx 5 seconds. So that's why our proposed method can be used for actual application on farms. The method can also be extended and used for other livestock and pets.

Bibliography

[1] Shojaeipour, A., Falzon, G., Kwan, P., Hadavi, N., Cowley, F.C. and Paul, D. Automated Muzzle Detection and Biometric Identification via Few-Shot Deep Transfer Learning of Mixed Breed Cattle. Agronomy, 11(11), p.2365, 2021

[2] Santosh Kumar, Amit Pandey, K. Sai Ram Satwik, Sunil Kumar, Sanjay Kumar Singh, Amit Kumar Singh, and Anand Mohan. "Deep learning framework for recognition of cattle using muzzle point image pattern." Measurement, vol -116, pp 1-17, 2018

[3] Cao, K. and Jain, A.K., October. Fingerprint indexing and matching: An integrated approach. In 2017 IEEE International Joint Conference on Biometrics (IJCB), pp. 437-445, 2017

[4] Lowe, D.G. Distinctive image features from scale-invariant keypoints. International journal of computer vision, vol - 60(2), pp.91-110, 2004

[5] Muja, Marius. "Approximate nearest neighbors with automatic algorithm configuration." In Proc. Int. Conf. on Computer Vision Theory and Applications, Volume 1, Lisboa Portugal, February 5-8, 2009

[6] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." Computer Vision and Pattern Recognition, arXiv preprint arXiv:1804.02767, 2018

[7] A. Bhole, O. Falzon, M. Biehl, G. Azzopardi, "A Computer Vision Pipeline that Uses Thermal and RGB Images for the Recognition of Holstein Cattle", Computer Analysis of Images and Patterns (CAIP), pp. 108-119, Salerno Italy, 3-5 Sept 2019

[8] Jang, Dong-Hwa, Kyeong-Seok Kwon, Jung-Kon Kim, Ka-Young Yang, and Jong-Bok Kim. "Dog Identification Method Based on Muzzle Pattern Image." Applied Sciences 10, no. 24, 8994, 2020.

[9] Kumar, Santosh, and Sanjay Kumar Singh. "Cattle recognition: A new frontier in visual animal biometrics research." Proceedings of the national academy of sciences, India section A: physical sciences 90, no. 4, pp.689-708, 2020

Ouriginal

Document Information

Analyzed document	Rohan_Thesis_Report_3.pdf (D137621586)
Submitted	2022-05-23T12:13:00.0000000
Submitted by	Central Library
Submitter email	centrallibrary@nitdelhi.ac.in
Similarity	2%
Analysis address	centrallibrary.nitdel@analysis.urkund.com

Sources included in the report

W	URL: https://arxiv.org/pdf/2006.09205 Fetched: 2021-01-24T12:12:21.9930000	2
W	URL: https://www.mdpi.com/2073-4395/11/11/2365/pdf Fetched: 2022-05-23T12:14:32.9430000	2
W	URL: https://isiarticles.com/bundles/Article/pre/pdf/127120.pdf Fetched: 2022-05-23T12:15:02.8530000	1
W	URL: https://www.mdpi.com/2079-9292/11/5/739/pdf?version=1646124113 Fetched: 2022-05-23T12:14:46.9130000	1
W	URL: https://www.mdpi.com/2076-3417/9/22/4914/htm Fetched: 2020-03-20T10:18:50.2100000	1
W	URL: https://link.springer.com/chapter/10.1007/978-981-10-7956-6_4 Fetched: 2021-06-17T18:21:17.1030000	1