Automatic ski-jump distance measurement with convolutional neural networks and computer vision

Matjaž Kukar[1], **David Nabergoj**[1]

[1] *University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenija*
E-mail: matjaz.kukar@fri.uni-lj.si, david.nabergoj@student.uni-lj.si

Abstract. *In ski jumping, video-assisted distance measuring is used only at the top-level competitions (world cup, continental cup). For smaller competitions it is prohibitively expensive and for this purpose we have developed a cost-effective system using commercially available equipment (a surveillance camera and an ordinary laptop computer). We further support distance-measuring umpires by introducing fully automatic distance measuring based on convolutional neural networks and computer vision techniques. We test our system on smaller ski jumping hills and show that while the system cannot completely replace the human operator, it can significantly speed-up the distance measuring process. Preliminary experiment on large hills confirm our experiences and will require only minor modifications for using multiple cameras.*

1 Introduction

In Slovenia, there has been a significant increase in ski jumping popularity in recent years, a consequence of excellent competitive results. At the primary level clubs reported doubled numbers of youngsters. This increased the burden on ski jumping coaches, organizers and professional staff in competitions. In our previous work [1, 2] we focused primarily on supporting distance umpires who have a demanding, exposed role, and their mistakes can significantly influence the competition outcomes. Our aim was to develop a system for supporting video-assisted distance measuring on smaller hills with accessible hardware requirements (a single-camera video system and a laptop).

In this paper we upgrade the system using convolutional neural networks and computer vision techniques in order to provide automatic distance measurements with reasonable accurracy. We evaluate the automatic measurements in ski jumping competitions on small hills in regional competitions (Cockta Cup) with respect to official results and show some directions for future development and use on larger hills.

2 Methods and materials

Ski jump distance is defined as a distance between the edge of the jumping ramp and a point where both ski jumper's skis have touched the ground with full surface [3, article 432.1]. The middle point between both legs is used when the legs are apart (e.g., Telemark landing style). FIS requires a jump distance accuracy of 0.5 m. There are several special cases [4] and even on the smallest competition hills it is difficult to manually measure the exact jump distance, as landing speeds exceed 10 m/s, and the angles between the landing slope and landing trajectories of ski jumpers are often minute [5]. Existing video distance measuring systems (Swiss Timing, Ewoxx) are basically video recorders and provide no automatic aid for umpires.

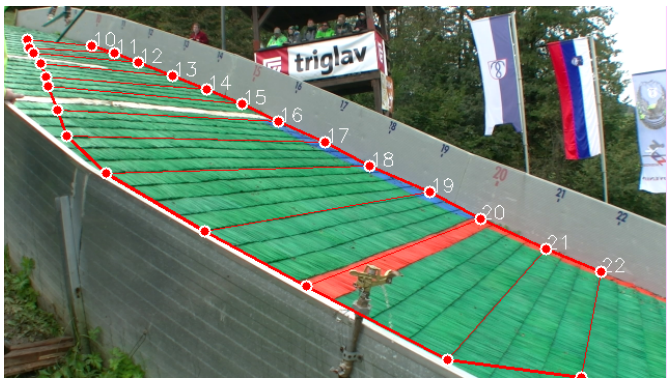

Figure 1: The measurement grid. Each measurement line is precisely calibrated to the particular hill and corresponds to a distance (in meters).

2.1 Automatic distance measurement system

The automatic distance measurement system analyses a sequence of frames where the ski jumper is visible. The ski jumper's location is determined for each frame, the frames without ski jumper are ignored (i.e. before of after the jump). The landing frame is the first frame in the sequence where the jumper has already landed. A measurement grid is overlaid on top of each frame. It contains lines annotated with distances (in meters) used to estimate the distance for each pixel in the frame. The main steps in the system are: determining the landing frame, finding the jumper's feet (a pixel) in this frame, and determining the distance based on the feet point and the measurement grid. The measurement grid is shown in figure 1. The distance measurement system is part of the full system, which helps the distance umpire.

2.2 Motion detection

The input to the distance measurement system is a sequence of frames where the jumper is visible. However, the jumper is only visible when jumping and near the landing area. Anything that occurs between jumps is not relevant to the system. We must first extract the relevant subsequence from a typically longer sequence of frames. We do so using an algorithm that receives a sequence of RGB images as input and outputs a vector $\mathbf{v}$ of equal length that tells us when a jumper is visible. The i-th element of $\mathbf{v}_i$ is equal to 1 if the jumper is visible in the i-th frame, and 0 otherwise. The algorithm works by using background subtraction [6] to determine the frames where motion occurs. A binary mask is generated for each frame, the white pixels in the mask correspond to the area with motion. The largest contour in the mask typically corresponds to the jumper. If it is not large enough (a parameter, dependent on

the camera distance), we conclude that the jumper is not in the frame. We use a median filter to remove some noise from this contour and finally extract the smaller image of the jumper from the axis-aligned bounding box of the processed contour [7]. Visualization of the process is shown in figure 2.

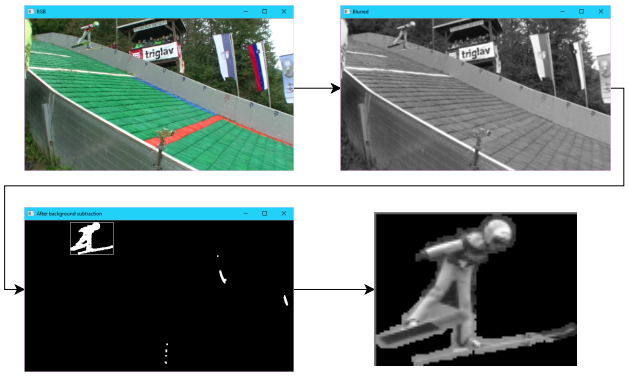

Figure 2: The jumper localization process. The input to the algorithm is a RGB image. It is converted to grayscale and blurred using a Gaussian filter. A motion mask is obtained using background subtraction. The largest moving object in the mask corresponds to the jumper. Finally, the jumper cutout is returned.

After the jump we get a sequence of boolean values, corresponding to the jumper being in the frame. The jump is therefore a subsequence of frames where the corresponding boolean value is true, meaning that the jumper is visible. There might be some frames in between the jump where the jumper is not detected. We allow some such frames in the jump, otherwise the complete jump might not be correctly extracted. We utilize a threshold $t = 2$, which allows for at most t consecutive frames without the detected ski jumper in the jump sequence.

2.3 Landing detection

The next part of the measurement pipeline is detecting the landing when it occurs. We use a convolutional neural network that is able to receive a frame of the jump segment as input. It activates the output 0 when the jumper in the frame is in the air and 1 when he is on the ground. 0 and 1 correspond to classes *Air* and *Ground*, respectively. We assign the landing to the first frame in the sequence classified as *Ground*. The architecture of the neural network is shown in figure 3.

2.4 Feet point calculation

Once we know the landing frame, we can process it using computer vision methods to find the precise jump distance. The procedure for this is two-fold: we first find the jumper's *feet point* and then use the measurement grid to calculate the actual distance. We find the feet point utilizing the mask generated in the motion detection phase. First we use a median filter afterwards to remove noise. The *feet point* is then calculated as the intersection of the line going through the jumper's body and the line through their skis (figure 4. We can to do this without a significant loss of accuracy because of the constrained position of the camera. The lines are found by using the RANSAC [8] procedure to fit two linear models on to motion mask. The models are selected if their lines are sufficiently vertical (corresponding to the line going through the body) or horizontal (corresponding to the line going through the skis). The use of RANSAC is justified since the motion mask contains many

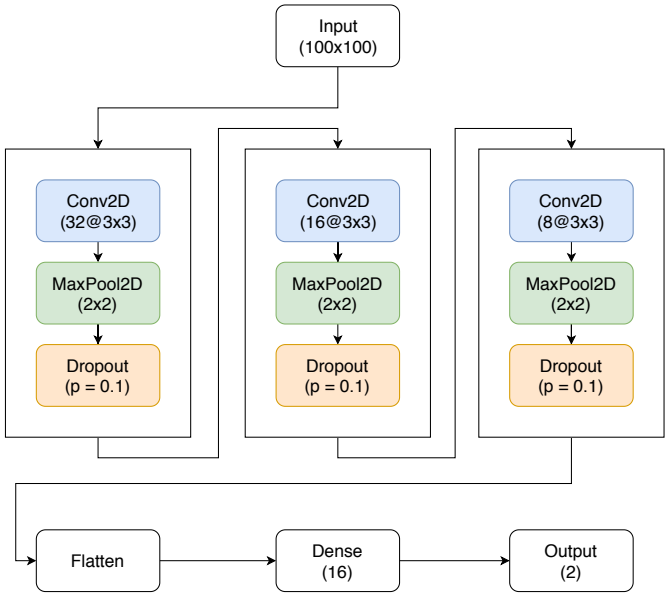

Figure 3: The CNN architecture. The input is a 100x100 grayscale image and the output is a vector with two elements whose values correspond to the probabilities of classifying the image as *Air* or *Ground*.

outlier pixels which are irrelevant to the lines (e.g., pixels corresponding to the jumper's hands are irrelevant to the line going through the body).

Figure 4: Feet point approximation. The blue lines correspond to the skis and the body of the jumper. The red circle represents the approximated feet point, which is computed as the intersection of the two lines.

2.5 Automatic measurement using measurement lines

Once the feet point is calculated, we use the measurement grid to find the measurement lines before above and below it. Each measurement line has a distance associated with it (e.g. 20 meters). We use linear interpolation with respect to the closest two measurement lines to calculate the precise distance at the *feet point*. Formally, the distance is calculated based on measurement line distances p_1 and p_2 as well as the euclidean distances d_1 and d_2 from the *feet point* to these lines along the

grid direction vector:

$$x = \frac{d_2}{d_1 + d_2} \cdot p_1 + \frac{d_1}{d_1 + d_2} \cdot p_2 \quad (1)$$

The values d_1 and d_2 correspond to the difference between T, P_1, and P_2. These points and lines are visualized in figure 5. The *grid direction vector* is based on *local direction vectors*. A *local direction vector* is computed based on two neighboring measurement lines, which are represented with their left and right points L_1, R_1 and L_2, R_2 as follows:

$$\mathbf{v_{1,2}} = \frac{\overrightarrow{L_1L_2} + \overrightarrow{R_1R_2}}{2} \quad (2)$$

The grid direction vector is computed as the component-wise weighted sum of local direction vectors. The grid direction vector should be more similar to local direction vectors, which correspond to measurement lines that are close to the feet point than to those further away. This reduces the degree to which imprecisely placed measurement lines affect the final distance. The values d_i and d_{i+1} denote the distance from the measurement lines i and $i+1$ to the feet point. The weights are calculated as follows:

$$w_{i,i+1} = e^{\left(-\frac{d_i + d_{i+1}}{2}\right)} \quad (3)$$

$$w'_{i,i+1} = \frac{w_{i,i+1}}{\sum_{i=1}^{n-1} w_{i,i+1}} \quad (4)$$

The weights are greater for local direction vectors, which are closer to the feet point. We use them to compute the grid direction vector:

$$\mathbf{v} = \sum_{i=1}^{n-1} w'_{i,i+1} \cdot \mathbf{v_{i,i+1}} \quad (5)$$

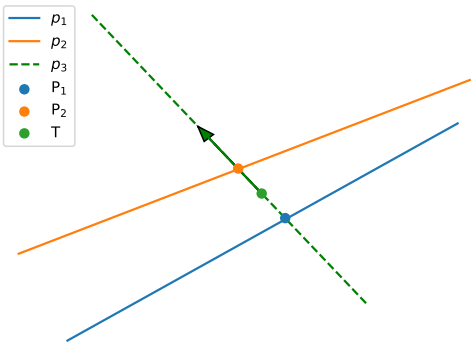

Figure 5: Visualization of the points and lines, used in the precise distance computation. The lines p_1 and p_2 correspond to measurement lines with their associated distances (e.g. 10 and 12 meters). The point T corresponds to the feet of the jumper. The measurement grid direction vector and the point T are used to describe the line p_3. The intersections of lines p_1 and p_2, as well as p_2 and p_3 are points P_1 and P_2, which use in the linear interpolation to obtain a more precise distance measurement.

2.6 Data

For 330 ski jumps recorded in junior competitions within PKP [1] and ŠIPK projects [2], we acquired the official results (measured by umpires using the eyes-only manual method). Ski jumps were recorded on smaller hills (HS 20-30m) and on average consisted of 36 frames recorded in HD resolution (1280×720 pixels) at 50 FPS. In order not to obstruct umpires' view, the camera was placed lower than the umpires. Each official measurement was further augmented by a ski-jumping professional coach using manual video measurement. For testing purposes the data was split to folds in a stratified manned, so that all frames belonging to a ski jump were assigned to the same fold (either for training or for testing).

3 Results

3.1 Landing frame detection results

The CNN by itself performed landing prediction relatively well. It was compared with a naive method which always predicted the landing to be in the middle of the sequence. The prediction results on an independent testing set are shown in a confusion matrix in table 1. Performing a 10-fold cross validation on the CNN results in an average accuracy of 0.922. There are two

Predicted class \ Actual class	*Air*	*Ground*
Air	995	56
Ground	122	1308

Table 1: Confusion matrix for the CNN predictions.

neurons in the last layer of the network, each outputting a value between 0 and 1. With normalization we get probabilities P_{Air} and P_{Ground} which correspond to the two possible classes. The landing is determined at the first frame where $P_{Air} < 0.5$. We can observe that the model often output probabilities of about 0.5 for frames around the true landing frame (figure 6).

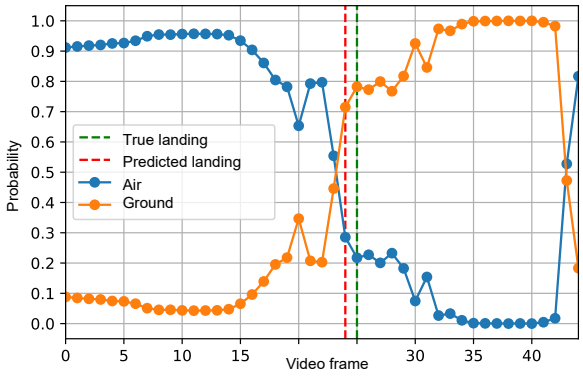

Figure 6: Sequence of predicted class probabilities on a test video. The model is initially very certain that the jumper is in the air. The value P_{Ground} increases as P_{Air} decreases. The landing is predicted as soon as P_{Air} first drops below 0.5.

3.2 Determining the landing distance

We considered several scenarios in order to evaluate important aspects of the system (table 2). First we evaluated the landing distance computation procedure independently of the landing frame predictions. This meant using the actual landing frames in the evaluation, which yielded the mean absolute error $MAE_{dist} = 0.404$ m. The error is larger when the jumper lands at the top of the landing area since the measurement lines are placed more densely together, a consequence of the camera placement very close to the hill and lower than umpires.

Jump distance	*MAE*
On true landing frame (testing set)	0.404
With landing frame detection (testing set)	0.586
With landing frame detection, 10-fold CV	0.946
With landing frame detection and bias correction, 10-fold CV	0.785

Table 2: Mean absolute error for jump distance predictions in meters. The first row correspond to predictions based on the actual (true) landing frames. The second row corresponds to full pipeline predictions, based on predicted landing frames.

The full measurement pipeline consists of the landing frame detection (CNN) and the landing distance determination. Due to misclassification of landing frames the $MAE_{pipeline} = 0.586$ m is a higher. Both numbers were obtained on an independent testing set without problematic jumps.

We also performed a 10-fold cross validation using all 330 jumps and achieved the total error of $MAE_{CV} = 0.946$ m. The predicted distances (figure 8) are mostly too short, indicating a systematic error (bias). On a typical laptop (without utilizing the GPU), the entire prediction procedure takes about 0.84 seconds. A systematic error (bias) in the system is caused be-

Figure 7: Measurement system in action at the HS=109 m hill in Kranj (bottom) and HS=21 m hill in Mengeš (top), Slovenia. The laptop and the camera are connected using a PoE switch.

cause the camera is placed too low on the side of the landing hill. As in World Cup competitions, we can specifically focus on jumps that are long enough (at least 17 meters in our case). By accounting for the median error of 0.51 meters to these predictions, the MAE_{CV} is reduced to 0.785 meters. By only analyzing the jumps where the CNN correctly predicts the landing, we achieve the *MAE* of 0.25 meters.

4 Conclusion

Our evaluation shows great potential for automatic ski jumping distance measuring. Even in current limited configuration we can measure distances with 1 m precision (at most 2 frames off, which requires very little human intervention) in reasonable conditions on the relevant part of the hill, both on small and

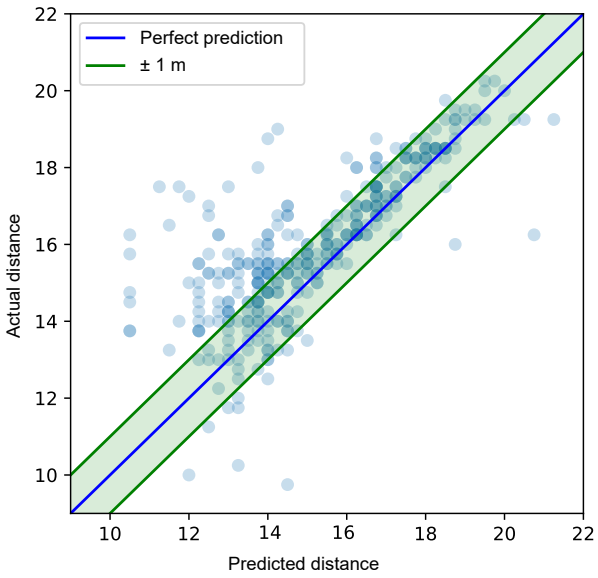

Figure 8: Distribution of the predicted and the true jump distances. The diagonal (blue line) indicates perfect predictions. Points within the green area refer to jumps where the prediction error is at most 1 meter. Points above the diagonal denote too short predictions.

large hills. While this is more than FIS-required 0.5 m, it is still a very useful addition. There is considerable interest from ski jumping clubs and Ski Association of Slovenia (SAS). For use on larger hills, slight modification of software are planned in order to allow for several cameras. The landing detection subsystem needs further testing under non-optimal (rain, snow) and artificial lighting conditions. We plan to achieve these aims in future partnership with SAS as we applied for co-founding from the Slovenian Foundation for Sports.

References

[1] T. Ciglarič et al. "Video meritve dolžin smučarskih skokov". In: *Zbornik šestindvajsete mednarodne Elektrotehniške in računalniške konference ERK 2017* (2017), pp. 337–340.

[2] M. Kukar. "Evaluation and Prospects of Semi-Automatic Video Distance Measurement in Ski Jumping". In: *Proceedings of the 21st International Multiconference - IS 2018* (2018), pp. 62–65.

[3] *FIS: The International Ski Competition Rules (ICR)*. `https://fis-ski.com`. Accessed: 9. 8. 2019.

[4] *FIS: Guidelines to Video Distance Measurement of Ski Jumping 2011*. `https://fis-ski.com`. Accessed: 9. 8. 2019.

[5] N. Sato, T. Takayama, and Y. Murata. "Early Evaluation of Automatic Flying Distance Measurement on Ski Jumper's Motion Monitoring System". In: *2013 IEEE 27th International Conference on Advanced Information Networking and Applications*. IEEE. 2013, pp. 838–845.

[6] Z. Živković. "Improved adaptive Gaussian mixture model for background subtraction." In: *ICPR*. 2004, pp. 28–31.

[7] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[8] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395. ISSN: 0001-0782. DOI: `10.1145/358669.358692`.